NASA Conference Publication 3295

Fourth NASA Goddard Conference on Mass Storage Systems and Technologies

Edited by
Benjamin Kobler
Goddard Space Flight Center
Greenbelt, Maryland

P.C. Hariharan
Systems Engineering and Security, Inc.
Greenbelt, Maryland

Proceedings of a conference held at
the University of Maryland
University College Conference Center
College Park, Maryland
March 28 - 30 1995

National Aeronautics and
Space Administration

Scientific and Technical
Information Branch

1995

Fourth NASA Goddard Conference on Mass Storage Systems and Technologies

Program Committee

Production, Copy Editing, and Layout

Len Blasso

This publication is available from the NASA Center for AeroSpace Information,
800 Elkridge Landing Road, Linthicum Heights, MD 21090-2934, (301) 621-0390.

Preface

This volume is a compilation of those presentations of the *Fourth Goddard Conference on Mass Storage Systems and Technologies* for which manuscripts or viewgraphs were received in time for publication. The Mass Storage community, as in the past, has shown enthusiasm for the Conference, as witnessed by the large number of excellent papers we have received. We are planning to bring out, on CD-ROM, the panel discussions and the after-dinner speech some time after the Conference.

The IEEE Mass Storage Systems Reference Model Version 5, on which the final vote was taken in July, 1994, will provide the framework for the definition of interfaces and standards. It is to be hoped that, as this activity is completed, it will result in the creation of interoperable software and hardware components from multiple vendors, thus affording users a truly wide selection of products with which both large and small archives can be built. 1994 also saw the definition of the architecture of the NASA Earth Observing System Data and Information System (EOSDIS). It embraces a federated design with autonomous, cooperating, systems which are distributed, both geographically and logically, and will be based, to a large extent, on COTS products. Unfortunately, most such products either do not exist today, or have not reached the required level of maturity. The standardization activities of the IEEE SSS Working Group should provide the basis and impetus for the creation of interoperable products for which the markets are emerging.

On a sad note, we regretfully report the passing away of three distinguished colleagues who had participated in previous mass storage conferences. We extend our condolences and sympathies to the families of:

John Corcoran, Ampex
L. Harris Robinson, Datatape
W. Cliff Brown, NASA Goddard Space Flight Center

In keeping with tradition, this year's Conference provides talks and presentations on:

- New storage technology from IBM, Quantum, Exabyte, Storage Technology and Primelink Technologies
- Stability of recorded media
- Performance studies
- Storage systems solutions
- The national information infrastructure, the Infobahn
- The future for storage technology
- Lessons learned from various projects
- The requirements of various agencies

The editors are grateful to:

Mr Jean-Jacques Bedet, Hughes STX Corporation
Mr John H Berbert, NASA/GSFC
Mr Jimmy F Berry, National Security Agency
Mr William Callicott, recently retired from NOAA
Dr Samuel Coleman, Lawrence Livermore Laboratory
Mr Ray Cook, Community Management Staff
Prof Charles Dollar, University of British Columbia

Ms Fynnette Eaton, National Archives and Records Administration
Mr Bernard O'Lear, National Center for Atmospheric Research
Dr Terry Pratt, former Director of CESDIS/USRA
Dr Sanjay Ranade, Infotech SA
Mr Bruce Rosen, NIST
Mr Don Sawyer, NASA/GSFC

who, as members of the Program Committee, worked diligently to make this Conference a success.

We also offer our thanks to:

Jorge Scientific Corporation, for logistics support; the staff at the University of Maryland Conference Center for the excellent conference facilities; and Len Blasso and Media Specialist Associates for the production and camera-ready preparation of this publication.

P C Hariharan
Systems Engineering & Security, Inc

Ben Kobler
Code 505, NASA Goddard Space Flight Center

Table of Contents

Applications Drivers For Data "Parking" On The Information Superhighway

Clark E. Johnson, Jr.
Research Program on Communications Policy
Massachusetts Institute of Technology
P.O. Box 50116
Minneapolis, MN 55405
clark@farnsworth.mit.edu
Tel: 612-922-8541
Fax: 612-922-8820

Thomas Foeller
Managing Director
Configured Health Care, Inc.
RR1, Box 111
Balsam lake, WI 54180
71732.243@compuserv.com

Abstract

As the cost of data storage continues to decline (currently about one-millionth of its cost four decades ago) entirely new application areas become economically feasible. Many of these new areas involve the extraordinarily high data rates and universal connectivity soon to be provided by the National Information Infrastructure (NII).

The commonly held belief is that the main driver for the NII will be entertainment applications. We believe that entertainment applications as currently touted--multi-media, 500 video channels, video-on-demand, etc.--will play an important but far from dominant role in the development of the NII and its data storage components. The most pervasively effective drivers will be medical applications such as telemedicine and remote diagnosis, education and environmental monitoring. These applications have a significant funding base and offer a clearly perceived opportunity to improve the nation's standard of living.

The NII's wideband connectivity both nationwide and worldwide requires a broad spectrum of data storage devices with a wide-range of performance capabilities. These storage centers will be dispersed throughout the system. Magnetic recording devices will fill the vast majority of these new data storage requirements for at least the rest of this century.

The storage needs of various application areas and their respective market sizes will be explored. The comparative performance of various magnetic technologies and competitive alternative storage systems will be discussed.

OVERVIEW

Evolving local and wide-area networks are opening and supporting increasingly wider interoperation and data sharing. A number of these architectures also support real-time inter-operable applications. Examples include on-line interactive games played over the Internet.

New commercial ventures are subjecting Internet facilities to increasing numbers of users with a wide variety of knowledge and skill levels. In addition, certain forthcoming Internet-based offerings (e.g. Delphi) will routinely use "agents" to traverse the Internet. These "agents" will be able to call up remote resources that can be aggregated to produce and deliver the desired result transparently as a "single" transaction.

Often times, valuable data artifacts resulting from or derived during these "aggregated" agent-driven operations are left at remote nodes for periods of time transcending the life of the transaction itself. These may create problems for both data owners and site operators where the data resides. Site operators will need to flush out such data from their storage devices from time to time, causing potential losses, without recourse, to the data owners.

To some degree, this same problem is already occurring at an increasing rate on smaller, local domains. Enterprise-wide networking that supports interoperable desktop computing often exhibits a potential soft underbelly of potential data loss. This is because there is no consistent nor convenient way to back-up widely dispersed data.

While there are currently a number of initiatives to develop and refine data-compacting interoperability models, there are none that we are aware of that are addressing back-up of widely dispersed data often typified by incompatible formats and inconsistent access mechanisms.

As much higher data rate networks such as the Global Information Infrastructure (GII) are developed that operate at speeds that are expected to approach a gigabit/second, the problems outlined above become intractable with current storage technology and system architecture. In order for such networks to become consistently useful and fulfill the promise of universal access to all, requires that dispersed data, located at nodes anywhere in the world be seamlessly available.

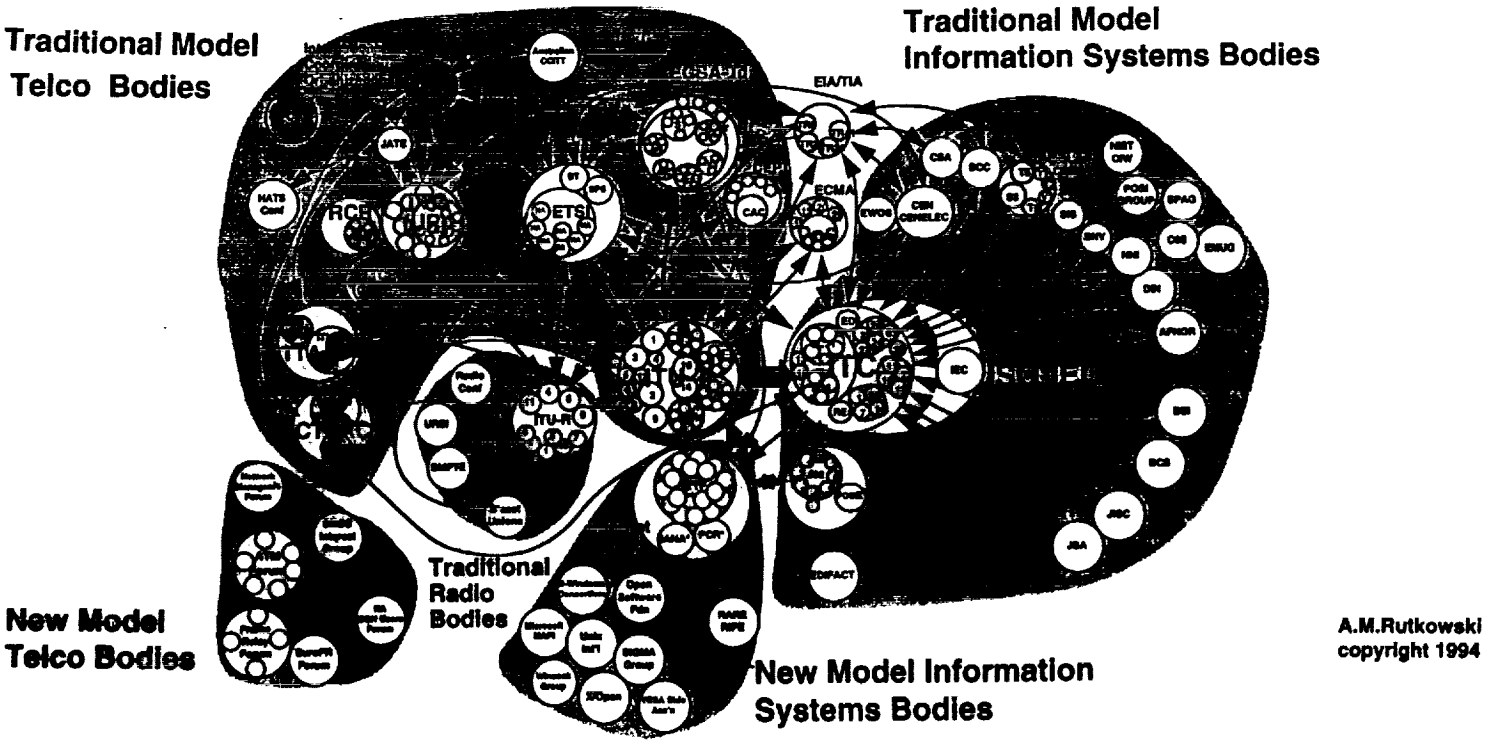


Figure 1. The Standards-Making Universe

This paper is a "first step" toward defining and formalizing a "universal" storage and retention model. We hope that this will encourage dialogue regarding the broad issues of the requirements of a truly user-friendly network, and to encourage the appropriate standards-setting groups to become actively involved. International standards setting is a convoluted process as Figure 1 shows.

SOME RELEVANT DRIVERS

Considerable effort is being made on a number of fronts to deal with the entire area of interoperability across dissimilar networks, user interfaces and underlying protocols. As these developments become commercialized, they will have a profound effect on the nature and structure of storage. Examples of the development efforts follow.

Most network management systems protocols and descriptive semantics, e.g. NMS (Novell), Openview (HP), Netmanager (SUN), Landesk (INTEL), Netview (IBM) do not currently interoperate across public network facilities such as the Internet, but they do operate across a number of commercial local and wide-area networks. Work is going on to provide upgrades that will soon enable certain user-specific backup and maintenance functions across heterogeneous networks.

Developments such as Microsoft "Windows '95" are expected to provide universal user interfaces including X-Windows along with the simultaneous support of multiple network transport mechanisms (e.g. TCP/IP, NetBIOS, netBEUI,...).

There are a number of applications program initiatives that will vie to become a "standard," and will enable application program modules written in one language to interoperate to some degree with programs written using one or more modules written in as many languages. These could, for example, share run-time resources in a single memory image; they could be segmented--running in separate memory images on different machines; or they might be distributed system components implemented by a number of programs on many machines. All of these implementations would appear seamless to the user; in fact, users may not know which were implemented and stored locally or at a remote node. As a consequence, slow or unreliable network transport mechanisms, inadequate long-term storage facilities located at network nodes, or missing software/data modules removed because of conflicting space requirements will increase and exacerbate real and perceived problems. Of course, increasing the number of interoperable components or modules expected by the user to be immediately available, will further increase the demand for high-performance storage.

BACKUP ISSUES

The above considerations lead to a number of data backup issues. As an example, consider the implementation of a flexible communication network backup facility that supports large block transfers of variable size and that functions without impacting the network's perceived performance by its on-line users. This is particularly important for digital video applications where even short time delays are intolerable.

In summary, because of this and myriad other issues, the overall storage architecture requires logical data space management so that it appears physically unbounded--regardless of the nature of the storage devices or their network topology. That is, object boundaries span storage hierarchies and media groupings. Consistent appearing data output formats are required so that the presentations are independent of origin data base, operational software and transport and are fully controllable by the user.

In addition, "agent" specifications and qualifications must be codified as there are compelling reasons to believe that these agents will be used to handle event log monitoring. billing (for example when the network is used to distribute intellectual property), reporting and internal data migration pathway control.

THE HIGH-SPEED NETWORK

Called the National Information Infrastructure in the United States, and the GII elsewhere, the vision is to have a "fat pipe" with Terabits/second bandwidth connecting major world centers by the next millennium. Currently, single-mode fiber-optic cables span the Atlantic and Pacific and operate at hundreds of megabits/second speeds. Terabytes of data per day will be traveling on this network and will require store-and-forward and destination node storage. Storage periods required will range from milliseconds to days.

Currently, Broadband Integrated Services Digital Network (BISDN), using the Synchronous Optical Network (SONET) physical layer protocol (SONET OC-3 operates at 155Mb/sec.) appears closest to commercialization, but requires three technological advances: in access, switching and storage.

In the area of storage, teleconferencing and multimedia applications such as MOSAIC can be accommodated on ISDN using commercialized bandwidth compression schemes. The storage requirements for applications such as medical image record transfer, where compression may not be acceptable to the profession, and for high-definition television are almost overwhelming.

For example, using the Qualcomm HDTV compression algorithm that provides 48:1 intra-frame compression, a full-length movie requires about 50 megabytes of storage. The average video store may have 10,000 titles in stock. To duplicate this on the server requires about 500 Terabytes of storage. These titles while not being required to be on one server, must be accessible under the same naming conventions, using consistent accessing and transfer methods, as if they were co-resident. Conventions and software required to meet these types of requirements are not yet available.

Missing or ill-defined technologies are not the only problems that must be faced. The economics of "video-on-demand" (VOD) is not comforting. The top one percent of Blockbuster customers spend $350/year. The average is about $50/year. Historically, citizens have spent a relatively fixed percentage of their income on entertainment--about 5% of their net after taxes; and this includes travel, restaurant meals and the like. It is questionable if, with currently available technology, there is a profitable business in downloading movies. The future availability of advanced, lower cost, faster storage means may well determine the fate of video on demand.

Medical image transmission and other telemedicine components are not yet inhibited by the same cost constraints as VOD. The pictures, which range from X-rays to magnetic-resonance images and sonograms, are sent without compression at relatively high cost because the medical profession (not to mention its regulators and the legal profession) are not yet willing to accept images reconstructed from image data that had been compressed using "lossy" compression algorithms--even though it can be shown that trained observers can rarely distinguish between the original images and those that have been compressed. Regardless of absolute cost, the value of using this technology to provide expert diagnosis to remote and rural areas cannot be overstated. Additionly, telemedicine using actual, high-resolution video

images provides the opportunity to perform almost real-time consultation and cooperative surgical procedures across the network.

America's schools are slowly becoming connected to each other and to world-wide data bases. While each institution's network usage, even if in multimedia format, will not put a significant demand on the network; the contemporaneous demand by many thousands of users will. A significant increase in response time when surfing the network for information using, for example Mosaic, will tend to discourage the students from using of the network's vast resources.

THE FUTURE OF STORAGE FOR THE HIGH-SPEED NETWORK

Certainly for the rest of this century, magnetic storage will dominate. The storage density continues to increase at the four decade-long historic rate of doubling every 2.5 years. In fact, this rate-of-change has recently increased. The per-bit cost of magnetic storage has gone down by a factor of over a million during the same time period. The retail price of disk storage is now roughly 40 cents/megabyte.

The always-increasing I/O problem (the mismatch between computational speed and memory access) is constantly being addressed by memory suppliers. The development of RAID (redundant array of inexpensive drives) disk technology and stripe-based tape storage provide performance and reliability improvements. Massive RAID systems now being developed are currently limited by the speed of silicon technology.

The usual hierarchy of storage will continue to prevail with the ultra-high speed cache requirements being fulfilled by solid-state memory. Massive caching is an economic issue because the cost of such memory is prohibitive for high data rate network applications. Thus, extensive effort is being devoted to refining and improving holographic storage technologies. Many researchers expect these to be commercialized by the end of the century and will combine the speed performance of RAM storage with the low-cost and capacity of magnetic-based systems. A particular attractiveness is the lack of moving parts and storage densities in the range of terabytes/cc.

The Design of a Petabyte Archive and Distribution System for the NASA ECS Project

Parris M. Caulk
LORAL Aerosys
1616A McCormick Drive
Landover, MD 20785
pcaulk@eos.hitc.com
Tel: 301-925-0610
Fax: 301-925-0327

Introduction

The NASA EOS Data and Information System (EOSDIS) Core System (ECS) will contain one of the largest data management systems ever built - the ECS Science and Data Processing System (SDPS). SDPS is designed to support long term Global Change Research by acquiring, producing, and storing earth science data, and by providing efficient means for accessing and manipulating that data. The first two releases of SDPS, Release A and Release B, will be operational in 1997 and 1998, respectively. Release B will be deployed at eight Distributed Active Archiving Centers (DAACs). Individual DAACs will archive different collections of earth science data, and will vary in archive capacity. The storage and management of these data collections is the responsibility of the SDPS Data Server subsystem. It is anticipated that by the year 2001, the Data Server subsystem at the Goddard DAAC must support a near-line data storage capacity of one petabyte.

The development of SDPS is a system integration effort in which COTS products will be used in favor of custom components in every possible way. Some software and hardware capabilities required to meet ECS data volume and storage management requirements beyond 1999 are not yet supported by available COTS products. The ECS project will not undertake major custom development efforts to provide these capabilities. Instead, SDPS and its Data Server subsystem are designed to support initial implementations with current products, and provide an evolutionary framework that facilitates the introduction of advanced COTS products as they become available.

This paper provides a high-level description of the Data Server subsystem design from a COTS integration standpoint, and discusses some of the major issues driving the design. The paper focuses on features of the design that will make the system scalable and adaptable to changing technologies.

SDPS Overview

SDPS [1] will support the services required to ingest, process, archive, manage, and access science data and related information from the entire EOSDIS. A typical DAAC will consist of the following SDPS components :

An Ingest subsystem for acquiring all data from EOS instruments, NASA Probe flight missions, and other remotely-sensed data.

A Data Processing subsystem for the generation of science data products from ingested instrument data, and from previously stored data products. The Data Processing subsystem executes hundreds of science algorithms which generate hundreds of gigabytes of science data on a daily basis. The subsystem generates the bulk of the data stored into the archive, and accounts for a large portion of the retrieval demand for archived data.

A Planning subsystem for planning the execution of data processing tasks.

A Data Management subsystem that provides functions for locating and accessing data distributed among ECS DAACs and other data systems with which ECS interoperates.

A Data Server subsystem with a capacity (at the largest DAACs) to archive, retrieve, and distribute hundreds of gigabytes per day.

An Interoperability subsystem that advertises ECS data and services to ECS clients.

A Service-Oriented System

The design of SDPS is based on a service-oriented approach to data search and access. Predecessor systems have employed a "product-ordering" approach that derived from the file-oriented retrieval mechanisms provided by conventional hierarchical storage systems. The contemporary science user requires sophisticated search and access methods which can locate and manipulate required data using a variety of search and processing criteria. Normally, the science user will only be interested in a fragment of a data product. Subsetting and subsampling operations will be applied to ECS data after it is retrieved and before it is distributed to the user. Ideally, all of the data would be stored inside of a data base system that could provide advanced search, access, and data manipulation capabilities on a petabyte of data in tertiary storage.

The service-based approach adopted by SDPS is intended to support the eventual introduction of advanced mass-storage data base technologies when they become available. The approach has the following features:

It provides the science user a set of services on earth science data, instead of providing copies of canned data products. The user issues service requests which typically involve the retrieval of stored data followed by one or more processing steps on that data. In effect, the user accesses data "objects" instead of data products.

It associates data and services according to science "views" of the data; usually a given view corresponds to a science discipline (e,g. Atmospheric Dynamics).

It provides a logical "Data Server" to provide each discipline-oriented view of the data. Each data server logically associates related data even though the data may be stored on a variety of physical devices and managed by different data management software. The science user accesses and manipulates the data as if it were stored and managed in the same manner.

It supports the experimental development and introduction of advanced user interfaces and advanced data management methods by providing application program interfaces to supported services.

The Data Server Subsystem

The SDPS Data Server subsystem provides the resources and services required for the storage, management, and access of the ECS science data collections. Figure 1 illustrates the subsystem and its relationship to the other parts of SDPS. The Data Server subsystem services data storage requests from the Ingest and Data Processing subsystems, and provides data search and retrieval services to ECS users and to the Data Processing subsystem. The subsystem manages the distribution of requested data to both hard media and to network clients.

Science users access the data and services provided by the Data Server subsystem by issuing service requests, either to the subsystem directly, or through the distributed access facilities of the Data Management subsystem. Users can request that processing (e.g. subsetting and subsampling) be performed on archived data products and have the result, or "result set", distributed to them. Additionally, the users can request processing on the result sets produced by previous service requests. The source data products for these operations can be many gigabytes in size. The capability to provide users the capability to perform processing at the archive site is essential because the size of the input data products is so large and the bandwidth of the distribution network is limited.

Physically, the Data Server subsystem is composed of the following major components:

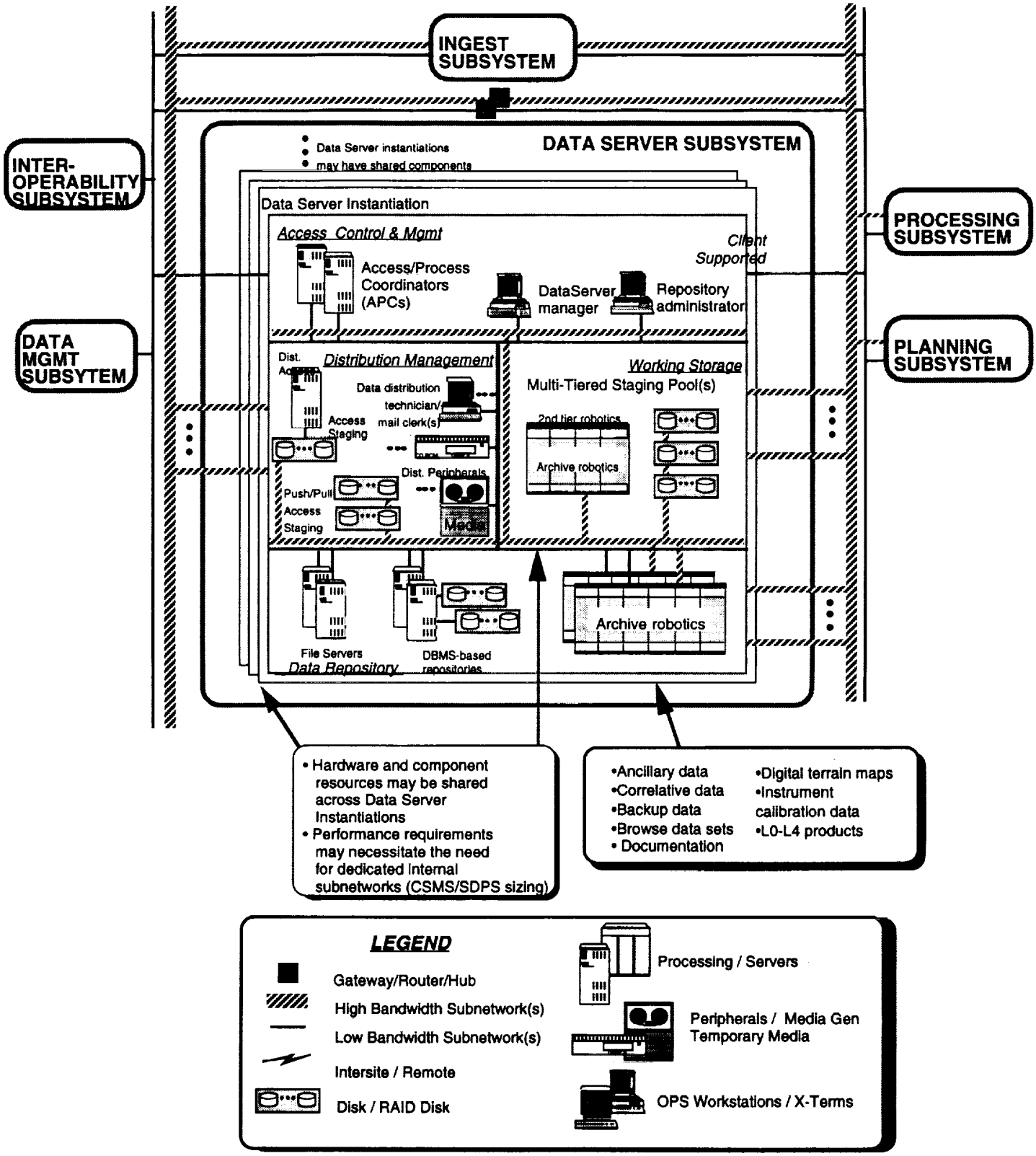


Figure 1 SDPS and the Data Server Subsystem

Access components - These support user/client access to data and services; they manage client sessions and manage the execution of service requests; and they advertise the services performed by the subsystem.

Data Repository - This component provides storage servers for the storage, searching, and retrieving of data. A storage server can be any software/hardware subsystem that stores and retrieves data on demand.

Distribution - This component supports the distribution of ECS data to users via networks and through the generation of hard media..

Working Storage - This component manages the resources required for the temporary storage and buffering of ECS data.

The Access Component

The Access component supports client access to ECS data and services. The client may be an interactive application system that is controlled directly by the user, an intermediary process provided by the SDPS Data Management subsystem, or a client belonging to the Data Processing subsystem. The client issues service requests for data and services on behalf of the user; the Access component manages the execution of the requests and directs them to appropriate Data Server or Data Processing subsystem components. Storage and retrieval requests are directed to storage servers (see Figure 2). Processing requests are executed directly by the Access component, or are directed to the Data Processing subsystem.

From the client perspective, the client is not interacting with the Access component or the storage servers, but is accessing a Data Server dedicated to a set of data and services that apply to a particular science discipline. The Access component can implement the logical functionality and services of multiple Data Servers. Data logically associated with a single Data Server may be physically stored across multiple storage servers.

A major function of the Access component is to insulate the client from the mechanisms used to store and retrieve data. In retrieval operations, the user specifies the data he wants and generates a service request containing a unique ECS Universal Reference (UR) to the data. The format of the UR is independent of the storage server and its underlying storage technology. The Access component uses the UR to determine which storage server contains the required data. The Access component then obtains the required data by directing the request, in a standard format, to the storage server.

Data Repository

The Data Repository component provides the storage servers for the storage, searching, and retrieving of data. A storage server provides permanent data storage and retrieval services. It is composed of a hardware storage system and a COTS storage management software system. The storage management software manages the hardware and data, and

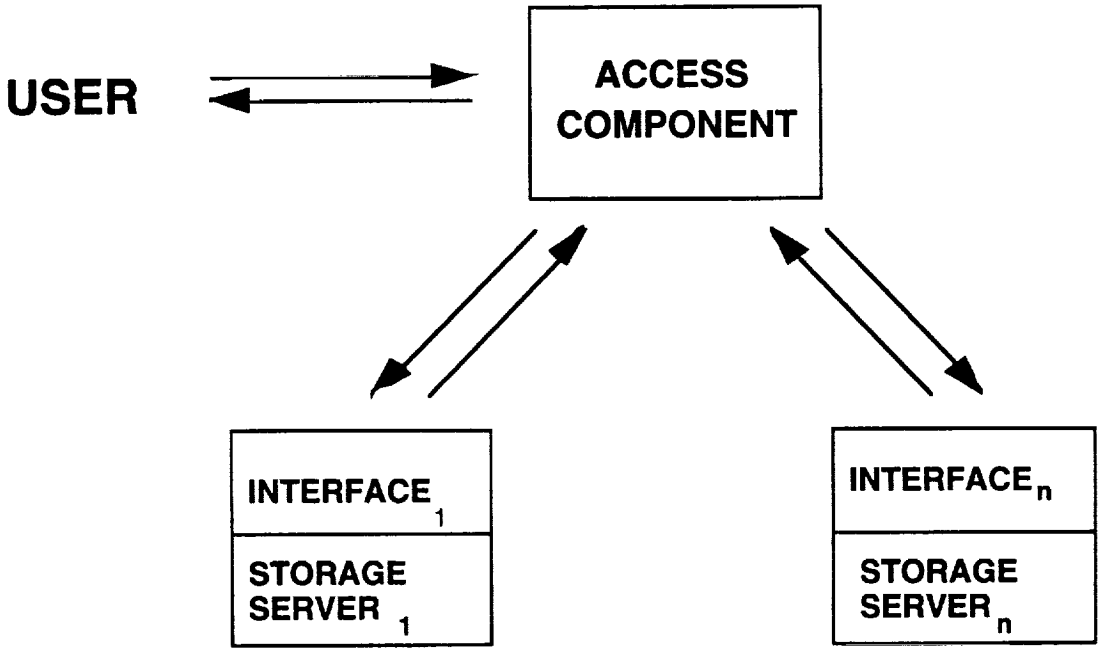


Figure 2. Access to Storage Servers

processes service requests. Examples of storage servers include DBMS servers and tertiary storage systems managed by COTS File Storage Management products.

The vast majority of ECS data will be stored using technologies which support high storage densities at the lowest cost. At this writing the ECS project has not selected the storage systems for any of its releases. However, this paper assumes that one or more robotic tape libraries will be employed in the storage servers that store the bulk of ECS data.

The COTS software packages available for managing storage servers use different mechanisms for identifying and accessing data. Each storage server will have a custom software "wrapper" to translate standard service requests, from the Access component, into service requests that are compatible with the underlying COTS storage management software. Part of the translation process will necessitate the translation of the ECS UR into a product-dependent data set (or file) identifier. A storage server could be replaced with another one based on different COTS products without affecting the ECS data identification scheme, provided a translation interface is built for the new server.

The Access and Data Repository components support the use of multiple and heterogeneous storage servers. This design reduces dependence on a single storage technology. The design allows the use of a mix of storage technologies tailored to the variety of data that must be stored. It supports system evolution by permitting new technologies to be introduced and old ones replaced, in a gradual manner. Finally, it supports the expansion of the storage system through the addition of storage servers.

Working Storage

The Working Storage component provides high-performance disk storage, i.e. "working storage," for the temporary storage of ECS data. Specifically, it (1) stores files retrieved from Storage servers, (2) stores the result sets generated by the Access component and the Data Processing subsystem, and (3) buffers data to be inserted into the storage servers. Working storage roughly corresponds to the secondary storage component in a conventional hierarchical storage system.

One of the uses of working storage is to support interactive sessions with users. During these sessions, result sets are generated by the execution of service requests and are placed in working storage. The result sets may be browsed, processed further, or distributed as directed by a subsequent service request. The Access component must be able to retain result sets in working storage until the session with the user is terminated.

The Data Processing subsystem requires a large portion of working storage in order to support the execution of hundreds of science algorithms daily. The amount of working storage required to support the Data Processing subsystem is a major cost driver for the development of SDPS. The execution of a single science algorithm may process multiple gigabytes of input data contained in multiple files, and may generate many gigabytes of output data.

To a significant degree, the output of one algorithm is used as input for the execution of another algorithm within a few minutes or hours. Ultimately, most of the algorithm output will be archived. The retrieval load on the archive can be significantly reduced if algorithm output can be retained in working storage long enough to be used as input by algorithms that require that output. The amount of working storage required to support processing depends on the required retention intervals for algorithm outputs. Algorithms are scheduled in order to minimize these intervals. The efficient management of working storage requires that the Data Processing subsystem control working storage file retention in coordination with algorithm scheduling.

Mass-Storage I/O Management

The greatest design challenge for the Data Server subsystem is the management of the massive I/O (multiple terabytes per day) between the mass-storage library and the Data Processing subsystem. The Data Server design approaches this problem by supporting the scaling of I/O capacity and the intelligent management of working storage.

The conventional approach to managing multi-terabyte mass storage is to use a COTS File Storage Management System (FSMS) hosted on a single supercomputer. All I/O must pass through the FSMS host. Scalability is achieved under this approach in a "vertical" fashion by expanding the power of the host computer. This approach will not meet long-term ECS requirements for multi-terabyte daily I/O throughput rates. Ideally, I/O to and from the media drives in the mass-storage archive would be conducted from and to working storage, along parallel I/O paths which bypass the FSMS host. Current FSMS products do not support all of the capabilities required to implement this I/O architecture, nor do they meet ECS requirements for the application control of working storage. The ECS project is anticipating the development of FSMS products that will meet these requirements.

The ideal FSMS product would have the following major features:

A volume management capability for controlling a variety of storage devices, including robotics.

A file management capability that formats and organizes files on tertiary media.

A capability to automatically monitor the integrity of the data in tertiary storage and monitor condition of the media.

The capability to direct I/O between (1) the media drives and working storage and (2) between the media drives and a specified computer, without traversing the FSMS host.

An application interface to working storage that allows the application to control the staging and retention of files in working storage.

The ECS near-term approach to scalability is to increase volume and throughput capacity "horizontally" by replicating storage servers supported by processors of moderate size. The current COTS-based design is intended to support the initial Releases of ECS and to support the introduction of high-performance I/O technologies later on.

Figure 3 illustrates a model for the initial implementation of a mass-storage storage server that (1) could be implemented with available COTS products, (2) could be evolved to support advanced I/O and storage management capabilities, and (3) provides for application control of working storage. In the model, a COTS FSMS is used to manage the files in the mass-storage archive, control the archive hardware, and move files between a mass-storage archive and working storage. Working storage is implemented by a disk array managed by a COTS network-accessible file system.

The FSMS and the network-accessible file system are separate products that interact across a standard UNIX file system interface. The separation of these capabilities is necessary in order to provide ECS flexibility in the selection and replacement of corresponding COTS technologies.

An essential feature of the model is that the management of files in working storage can be controlled by components external to the FSMS. This allows the Access component and the Data Processing subsystem to control the retention of files in working storage and allows the Data Processing subsystem to direct the staging of files prior to the execution of science algorithms. External components exercise this control through an "ECS file manager". The ECS file manager does the following:

Accepts external requests for staging files, for retaining and releasing files from working storage, and for storing files in the archive.

Keeps track of which files are in working storage and assigns a retention interval to each file.

Directs the FSMS to migrate files between the archive and working storage, in response to external requests.

Conclusion

The Data Server subsystem design is constrained by the availability of applicable COTS technologies and the requirement to evolve to accommodate increasing system loads and advances in mass-storage technology. The efficient management of its storage resources require that the application have control over file caching mechanisms and over the movement of data between storage resources. Required support for multi-terabyte daily I/O rates necessitate the ultimate use of advanced I/O architectures. Currently no product (or family of products) provides a comprehensive approach to these issues. The current

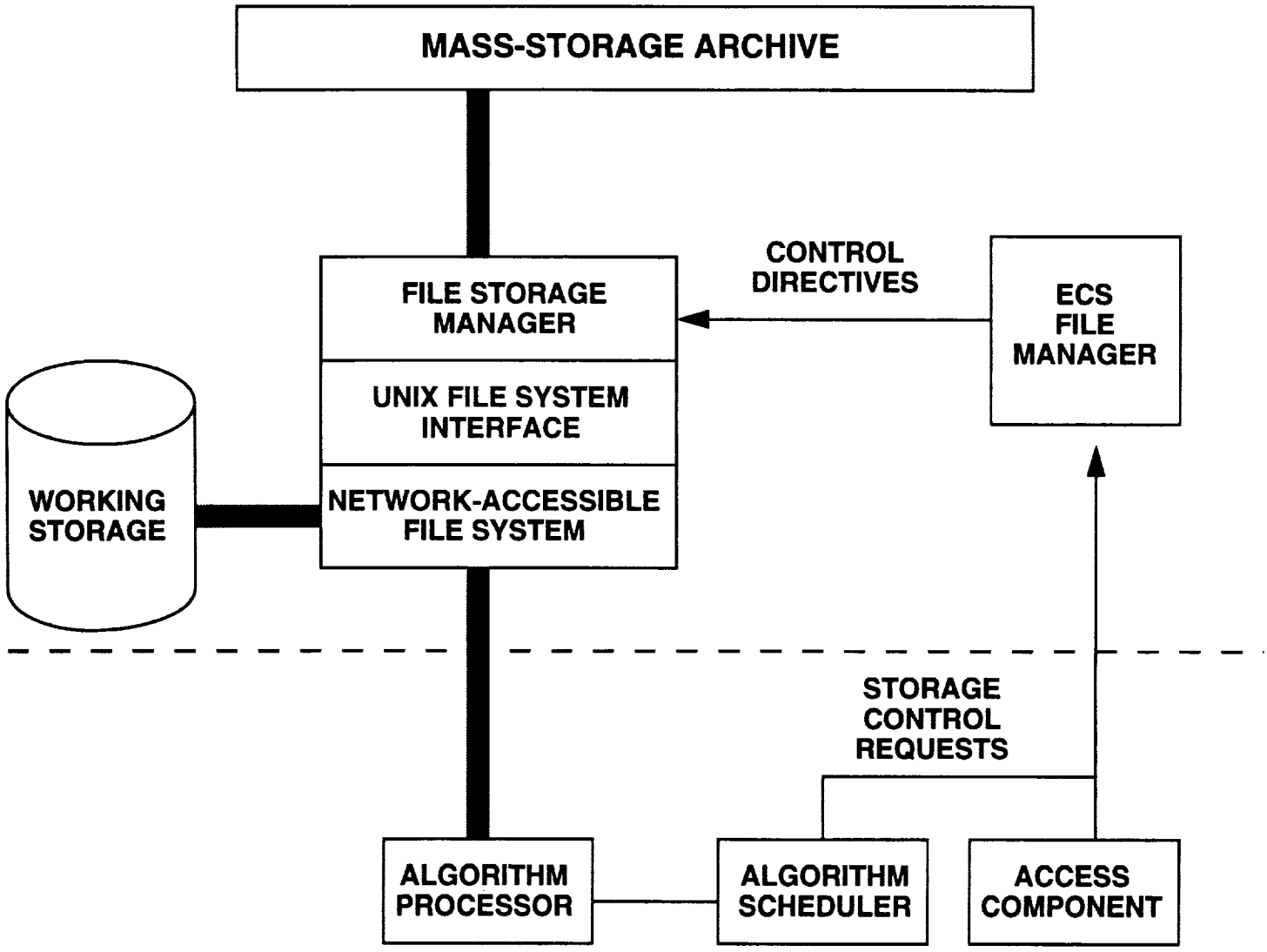


Figure 3. Initial I/O Management Approach

COTS-based design is intended to support the initial releases of ECS with a combination of COTS products, and will support the introduction of high-performance I/O technologies in later phases of the project.

Acknowledgments

This paper is based on the design efforts and helpful comments of Mark Huber, Tom Smith, and Alla Lake of LORAL Aerosys, and Eric Dodge and Evelyn Nakamura of Hughes Applied Information Systems.

References

1. EOSDIS Core System Project, "System Design Specification for the ECS Project" (194-207-SE1-001)

A Distributed Parallel Storage Architecture and its Potential Application Within EOSDIS

William E. Johnston and Brian Tierney
Lawrence Berkeley Laboratory[1]
University of California
Berkeley, CA, 94720

Jay Feuquay and Tony Butzer
EROS Data Center / Hughes STX
Mundt Federal Building, Sioux Falls, SD, 57198

Abstract

We describe the architecture, implementation, use, and potential use of a scalable, high-performance, distributed-parallel data storage system developed in the ARPA funded MAGIC gigabit testbed[1]. A collection of wide area distributed disk servers operate in parallel to provide logical block level access to large data sets. Operated primarily as a network-based cache, the architecture supports cooperation among independently owned resources to provide fast, large-scale, on-demand storage to support data handling, simulation, and computation.

1.0 Introduction

We have designed and implemented a wide area network-based, distributed-parallel storage system ("DPSS") as part of an ARPA funded collaboration known as the MAGIC gigabit testbed [1], and as part of DOE's high speed distributed computing program. This technology has been quite successful in the MAGIC environment, and it has the potential for enhancing data rich environments like EOSDIS (see [2] and Figure 7). The DPSS provides an economical, high performance, widely distributed, and highly scalable architecture for caching large amounts of data that can potentially be used by many different users and processes within EOSDIS. Our current implementation of the DPSS technology is called the Image Server System ("ISS"), and is optimized for providing access to large, image-like, read-mostly data sets such as those found in the environment of the EROS

1. The work described in this paper is supported by ARPA, Computer Systems Technology Office (http://ftp.arpa.mil/ResearchAreas.html) and the U. S. Dept. of Energy, Office of Energy Research, Office of Scientific Computing (http://wwwosc.er.doe.gov/), under contract DE-AC03-76SF00098 with the University of California. Reference herein to any specific commercial product, etc., does not imply its endorsement by the United States Government or the University of California. Likewise the views and opinions of authors expressed herein. Authors: wejohnston@lbl.gov (Lawrence Berkeley Laboratory, mail stop: B50B-2239, Berkeley, CA, 94720, ph: 510-486-5014, fax: 510-486-6363, http://www-itg.lbl.gov), tierney@george.lbl.gov, feuquay@sunh.cr.usgs.gov. Report no. LBL-36680.

Data Center (EDC) as the Land Processes DAAC. In the MAGIC testbed the ISS is distributed across several sites separated by more than 1000 Km of high speed IP over ATM network and stores very high resolution images of several geographic areas. The "TerraVision" terrain visualization application uses the ISS to let a user to explore / navigate a "real" landscape represented in 3D by ortho-corrected, one meter images and digital elevation models (see [3]). TerraVision requests from the ISS, in real-time, the sub-images ("tiles") needed to produce a view of the landscape. Typical use requires aggregated data streams of from 100 Mbits/sec to 400 Mbits/sec that are supplied from several servers on the network. Even in the current prototype system the ISS is easily able to supply these data rates.

The ISS architecture is that of multiple network disk servers that are based on Unix workstations. The system coordinates multiple servers to aggregate high-bandwidth data streams to network-based client application (e.g. TerraVision). Alternatively, many lower data rate streams can be supplied to many applications simultaneously (in a "video server" style of operation). The DPSS implementation uses an open systems, platform-independent, software approach. High performance is achieved in two ways: First, the functionality of the disk servers has been kept very simple - they are essentially "block" servers (a block being a fixed size unit of data like an image tile). Second, image data sets are easily partitioned over network distributed servers in such a way that ensures parallel operation of many independent servers in order to supply a high bandwidth data stream to an application.

The DPSS technology potentially fits into the EOSDIS environment in various ways. First, there are several uses that supplement EOSDIS, and that do not require direct integration into existing EOSDIS systems: For example, the DPSS might be used for buffering data coming into DAACs (data archive sites) prior to archiving, and it might be used as a large scale query results cache to support SCFs (data analysis sites). Second, DPSS technology also has potential use within the EOSDIS system itself: It could provide a mechanism at several points in the EOSDIS architecture for rapid reorganization of large volumes of data, and it might be used as a cache for high speed in-line processing operations.

We will describe the implementation, performance, and uses of the current prototype DPSS, including the operation of the ISS in the MAGIC testbed and its use in a regional medical imaging experiment.

2.0 Background

Current workstation disk technology delivers about four Mbytes/s (32 Mbits/s) per drive, a rate that has improved at about 7% each year since 1980 [4], and there is reason to believe that it will be some time before a single disk is capable of delivering streams at the rates needed for the applications mentioned. While RAID [4] and other parallel disk array technologies can deliver higher throughput, they are still relatively expensive, and do not scale well economically, especially in an environment of multiple network based users

where we assume that the sources of data, as well as the multiple users, will be widely distributed. Asynchronous Transfer Mode (ATM) networking technology, due to the architecture of the SONET infrastructure that underlies large-scale, wide area ATM networks, will provide the bandwidth that will enable the approach of using network-based distributed, parallel data servers to provide high-speed, scalable storage systems. Data transport is provided by IP datagram services (UDP and RTP) and high performance versions of TCP (see [5]).

The approach described here differs in many ways from RAID, and should not be confused with it. RAID is a particular data strategy used to secure reliable data storage and parallel disk operation. Our approach, while using parallel disks and servers, deliberately imposes no particular layout strategy (which is free to be optimized on an application or data structure basis), and is implemented entirely in software (though the data redundancy idea of RAID might be usefully applied across servers to provide reliability in the face of network problems).

3.0 System Architecture Overview

The Image Server System (ISS) is an implementation of a distributed-parallel data storage architecture. It is essentially a "block" server that is distributed across a wide area network and used to supply data to applications located anywhere in the network. Figure 1 illustrates the architecture. There is no inherent organization to the blocks; however, layout strategies that maximize parallelism are clearly desirable. The data organization is determined by the application as a function of data structures and access patterns, and is implemented during a data load process. When data structures and access patterns are well understood then specific placement algorithms can be designed to optimize data placement for maximum parallelism (e.g. see [6]). In other cases blocks can be scattered randomly across the disks and servers (a strategy that can work surprisingly well). The usual goal of the data organization is that data is declustered (dispersed in such a way that as many system elements as possible can operate simultaneously to satisfy a given request) across both disks and servers. This strategy allows a large collection of disks to seek in parallel, and all servers to send the resulting data to the application in parallel, enabling the ISS to perform as a high-speed image server.

The functional design strategy is to provide a high-speed "block" server, where a block is a unit of data request and storage. The ISS essentially provides only one function - it responds to requests for blocks. However, for greater efficiency and increased usability, we have attempted to identify a limited set of functions that extend the core ISS functionality while allowing support for a range of applications. First, the blocks are "named." In other words, the view from an application is that of a *logical* block server. Second, block requests are in the form of lists that are taken by the ISS to be in priority order. Therefore the ISS attempts (but does not guarantee) to return the higher priority blocks first. Third, the application interface to the ISS provides the ability to ascertain certain configuration parameters (e.g., disk server names, performance, disk configuration, etc.) in order to permit parameterization of block placement strategy algorithms (for example, see [6]). Addi-

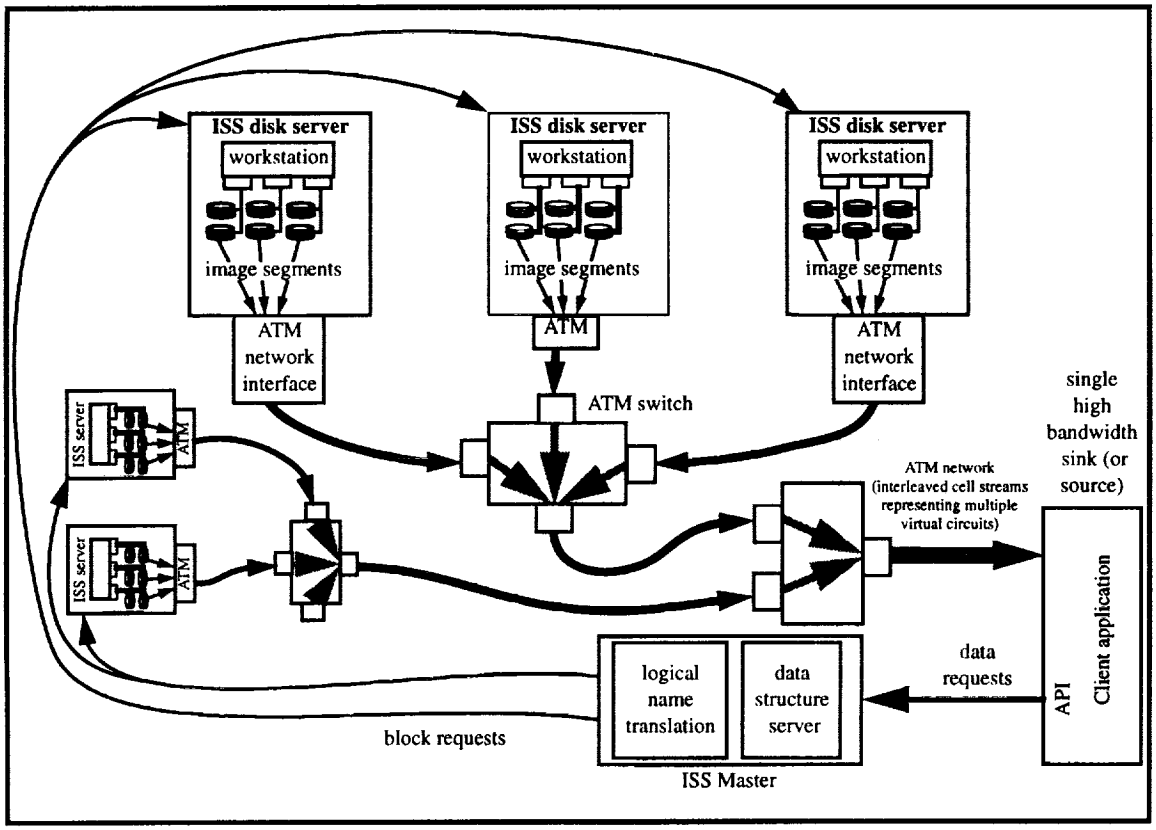


Figure 1 Distributed-Parallel Server System Architecture

tionally, the ISS is instrumented to permit monitoring of almost every aspect of its functioning during operation. This monitoring functionality is designed to facilitate performance tuning and network performance research. However, the information about individual server performance characteristics provided as part of this monitoring can be used by a client's data layout algorithm. Such performance information can facilitate a distribution of the data that better accounts for the differences between individual servers' demonstrated capabilities regardless of the cause: disk hardware, OS, location in the network, etc. Asymmetric server performance is accounted for in the image-tile placement algorithm used to support the TerraVision application in MAGIC.

At the present state of development and experience, the ISS that we describe here is used primarily as a large, fast, wide area network distributed "cache". Reliability with respect to data corruption is provided only by the usual OS and disk mechanisms, and data delivery reliability of the overall system is a function of user-level strategies of data replication and/or re-request and retransmission.

The data of interest (tens to hundreds of GBytes) is typically loaded onto the ISS from archival tertiary storage, or written into the system from live data sources. Data layout strategy is used when the organization of the data and the application use patterns are well

understood (as with images). In the case of writing from live data sources some variation of a "round-robin" scheme optimizes the speed of writing to the ISS.

Client Use

The client-side (application) use of the ISS is provided through a library-based API that handles initialization (for example, an "open" of a data set requires discovering all of the disk servers with which the application will have to communicate), and the basic block request / receive interface. It is the responsibility of the client (or, more typically, its agent) to maintain information about any higher-level organization of the data blocks, to maintain sufficient local buffering so that "smooth playout" requirements may be met locally, and to run predictor algorithms that will pre-request blocks so that application response time requirements can be met. The prediction algorithm enables pipelining the operation of the disk servers, with the goal of overcoming the inherent latency of the disks. (See [7] and [8]). None of this has to be explicitly visible to the user-level application, but some agent in the client environment must deal with these issues because the ISS always operates on a best-effort basis: if it did not deliver a requested block in the expected time or order, it was because it was not possible to do so. In fact, a typical mode of operation is that pending block requests are flushed from the server read queues when they age more than a few hundred milliseconds. The application routinely re-requests some fraction of the data. This deliberate "overloading" of the disk servers ensures that they will be kept busy looking for relevant blocks. This behavior is one aspect of the pipelining strategy on the servers.

Name Server Functions

The primary function of the name server is to translate the logical block names used by the applications into physical block names. Typical operation involves the application making an initial request of the name server for a particular data set and getting back a list of servers that will be supplying data. After the "open" operation, priority ordered logical block request lists are sent to the name server, which translates requests to physical block locations (disk server address, disk number, and disk block). The data is returned via the client's direct connections to individual servers. The name server ("ISS Master") also does housekeeping and monitoring functions, and these are described in [8]. One of the design decisions was that the name server only do logical block name translation. All other higher level information about the structure of the data (e.g., what list of blocks comprise a file) are relegated to a "structure server" mechanism that can maintain as complex a view of the data as is needed by the application (or even different views of the same data). We have not attempted to standardize the structure server (different applications can have very different ways of viewing their data), but several functions are provided by the name severs to assist the structure server.

Use of the DPSS approach for management of large data archives will be facilitated by the ability to rapidly reconfigure the scope and organization of the storage. The extent of a "unit of storage" (a logically associated collection of DPSS disk blocks) is only a function of the name server. Multiple name servers of storage will, in the future, share, request, or

relinquish servers via cooperation among the name servers operated by different organizations. No reorganization of the disk servers (internally or externally) is necessary. This ability will facilitate "just-in-time" configuration of cache storage for a large data set resulting from a query that, for example, extends across several DAACs, or in buffering large incoming data sets resulting from, for example, several sources turning on at the same time.

Implementation

In our prototype implementations, the typical ISS consists of several (four - five) UNIX workstations (e.g. Sun SPARCStation, DEC Alpha, SGI Indigo, etc.), each with several (four - six) fast-SCSI disks on multiple (two - three) SCSI host adaptors. Each workstation is also equipped with an ATM network interface. An ISS configuration such as this can deliver an aggregated data stream to an application at about 400 Mbits/s (50 Mbytes/s) using these relatively low-cost, "off the shelf" components by exploiting the parallelism provided by approximately five servers, twenty disks, ten SCSI host adaptors, and five network interfaces.

The software implementation is based on Unix interprocess communication mechanisms and a POSIX threads programming paradigm (see [9] and [10]). The three primary operating systems (Sun's Solaris, DEC's OSF, and SGI's IRIX) all have slightly different implementations of threads, but they are close enough that maintaining a single source is not too difficult.

The implementation supports a number of transport strategies, including TCP/IP and UDP/IP. UDP does not guarantee reliable data delivery, and never retransmits. Lost data are handled at the application level. This approach is appropriate when data has an age determined value. That is, data not received by a certain time is no longer useful, and therefore should not be retransmitted, as is true in certain visualization scenarios.

Prototypes of the ISS have been built and operated in the MAGIC network testbed. Other papers on the ISS are [11], which focus on the major implementation issues, [7], which focuses on the architecture and approach, as well as optimization strategies, and [12], which focuses on ISS applications and ISS performance issues.

Performance

Scalability of capacity and performance are inherent in the architecture: the individual servers are effectively completely independent of each other. The time spent locating blocks is minimal, and in principle (and frequently in fact), many servers can be sending blocks simultaneously to the application. In other words, the performance limits are typically at the client application. This architecture means that capacity and performance scale by simply adding more disk servers anywhere in the network. (Obviously some limits exist: network bandwidth will limit the aggregate throughput, if the number of servers exceeds the number of blocks in a file then adding servers will not increase the through-

put, etc.). The strategy of a centralized name server seems to add very little overhead compared to the time required to request and deliver blocks.

The current implementation of the servers is memory bandwidth limited, a situation common to almost all current workstation hardware architectures. Our implementation does no user-space copies of the data, which means a total of three memory copies for most OS's: disk to memory, and two copies to get to the network. The performance of the server then is typically the memory copy speed divided by three (a metric that has held for all of the six or eight platforms that we have tested. Table 1 shows performance measurements for

TABLE 1. ISS Disk Server Performance

System	Max ATM LAN *ttcp*	*ttcp* w/ disk read	Max ISS speed
Sun SS10-51	70 Mbits/sec	60 Mbits/sec	55 Mbits/sec
Sun SS1000 (2 proc)	75 Mbits/sec	65 Mbits/sec	60 Mbits/sec
SGI Challenge L	82 Mbits/sec	72 Mbits/sec	65 Mbits/sec
Dec Alpha 3000/600	127 Mbits/sec	95 Mbits/sec	88 Mbits/sec

several platforms. "ttcp" is effectively a memory-to-network copy, and the ISS numbers include the overhead for locating blocks and moving them from disk to network.

For more specific performance analysis of the current system, see [12].

4.0 Related Work

There are other research groups working on solving problems related to distributed storage and fast multimedia data retrieval. For example, Ghandeharizadeh, Ramos, et al., at USC are working on declustering methods for multimedia data [13], and Rowe, et al., at UCB are working on a continuous media player based on the MPEG standard [14]. Similar problems are also being solved by the Massively-parallel And Real-time Storage (MARS) project [15], which is similar to the ISS, but uses special purpose hardware such as RAID disks and a custom ATM Port Interconnect Controller (APIC).

In some respects, the ISS resembles the Zebra network file system, developed by John H. Hartman and John K. Ousterhout at the University of California, Berkeley [16]. However, the ISS and the Zebra network file system differ in the fundamental nature of the tasks they perform. Zebra is intended to provide traditional file system functionality, ensuring the consistency and correctness of a file system whose contents are changing from moment to moment. The ISS, on the other hand, tries to provide very high-speed, high-throughput access to a relatively static set of data.

5.0 Applications

There are several target applications for the initial implementation of the ISS. These applications fall into two categories: image servers and multimedia / video file servers.

Image Server

The initial use of the ISS is to provide data to a terrain visualization application in the MAGIC testbed. This application, known as TerraVision [17], allows a user to navigate through and over a high resolution landscape represented by digital aerial images and elevation models. TerraVision is of interest to the U.S. Army because of its ability to let a commander "see" a battlefield environment. TerraVision is very different from a typical "flight simulator"-like program in that it uses high-resolution aerial imagery for the visualization instead of simulated terrain. TerraVision requires large amounts of data, transferred at both bursty and steady rates. The ISS is used to supply image data at hundreds of Mbits/s rates to TerraVision. No data compression is used with this application because the bandwidth requirements are such that real-time decompression is not possible without using special purpose hardware.

In the case of a large-image browsing application like TerraVision, the strategy for using the ISS is straightforward: the image is tiled (broken into smaller, equal-sized pieces), and the tiles are scattered across the disks and servers of the ISS. The order of tiles delivered to the application is determined by the application predicting a "path" through the image (landscape), and requesting the tiles needed to supply a view along the path. The actual

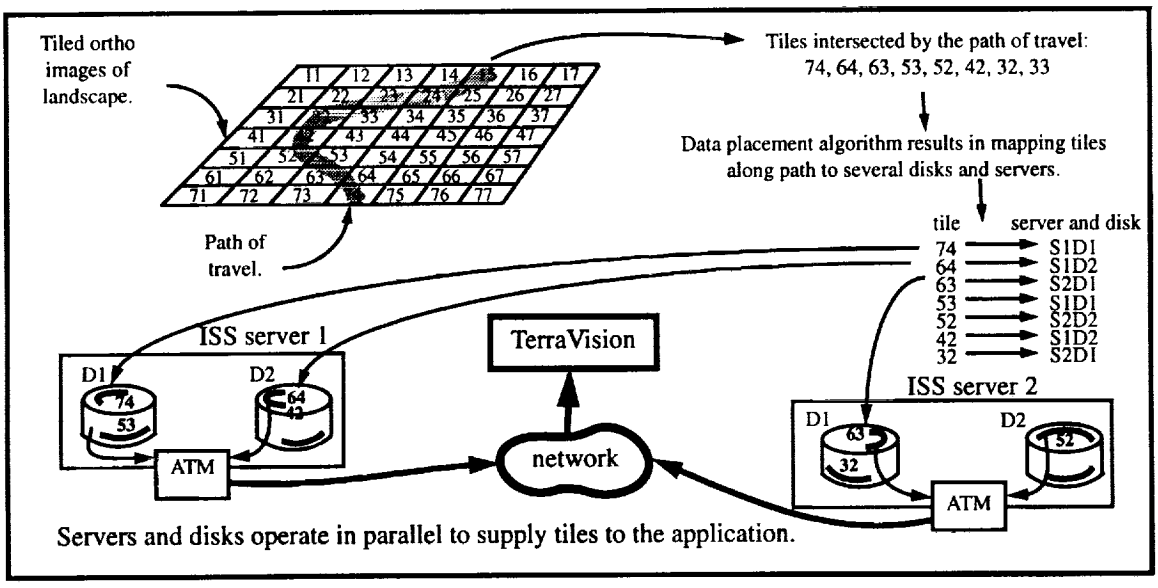


Figure 2 ISS Parallel Data Access Strategy as Illustrated by the TerraVision Application

delivery order is a function of how quickly a given server can read the tiles from disk and

send them over the network. Tiles will be delivered in roughly the requested order, but small variations from the requested order will occur. These variations must be accommodated by buffering, or other strategies, in the client application.

Figure 2 shows how image tiles needed by the TerraVision application are declustered across several disks and servers. More detail on this declustering is provided below.

Each ISS server is independently connected to the network, and each supplies an independent data stream into and through the network. These streams are formed into a single network flow by using ATM switches to combine the streams from multiple medium-speed links onto a single high-speed link. This high-speed link is ultimately connected to a high-speed interface on the visualization platform (client). On the client, data is gathered from buffers and processed into the form needed to produce the user view of the landscape.

This approach could supply data to any sort of large-image browsing application, including applications for displaying large aerial-photo landscapes, satellite images, X-ray images, scanning microscope images, and so forth.

Figure 3 shows how the network is used to aggregate several medium-speed streams into one high-speed stream for the image browsing application. For the MAGIC TerraVision

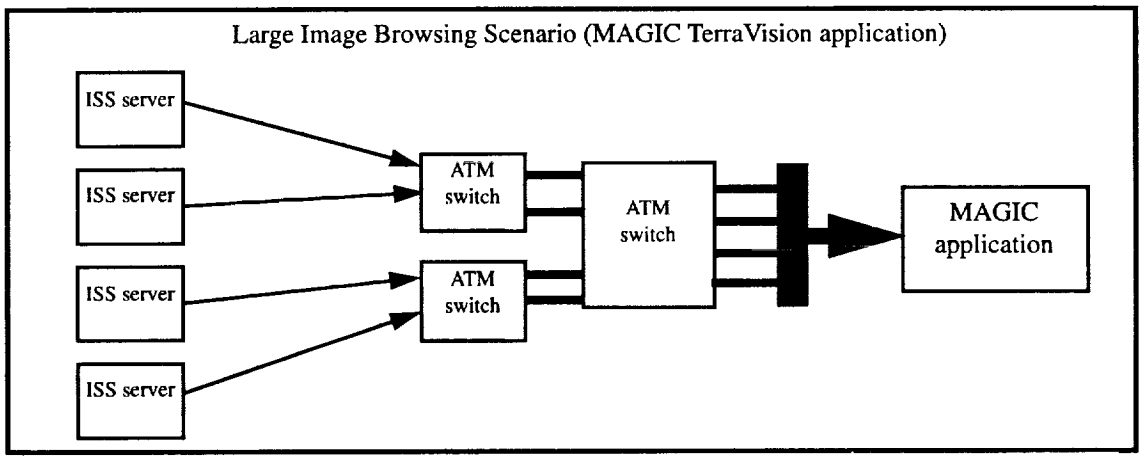


Figure 3 Use of the ISS for Single High-Bandwidth Application

application, the application host (an SGI Onyx) is using multiple OC-3 (155 Mbit/s) interfaces to achieve the bandwidth requirements necessary. These multiple interfaces will be replaced by a single OC-12 (622 Mbit/s) interface when it becomes available.

In the MAGIC testbed (see Figure 4), the ISS has been run in several ATM WAN configurations to drive several different applications, including TerraVision. The configurations include placing ISS servers in Sioux Falls, South Dakota (EROS Data Center), Kansas City, Kansas (Sprint), and Lawrence, Kansas (University of Kansas), and running the TerraVision client at Fort Leavenworth, Kansas (U. S. Army's Battle Command Battle Lab).

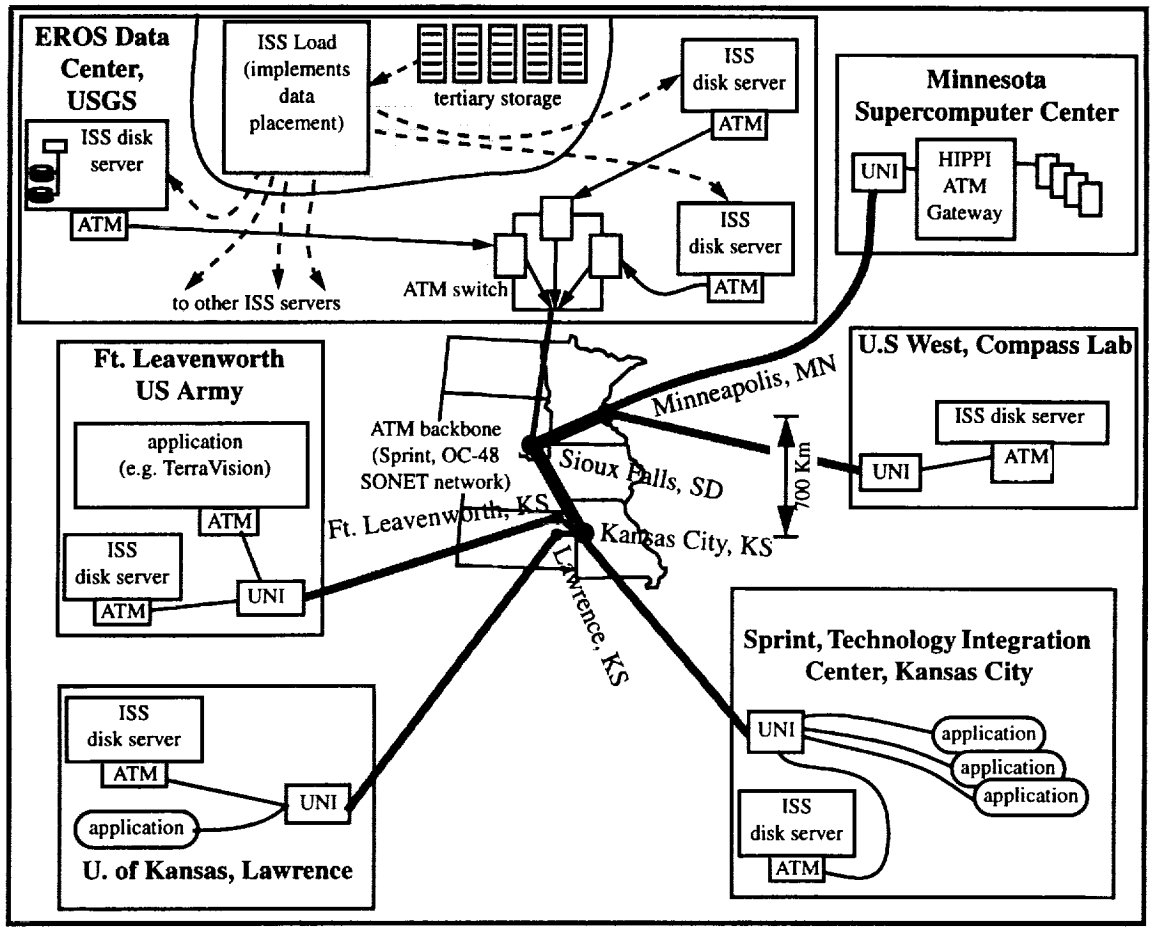


Figure 4 MAGIC Testbed Application and Storage System Architecture

The ISS disk server and the TerraVision application are separated by several hundred kilometers, the longest single link being about 700 kilometers.

Video Server

Examples of video server applications include video players, video editors, and multimedia document browsers. A video server might contain several types of stream-like data, including conventional video, compressed video, variable time base video, multimedia hypertext, interactive video, and others. Several users would typically be accessing the same video data at the same time, but would be viewing different streams, and different frames in the same stream. In this case the ISS and the network are effectively being used to "reorder" segments (see Figure 5). This reordering affects many factors in an image server system, including the layout of the data on disks. Commercial concerns such as Time Warner and U.S. West are building large-scale commercial video servers such as the Time Warner / Silicon Graphics video server [17]. Because of the relatively low cost and ease of scalability of our approach, it may address a wider scale, as well as a greater diversity, of data organization strategies so as to serve the needs of schools, research institu-

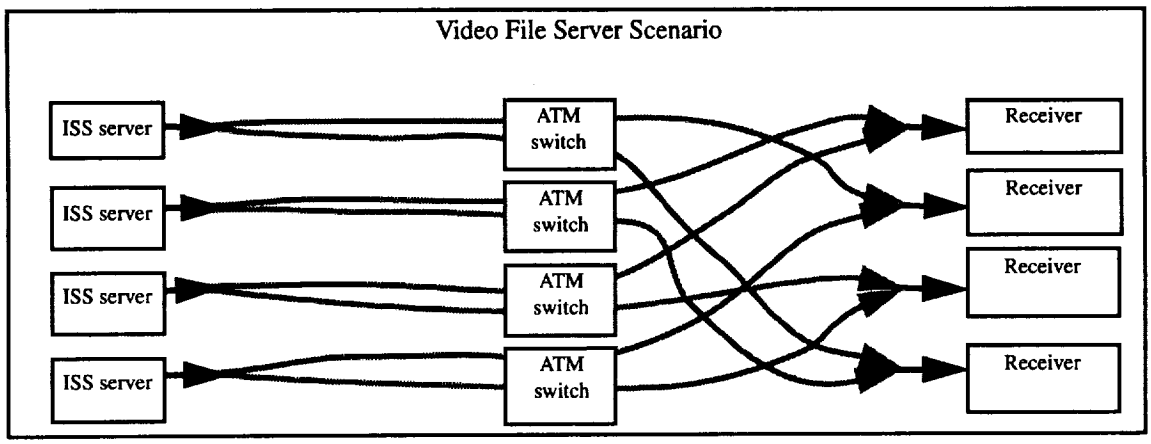


Figure 5 Use of the ISS to Supply Many Low-Bandwidth Streams

tions, and hospitals for video-image servers in support of various educational and research-oriented digital libraries.

Health Care Application[2]

An example of a medical application where we will be using this technology is the collection and playback of angiography images. Procedures used to restore coronary blood flow, though clinically effective, are expensive and have contributed significantly to the rising cost of medical care. To minimize the cost of such procedures, medical care providers are beginning to concentrate these services in a few high-volume tertiary care centers. Patients are usually referred to these centers by cardiologists at their home facilities; the centers then must communicate the results back to the local cardiologists as soon as possible after the procedure.

The advantages of providing specialized services at distant tertiary centers are significantly reduced if the medical information obtained during the procedure is not delivered rapidly and accurately to the treating physician in the patient's home facility. The delivery systems currently used to transfer patient information between facilities include interoffice mail, U.S. Mail, fax machine, telephone, and courier. Often these systems are inadequate and potentially could introduce delays in patient care.

With an ATM network and a high-speed image file server, still image and video sequences can be collected from the imaging systems. These images are sent through an ATM network to storage and analysis systems, as well as directly to the clinic sites. Thus, data can be collected and stored for later use, data can be delivered live from the imaging device to

2. This work is being done in conjunction with Dr. Joseph Terdiman, Kaiser Permanente Division of Research, and Dr. Robert Lundstrum, San Francisco Kaiser Hospital Cardiac Catheterization Laboratory. The implementation is being done with the support of a Pacific Bell CalREN grant (ATM network access), and in collaboration with Sun Microsystems and Phillips Palo Alto Research Laboratory. See http://www-itg.lbl.gov/Kaiser/home-page.html

remote clinics in real-time, or these data flows can all be done simultaneously. Whether the ISS servers are local or distributed around the network is entirely a function of the optimal logistics. There are arguments in regional healthcare information systems for centralized storage facilities, even though the architecture is that of a distributed system. See, for example, [18].

EOS-DIS

There are several possible uses of the DPSS technology within the EOSDIS architecture

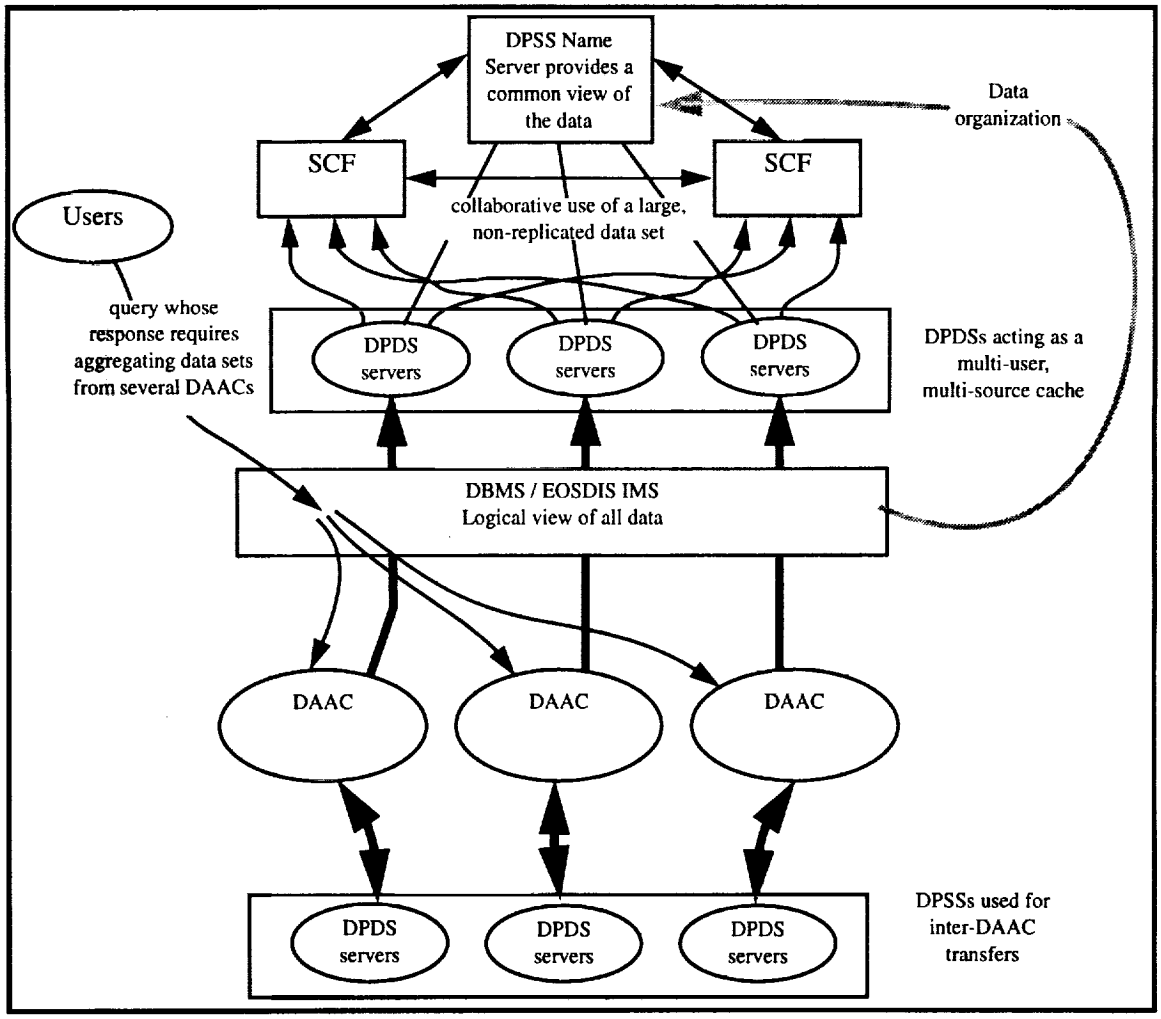


Figure 6 Possible Uses of DPSS Within the EOSDIS Architecture

as a new element providing a shared, high speed cache (see Figure 5).

One use provides for a "community" cache supporting a single instance of a large data set being used independently or collaboratively by several sites. This use is largely independent of the existing EOSDIS system, but would require an application to coordinate the data transfer from one or more DAACs to the DPSS. This application could also provide

the data structure definition and resource allocation by communication with the DPSS name server. One possible origin of large data sets that need to be available on-line as a unit are those that result from queries to multiple databases (e.g. data from multiple DAACs).

A second potential use is as a buffer for high speed data sources. As a data source turns on and off, it could write data to the DPSS. Once on the DPSS servers, the data can be read off at rates suitable to an application loading a database. During the read process the data can easily be reorganized since the DPSS provides very fast random access to data bocks. (The whole DPSS is optimized as a random-access block server.) Similarly, the DPSS could provide a high-speed, random-access cache for reorganizing and moving data among DAACs.

6.0 Glossary

EOSDIS: Earth Observing System, Data and Information System

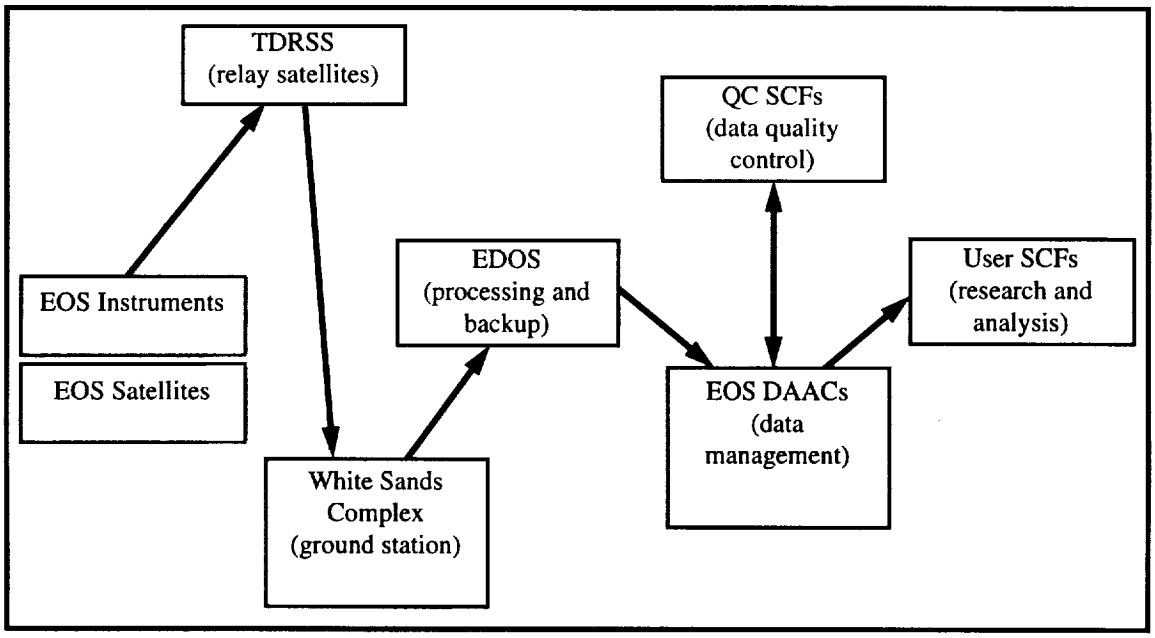


Figure 7 EOS DIS Architecture (from [2])

DAAC: Distributed Active Archive Center (of EOSDIS)

EDOS: EOS Data and Operations System

SCF: Science Computing Facility (of EOSDIS - both NASA and user facility)

TDRSS: Tracking and Data Relay System

7.0 References

[1] MAGIC (Multidimensional Applications and Gigabit Internetwork Consortium) is a gigabit network testbed that was established in June 1992 by the U. S. Government's Advanced Research Projects Agency (ARPA)[19]. The testbed is a collaboration between Mitre, LBL, Minnesota Supercomputer Center, SRI, Univ. of Kansas, Lawrence, KS, USGS - EROS Data Center, Sprint, Northern Telecom, U. S. West, Southwest Bell, and Splitrock Telecom. More information about MAGIC may be found on the WWW home page at: http://www.magic.net/ .

[2] See any of several NASA documents on EOSDIS. For example: *EOS Data and Information System (EOSDIS), NASA*, May, 1992, available from ESSO Document Resource Facility via NASA Headquarters, Earth Science and Applications Division (Code SE), Washington, D. C. 20546. Also see http://harp.gsfc.nasa.gov:1729/eosdis_documents/eosdis_home.html .

[3] Leclerc. Y. G. and S. Q. Lau, "TerraVision: A Terrain Visualization System," Technical Note 540, SRI International, Menlo Park, CA, Mar. 1994. Available from http://www.ai.sri.com/~magic/terravision.html .

[4] Patterson, D., Gibson, R., and Katz, R., "The Case for RAID: Redundant Arrays of Inexpensive Disks", Proceedings ACM SIGMOD Conference, Chicago, IL, May, 1988 (pp. 106-113) (See http://cs-tr.cs.berkeley.edu/TR/Search/ .)

[5] V. Jacobson, V., R. Braden, D. Borman, "TCP Extensions for High Performance," Internet Engineering Task Force, Request for Comments (RFC) 1323, May, 1992. (Available from http://ds.internic.net/ds/dspg1intdoc.html .)

[6] Chen L. T. and Rotem D., "Declustering Objects for Visualization", Proc. of the 19th VLDB (Very Large Database) Conference, 1993.

[7] Tierney, B., Johnston, W., Chen, L.T., Herzog, H., Hoo, G., Jin, G., Lee, J., and Rotem, D., "Distributed Parallel Data Storage Systems: A Scalable Approach to High Speed Image Servers", Proceedings of ACM Multimedia '94, Oct. 1994, LBL-35408. Also see http://george.lbl.gov/ISS/papers.html .

[8] The most current (and evolving) description of the DPSS / ISS technology is in the report LBL-36002 at http://www-itg.lbl.gov/ISS/papers.html .

[9] Open Software Foundation, *OSF DCE Applications Development Guide*, Prentice Hall, Englewood Cliffs, New Jersey, 1993. (Also see http://www.osf.org:8001/ .)

[10] Shirley, J., W. Hu, and D. Magid, *Guide to Writing DCE Applications*, 2ed., O'Reilly & Associates, Sebastopol, CA, 1994. (Also see http://www.ora.com/ .)

[11] Tierney, B., Johnston, W., Herzog, H., Hoo, G., Jin, G., and Lee, J., "System Issues in Implementing High Speed Distributed Parallel Storage Systems", Proceedings of the USENIX Symposium on High Speed Networking, Aug. 1994, LBL-35775. Also see http://george.lbl.gov/ISS/papers.html .)

[12] Tierney, B., Johnston, W., Chen, L.T., Herzog, H., Hoo, G., Jin, G., Lee, J., "Using High Speed Networks to Enable Distributed Parallel Image Server Systems", Proceedings of Supercomputing '94, Nov. 1994, LBL-35437. Available from http://george.lbl.gov/ISS/papers.html .)

[13] Ghandeharizadeh, S. and Ramos, L, "Continuous Retrieval of Multimedia Data Using Parallelism", IEEE Transactions on Knowledge and Data Engineering, Vol 5, No 4, August 1993.

[14] Rowe, L. and Smith, B.C., "A Continuous Media Player", Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video, San Diego, CA, Nov. 1992. (See http://cs-tr.cs.berkeley.edu/TR/Search/ .)

[15] Buddhikot, M. M., Parulkar, G., and Cox, J., "Design of a Large Scale Multimedia Storage Server", Proceedings of INET '94 / JENC5, 1994.

[16] Hartman, J. H. and Ousterhout, J. K., "Zebra: A Striped Network File System", Proceedings of the USENIX Workshop on File Systems, May 1992. (See http://cs-tr.cs.berkeley.edu/TR/Search/ .)

[17] Langberg, M., "Silicon Graphics Lands Cable Deal with Time Warner Inc.", San Jose Mercury News, June 8, 1993.

[18] Johnston, W., and Allen, A., M.D., "Regional Health Care Information Systems: Motivation, Architecture, and Implementation", LBL report no. 34770, Lawrence Berkeley Laboratory, Berkeley, CA, 94720.

[19] Richer, I. and Fuller, B.B., "An Overview of the MAGIC Project," M93B0000173, The MITRE Corp., Bedford, MA, 1 Dec. 1993. (Available from http://www.magic.net/MAGIC_Summary.ps .)

Robotic Tape Library System Level Testing at NSA, Present and Planned

Michael F. Shields
Department of Defense
9800 Savage Road
Fort Meade, Maryland 20755
Tel: 301-688-9509
Fax: 301-688-9454

In the present era of declining Defense budgets, increased pressure has been placed on our Agency to utilize Commercial Off The Shelf (COTS) solutions to incrementally solve a wide variety of our computer processing requirements. With the rapid growth in processing power, significant expansion of high performance networking, and the increased complexity of applications data sets, the requirement for high performance, large capacity, reliable and secure, and most of all affordable robotic tape storage libraries has greatly increased. Additionally, the migration to a heterogeneous, distributed computing environment has further complicated the problem. With today's open system compute servers approaching yesterday's supercomputer capabilities, the need for affordable, reliable secure Mass Storage Systems (MSS) has taken on an ever increasing importance to our processing centers' ability to satisfy operational mission requirements. To that end, NSA has established an in-house capability to acquire, test, and evaluate COTS products. Its goal is to qualify a set of COTS MSS libraries, thereby achieving a modicum of standardization for robotic tape libraries which can satisfy our low, medium, and high performance file and volume serving requirements. In addition, NSA has established relations with other Government Agencies to complement this in-house effort and to maximize our research, testing, and evaluation work. While the preponderance of the effort is focused at the high end of the storage ladder, considerable effort will be extended this year and next at the server class or mid range storage systems.

Over the past year, we have performed extensive testing of several high performance, high capacity Mass Storage Systems. In the open systems arena, we have evaluated the Convex based EMASS FileServ hierarchical storage management (HSM) product, see figure 1. Initially, the system was tested for use in one of our processing areas as the deep storage/archive for multiple server class UNIX based systems. These client systems were networked using FDDI to the HSM which managed the multiple clients' stored files. Classes were created, disk and tape capacity were dedicated to each client, and policies were established to tune the system for each client's storage and retrieval needs. A dedicated client system under the control of the test team was also included in the configuration under test, so as to baseline the load and to control feeds and flows as the test progressed. This element (a dedicated client system) is a recommended must for any system level test. To establish a consistent approach to testing this and other Mass Storage Systems, a standard test approach was developed. The first phase of this standard test was

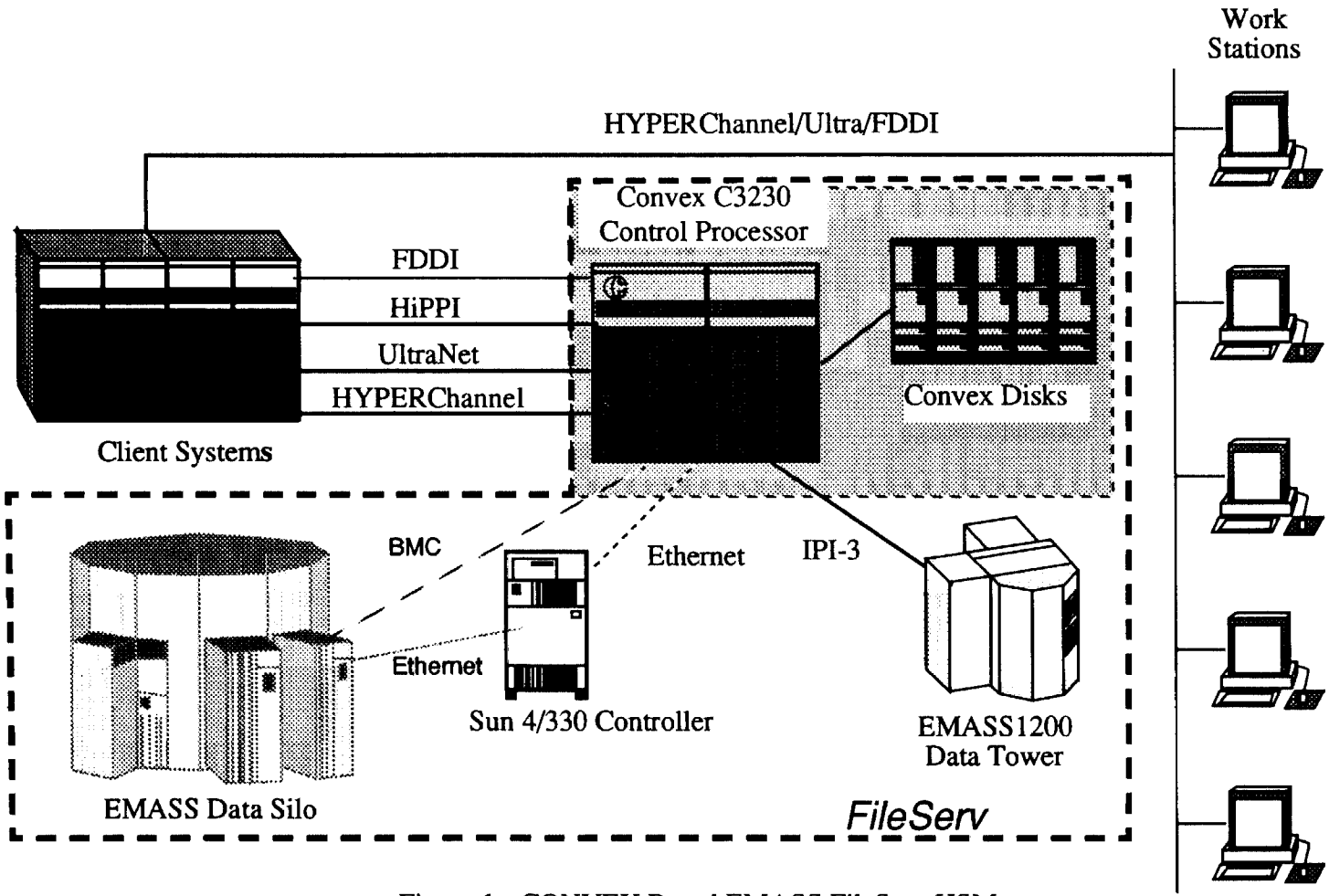


Figure 1 - CONVEX Based EMASS FileServ HSM

to qualify all of the vendor's commands and extensions and to verify that they operated as advertised. Once this was completed, we used the dedicated Test Client System to generate files of varying sizes and frequencies. This was essential to establish the baseline load. We then would vary the feeds and flows and measure the change as multiples of our baseline load, (e.g. 2X, ... 10X, etc.). Since almost every HSM's performance is highly dependent upon file size, we established three sizes of files (small: 1.5 MB, medium: 10-15 MB, and large 150 MB) in order to adequately categorize the system's end-end performance. Over the duration of our testing we carefully controlled the file size parameter by phase so as to measure the optimal disk and tape allocations for each client system. Our goal was not to break the system, but to establish the optimal range in which to have it operate most efficiently.

In the early phases of our testing, we spent significant time comparing the file sent to the HSM with the data stored. By performing check sums on each file stored for about a two week period, we discovered a flaw in the Convex D2 tape driver microcode, which was quickly fixed by the vendor. After two weeks of verifying that all of the data in each file was successfully written to tape and could be retrieved, this testing was suspended. At our Agency, data integrity is paramount and must always perform at 100%. Suffice it to say that, although we were about the 20th customer for this commercial product, no other customer had experienced the data integrity problem in their facility. We believe that this was due to their testing approach. Although we do NOT normally perform this degree of data integrity testing for most commercial products, it is strongly recommended that it be done for any new tape drive that is introduced. Since we were using the EMASS ER90 Helical Scan D2 drives, we felt it necessary to verify data integrity at a high confidence level; as our tests indicated, this was a wise step. We will also do this for IBM's NTP and STK's REDWOOD drives before they are placed into production.

The next phase of our testing was aimed at sustainability and reliability. Since our storage paradigm is to have all Mass Storage Systems located in unmanned spaces and to be remotely monitored by a geographically separated command center, production storage systems must be highly reliable and be capable of degraded mode operations. They must operate for long periods of time without operator/maintenance intervention to justify their existence. Our current standard for reliability is the STK silo which is our main line Mass Storage System for today's production, see figure 2. Over the past year, all of the drives/controllers have been upgraded to 36 track, and we are about 35% completed with the infusion of 800 MB tapes. Over the past five years, we have had only a small level of problems with these systems as they only require preventative maintenance at 6 month intervals. With self contained cleaning cartridges, they have proven to be highly reliable and satisfy our personnel staffing limitations.

Our Convex/EMASS HSM was initially tested with 4 ER90 D2 Helical Scan drives which were housed in an Odetics Data Tower. Its capacity was about 5.7 TBs. During the reliability/sustainability testing phase, we experienced significant difficulty with the

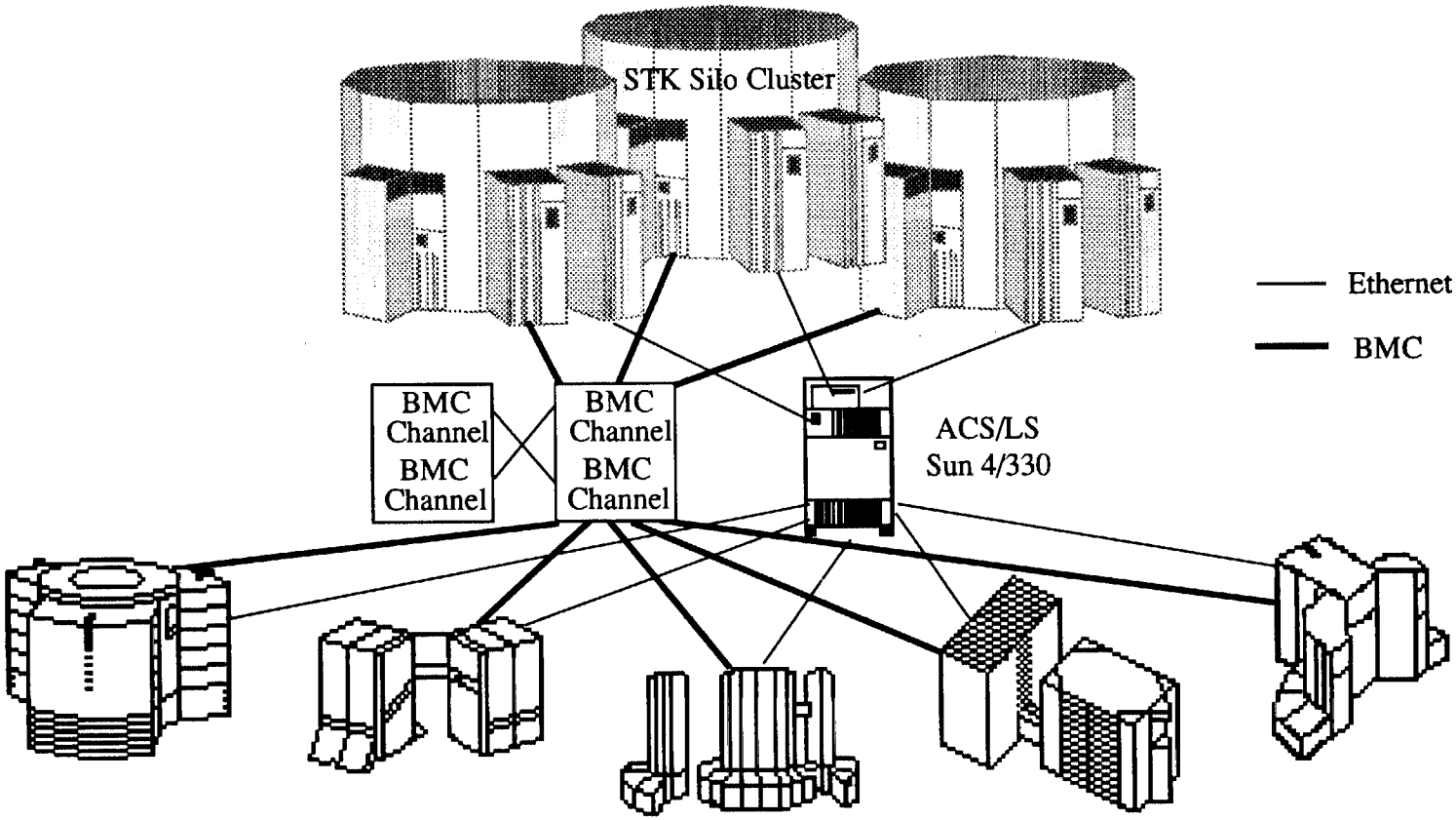


Figure 2 - Current Standard Mass Storage System for Production

robotics. Problems encountered included several instances of "stuck tapes", several dropped tapes, excessive mechanical wear on the cassettes themselves, and repeated failure of the robotic hub itself. Over a nine month period, five hub failures were experienced. The lack of reliability of the hub in large measure caused the Government to fail the system acceptance test. While the contractor went to yeoman efforts to attempt to correct these deficiencies, the problem persisted. A side effect of the robotics failures made endurance testing of the drives impossible; even still, we experienced a fair level of problems which made the drive questionable for "lights out" use. The principal problems encountered with the drives were head related. We determined that in order to have a margin of safety, we needed to have operators clean the heads shortly after 50 head/tape contact hours of use. In order to accurately monitor remotely when this event occurred, software had to be written which accessed firmware counters in the drives/controllers. In addition, operators had to monitor the error correction code counters in the drives. Again, software had to be written to enable this activity. We discovered that as soon as a drive had to employ the second level ECC, it was prudent to vary the drive off line to preclude a permanent write error. Employing this technique, we never encountered a hard write error. However, the degree of monitoring by our test team and operations personnel was deemed excessive. Another significant problem encountered was in the quality of the replacement heads. While some heads greatly exceeded their warranty hour limit., others failed prematurely. We concluded that this resulted from poor quality control in the manufacturing process. However, we noted that once a set of heads got past the 70-100 hour mark, they tended to be reliable for their design life, and often exceeded it. But once again, the degree of monitoring and maintenance intervention made this library unsuitable for our lights out processing scenario.

As a result of the aforementioned problems, the Government acquired an STK POWDERHORN 36 track/800 MB per tape Silo system with eight 4490 drives. This system was connected to our Convex/EMASS HSM as the second archive. It underwent its acceptance testing without a single problem. With the long tape, it provided a capacity of 4.4 TBs. The Government was highly interested in verifying that the EMASS FileServ software could effectively control two archives with different drive types. Since the client systems had a preponderance of small files, on the order of 6-10 MBs, the STK robotics and drives outperformed the Odetics/ER90 configuration. However, when the file sizes were changed to 100-150 MBs, the ER90s were more efficient. Testing of the mixed mode archive continued until the Government reached a level of confidence that the system could perform as advertised. At that time, the Odetics Tower and ER90 drives were dismantled and returned to the integration contractor. Noting the deficiencies encountered during the testing, the contractor offered to deliver a Grau ABBA/2 robotic library with IBM NTP drives as a replacement, see figure 3. This system will be integrated and tested with the Government's loading scenarios at the contractor's facility prior to shipping the system to NSA in January 1996.

The Convex/EMASS FileServ System with STK Silo and drives is now in production at NSA. While the steps cited above are somewhat skewed to our specific clients/networks,

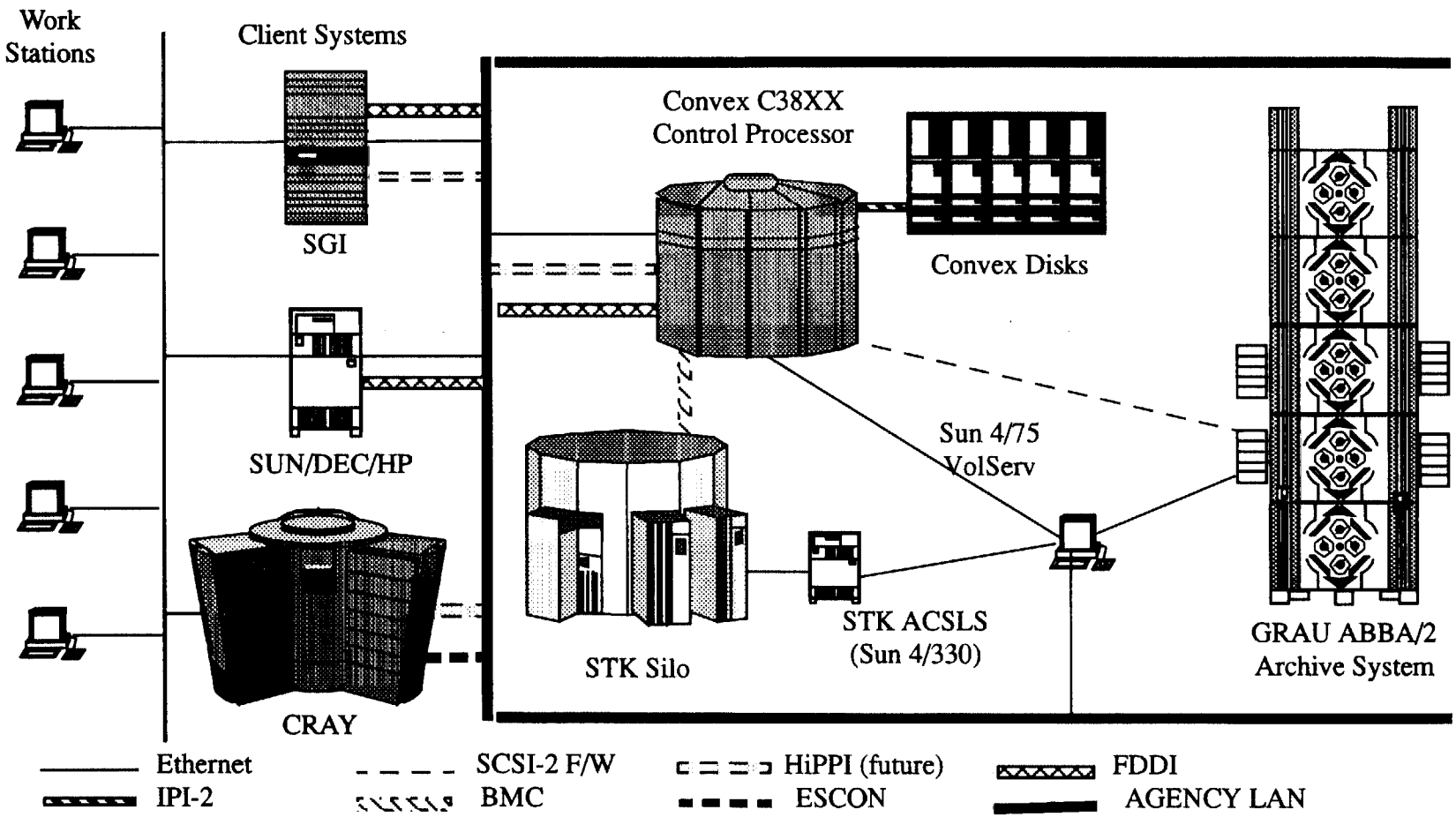


Figure 3 - CONVEX Based EMASS FileServ HSM with GRAU ABBA/2 and NTP Drives

we believe that our test approach is sound and generally applicable to any robotic tape HSM. In early 1994, we applied the same testing approach to a Sun/AMASS/Metrum RS48 robotic tape system. Once again, the theme was to verify the command set and functionality, verify the data integrity, evaluate the reliability and sustainability of the drives/robotics, and to categorize the sustained throughput of the system. We found this to be a stable product for low performance Mass Storage requirements.

NSA will evaluate the following server class systems during CY95. For the medium performance solution we have acquired an SGI Challenge series computer running the AMASS software. Three different robotic/drive configurations will be tested, see figure 4. They are IBM 3494/NTP for high performance/high capacity, Quantum DLT/Odetics 2640 for medium performance/capacity, and Exabyte 480/Mammoth for low/medium performance/capacity. Once again we will use the same approach as outlined above to evaluate/categorize these configurations.

For the high end high performance/high capacity robotic tape requirements of the Scientific Processing Complex, we have acquired two different volume servers. The first is an IBM 3495 L20 with 8 NTP drives which is being qualified by Cray Research Inc (CRI), see figure 5. Once this qualification is completed, the system will be fielded at NSA and will undergo in-house testing in late CY95. The second high performance/high capacity system to be tested is a Grau ABBA/2 robotics with NTP drives. It also will be qualified by a cooperative effort by E-Systems and CRI. Once the system is qualified, it will be shipped to NSA and will undergo in-house testing in early CY96. Both of these systems will have ESCON connectivity to the drives, which will facilitate sharing of the system by any of our Crays.

We feel that the Mass Storage community should establish performance benchmarks for products to aid the customer community in selecting the right Mass Storage products for their operational requirements. Our experience is that none of the vendors can provide the right size system configuration for any customer's needs. Today the entire burden for system sizing and delivered performance rests on the customer. Vendors need to perform and disseminate more evaluation information. They need to cover as broad a range as possible, either with their own testing or teamed with another, larger vendor who has the resources needed to perform the tests. Given a broad enough range of tests, customers should be able to take the results and extrapolate the expected performance characteristics for their environment. Solutions should be predictable and must include control processor network bandwidth, memory and disk needs, channels and I/O bandwidth, numbers of drives, and controllers, and robotics speeds. All that really matters for a Mass Storage System is the end-to-end sustainable bandwidth for stores and retrieves between the clients and the HSM. The industry must address some form of performance benchmark standards which will be the first step in aiding the customer in selecting the right system configuration for their unique problem. We have SpecMarks for processors, TP benchmarks for Data Base machines, but have nothing for storage systems. This area must be addressed soonest

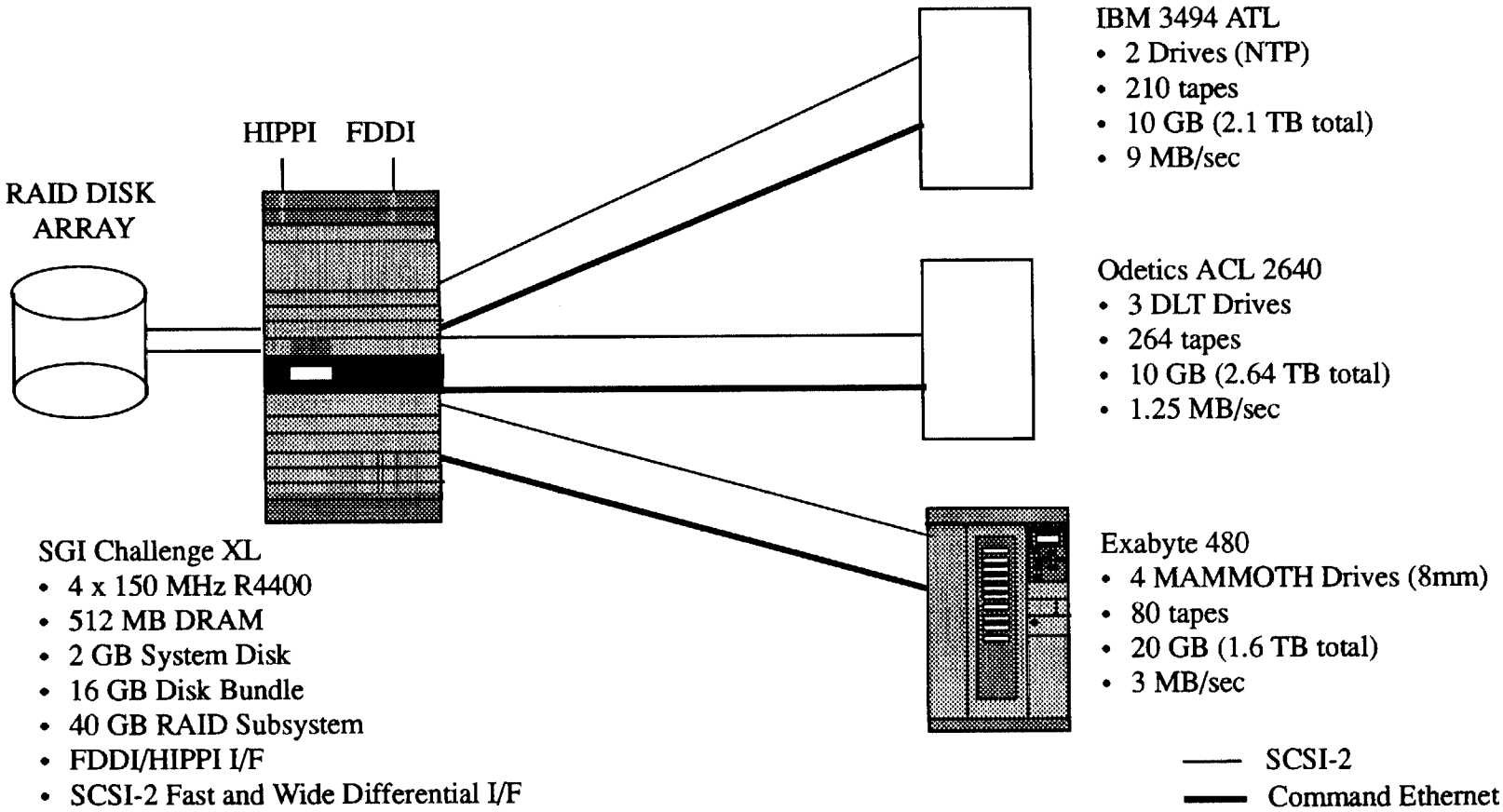


Figure 4 - Medium Performance Soution Evaluation

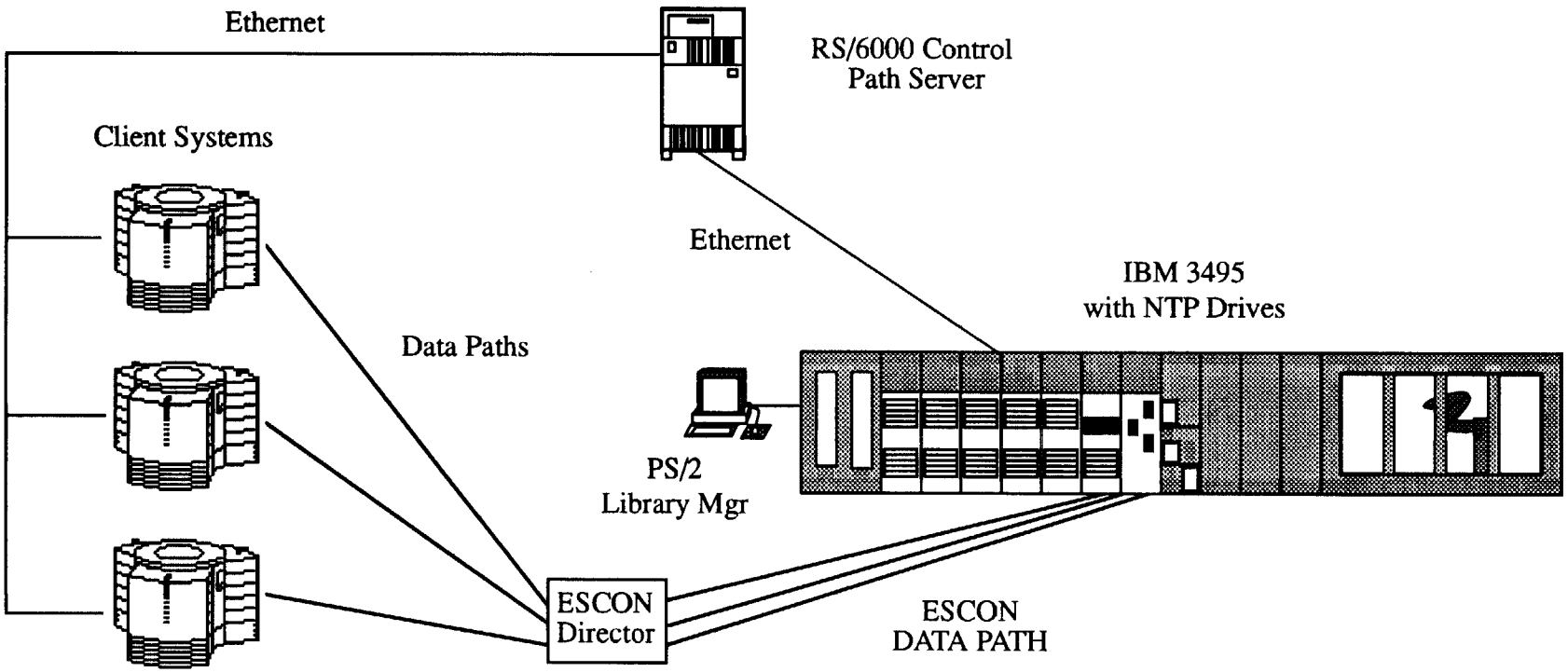


Figure 5 - High Performance / High Capacity Soution Evaluation

by the storage vendors. NSA has invested significant in-house resources in evaluating just a few of the systems available on the commercial marketplace. Scalability of the number of files that the HSM can manage is another key area of uncertainty. NSA has discovered the EMASS FileServ HSM has scalability limits largely caused by their use of Ingres RDBMS software in their commercial product. While this is a temporal limit, scalability testing of the product by vendors in-house, prior to first customer ship is a must for companies to survive. Competition dictates that this must be done and done quickly.

In summary, our approach should be clear as we have standardized on three different robotic tape systems for our high end processors across our various computer complexes. These are IBM 3494/5, Grau ABBA/2, and STK 4400 silos. The drives used with these robotics include: IBM/STK 36 track, NTP, and D3. For our server class systems we have a similar approach envisioned, as outlined above. The principal common entity for the server class problem is the AMASS HSM product. The specific drive/robotics and platform will be selected based on the required performance and capacity. Regarding high performance file servers, we will evaluate the scalability of the EMASS FileServ product during early 1996 and make our decision regarding its suitability for 150+ TB libraries. Our ultimate goal for the future would be to have one logical shared robotic tape library, accessible by any of our computer complexes.

The Architecture of the High Performance Storage System (HPSS)

Danny Teaff
IBM Federal
3700 Bay Area Blvd.
Houston, TX 77058
(713) 282-8137
Fax (713) 282-8074
teaff@vnet.ibm.com

Dick Watson
Lawrence Livermore National Laboratory
PO Box 808, L-560
Livermore, CA 94550
(510) 422-9216
Fax (510) 423-7997
dwatson@llnl.gov

Bob Coyne
IBM Federal
3700 Bay Area Blvd., 5th Floor
Houston, TX 77058
(713) 282-8039
Fax (713) 282-8074
coyne@vnet.ibm.com

Abstract

The rapid growth in the size of datasets has caused a serious imbalance in I/O and storage system performance and functionality relative to application requirements and the capabilities of other system components. The High Performance Storage System (HPSS) is a scalable, next-generation storage system that will meet the functionality and performance requirements of large-scale scientific and commercial computing environments.

Our goal is to improve the performance and capacity of storage systems by two orders of magnitude or more over what is available in the general or mass marketplace today. We are also providing corresponding improvements in architecture and functionality. This paper describes the architecture and functionality of HPSS.

Introduction

The rapid improvement in computational science, processing capability, main memory sizes, data collection devices, multimedia capabilities, and integration of enterprise data are producing very large datasets. These datasets range from tens to hundreds of gigabytes up to terabytes. In the near future, storage systems must manage total capacities, both distributed and at single sites, scalable into the petabyte range. We expect these large datasets and capacities to be common in high-performance and large-scale national information infrastructure scientific and commercial environments. The result of this rapid growth of data is a serious imbalance in I/O and storage system performance and

functionality relative to application requirements and the capabilities of other system components.

To deal with these issues, the performance and capacity of large-scale storage systems must be improved by two orders of magnitude or more over what is available in the general or mass marketplace today, with corresponding improvements in architecture and functionality. The goal of the HPSS collaboration is to provide such improvements. HPSS is the major development project within the National Storage Laboratory (NSL). The NSL was established to investigate, demonstrate, and commercialize new mass storage system architecture to meet the needs above [5,7,21]. The NSL and closely related projects involve more than 20 participating organization from industry, Department of Energy (DOE) and other federal laboratories, universities, and National Science Foundation (NSF) supercomputer centers. The current HPSS development team consists of IBM U.S. Federal, four DOE laboratories (Lawrence Livermore, Los Alamos, Oak Ridge, and Sandia), Cornell University, and NASA Langley and Lewis Research Centers. Ampex, IBM, Maximum Strategy Inc., Network Systems Corp., PsiTech, Sony Precision Graphics, Storage Technology, and Zitel have supplied hardware in support of HPSS development and demonstration. Cray Research, Intel, IBM, and Meiko are cooperating in the development of high-performance access for supercomputers and MPP clients.

The HPSS commercialization plan includes availability and support by IBM as a high-end Service offering through IBM U.S. Federal. HPSS source code can also be licensed and marketed by any US. company.

Architectural Overview

The HPSS architecture is based on the IEEE Mass Storage Reference Model: version 5 [6,9] and is network-centered, including a high speed network for data transfer and a separate network for control (*Figure 1*) [4,7,13,16]. The control network uses the Open Software Foundation's (OSF) Distributed Computing Environment DCE Remote Procedure Call technology [17]. In actual implementation, the control and data transfer networks may be physically separate or shared. An important feature of HPSS is its support for both parallel and sequential input/output (I/O) and standard interfaces for communication between processors (parallel or otherwise) and storage devices. In typical use, clients direct a request for data to an HPSS server. The HPSS server directs the network-attached storage devices or servers to transfer data directly, sequentially or in parallel to the client node(s) through the high speed data transfer network. TCP/IP sockets and IPI-3 over High Performance Parallel Interface (HIPPI) are being utilized today; Fibre Channel Standard (FCS) with IPI-3 or SCSI, or Asynchronous Transfer Mode (ATM) will also be supported in the future [3,20,22]. Through its parallel storage support by data striping HPSS will continue to scale upward as additional storage devices and controllers are added to a site installation.

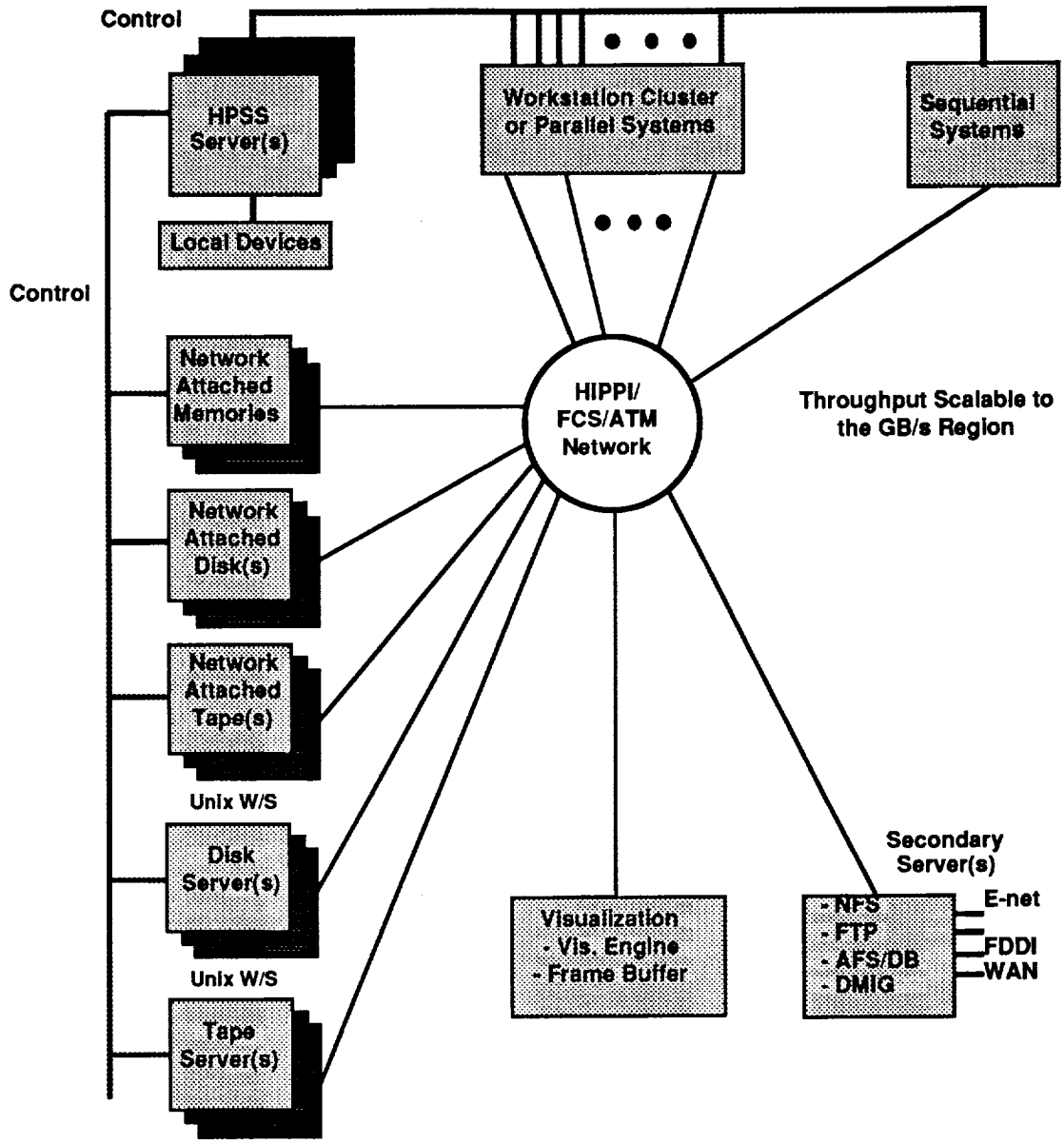


Figure 1 - Example of the type of configuration HPSS is designed to support

The key objectives of HPSS are now described.

Scalability

A major driver for HPSS is to develop a scalable, distributed, high performance storage management system. HPSS is designed to scale in several dimensions.

The HPSS I/O architecture is designed to provide I/O performance scaling by supporting parallel I/O through software striping [1]. The system will support application data transfers from megabytes to gigabytes per second with total system throughput of many

gigabytes per second. Data object number and size must scale to support billions of data objects, each potentially terabytes or larger in size, for total storage capacities in petabytes. This is accomplished through 64-bit metadata fields and scalable organization of system metadata. The system also is required to scale geographically to support distributed systems with hierarchies of hierarchical storage systems. Multiple storage systems located in different areas must integrate into a single logical system accessible by personal computers, workstations, and supercomputers. These requirements are accomplished using a client/server architecture, the use of OSF's DCE as its distributed infrastructure, support for distributed file system interfaces and multiple servers. HPSS also supports a scalable storage object name service capable of managing millions of directories and the ability to support hundreds to thousands of simultaneous clients. The latter is achieved through the ability to multitask, multiprocess and replicate the HPSS servers.

Modularity and APIs

The HPSS architecture is highly modular. Each replicable software component is responsible for a set of storage objects, and acts as a service provider for those objects. The IEEE Reference Model, on which the HPSS design is based, provides the modular layered functionality (see *Figure 2*) [6,9]. The HPSS software components are loosely coupled, with open application program interfaces (APIs) defined at each component level. Most users will access HPSS at its high level interfaces–currently client API, FTP (both parallel and sequential), NFS, Parallel File System (PFS), with AFS/DFS, Unix Virtual File System (VFS), and Data Management Interface Group (DMIG) interfaces in the future) [11,15,18,19]. However, APIs are available to the underlying software components for applications, such as large scale data management, digital library or video-on-demand requiring high performance or special services. This layered architecture affords the following advantages:

- **Replacement of selected software components**–As new and better commercial software and hardware components became available, an installation can add or replace existing components. For example, an installation might add or replace Physical Volume Repositories, Movers or the HPSS Physical Volume Library with other commercially available products.

- **Support of applications direct access to lower level services**–The layered architecture is designed to accommodate efficient integration of different applications such as digital library, object store, multimedia, and data management systems. Its modularity will enable HPSS to be embedded transparently into the large distributed information management systems that will form the information services in the emerging national information infrastructure. Support for different name spaces or data organizations is enabled through introduction of new Name Servers and data management applications.

Portability and Standards

Another important design goal is portability to many vendor's platforms to enable OEM and multivendor support of HPSS. HPSS has been designed to run under Unix requiring no kernel modifications, and to use standards based protocols, interfaces, and services where applicable. HPSS is written in ANSI C, and uses POSIX functions to enhance software portability. Use of existing commercial products for many of the infrastructure services

supported on multiple-vendor platforms enables portability, while also providing market proven dependability. Open Software Foundation (OSF) Distributed Computing Environment (DCE), Transarc's Encina transaction manager [8], Kinesix SAMMI and X-windows are being used by HPSS because of their support across multiple vendor platforms, in addition to the rich set of functionality provided. The HPSS component APIs have been turned over to the IEEE Storage System Standards Working Group as a basis for its standards activities.

Reliability and Recovery

Reliable and recoverable storage of data is mandatory for any storage system. HPSS supports several mechanisms to facilitate this goal. The client-server interactions between HPSS software components have been designed to be based on atomic transactions in order to maintain system state consistency [14]. Within the scope of a given request, a transaction may be established so that an abort (or commit) in one component will cause the other participating components to abort (or commit). The HPSS Metadata Manager is fully integrated with its Transaction Manager. Following an abort, the non-volatile file and name space metadata changes within the scope of the transactions will automatically be rolled back. For recovery purposes, mirroring of the storage object and name space metadata is supported. The HPSS architecture will also support data mirroring if desired in a future release.

Support is also provided to recover from failed devices and bad media. An administrator interface is provided to place a device off line. Once the device has been repaired, it may then be placed back on line. For bad media, an application interface is provided to move storage segments from a virtual volume to a new virtual volume.

The HPSS software components execute in a distributed manner. Should a processor fail, any of the HPSS software components may be moved to another platform. Component services are registered with the DCE Cell Directory Service (CDS) so that components may locate the services. Each component has also been designed to perform reconnect logic when a connection to a peer component fails. Connection context is maintained by selected components. When a connection context is established, a keep-alive activity is started to detect broken connections. A server may use the context information associated with a broken connection to perform any necessary clean up.

Security and Privacy

HPSS uses DCE and POSIX security and privacy mechanisms for authentication, access control lists, permissions and security labels. Security policy is handled by a separate policy module. Audit trails are also supported. Further, HPSS design and implementation use a rigorous software engineering methodology which support its reliability and maintainability.

Storage System Management

HPSS has a rich set of storage system management services for operators and system administrators based on managed object definitions. The application programming interface supports monitoring, reporting and controlling operations (see Appendix A).

Software Components

The HPSS software components are shown *Figure 2*. The shaded boxes are defined in the IEEE Mass Storage Reference Model: version 5 [9].

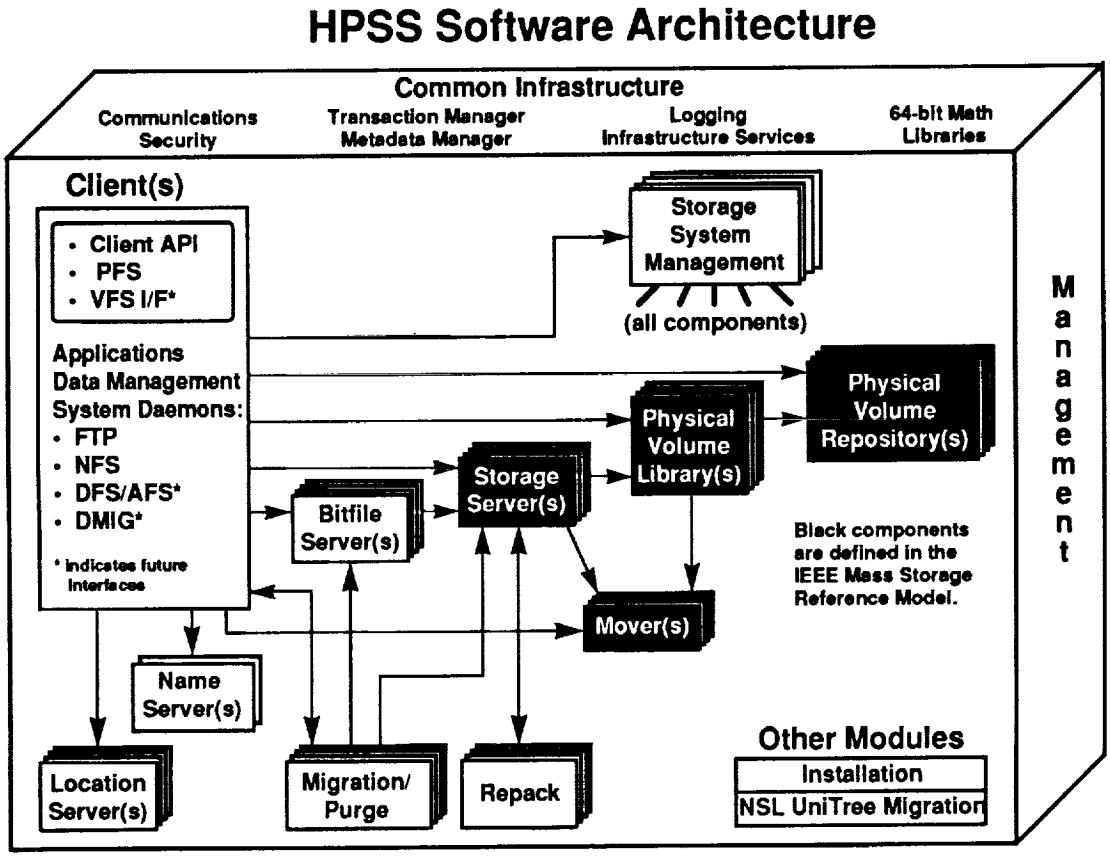


Figure 2 - Software Model Diagram

This section outlines the function of each component.

Infrastructure

HPSS design is based upon a well-formed industry standard infrastructure. The key infrastructure components are now outlined.

Distributed Computing Environment

HPSS uses OSF's DCE as the base infrastructure for its distributed architecture [17]. This standards-based framework will enable the creation of distributed storage systems for a national information infrastructure capable of handling gigabyte-terabyte-class files at gigabyte per second data transfer rates.

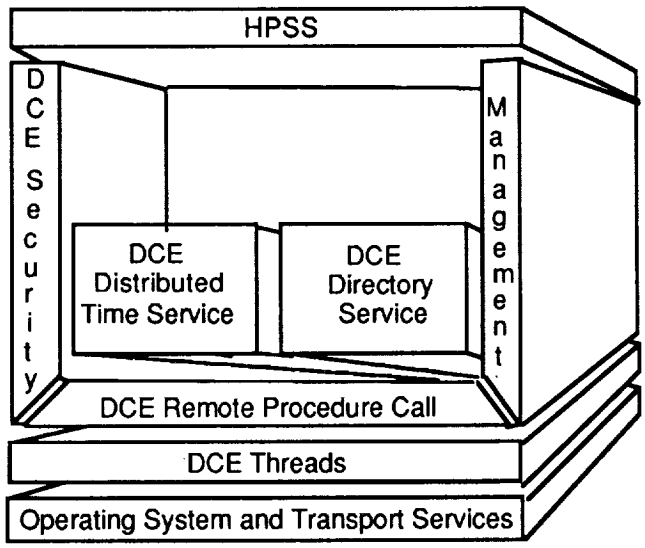


Figure 3 - HPSS DCE Architecture Infrastructure

DCE was selected because of its wide adoption among vendors and its near industry-standard status. HPSS uses the DCE Remote Procedure Call (RPC) mechanism for control messages and DCE Threads for multitasking. The DCE threads package is vital for HPSS to serve large numbers of concurrent users and to enable multiprocessing of its servers. HPSS also uses the DCE Security, Cell Directory, and Time services. A library of DCE convenience functions was developed for use in HPSS.

Transaction Management

Requests to HPSS to perform actions such as creating bitfiles or accessing file data results in client/server interactions between software components. Transaction integrity is required to guarantee consistency of server state and metadata in case a particular component should fail. As a result, a transaction manager was required by HPSS. Encina, from Transarc, was selected by the HPSS project as its transaction manager [8]. This selection was based on functionality, its use of DCE, and multi-platform vendor support.

Encina provides begin-commit-abort semantics, distributed two-phase commit, and nested transactions. In addition, Transaction RPCs (TRPCs), which extend DCE RPCs with transaction semantics, are provided. For recovery purposes, Encina uses a write-ahead log for storing transaction outcomes and updates to recoverable metadata. Mirroring of data is also provided.

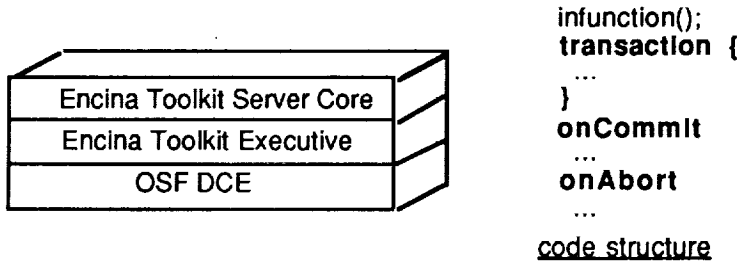


Figure 4 - Encina Components

Metadata Management

Each HPSS software component has system metadata associated with the objects it manages. Each server with non-volatile metadata requires the ability to reliably store its metadata. It is also required that metadata management performance be scalable as the number of object instances grow. In addition, access to metadata by primary and secondary keys is required. The Structured File Server (SFS), an Encina optional product, was selected by the HPSS project as its metadata manager. SFS provides B-tree clustered file records, record and field level access, primary and secondary keys, and automatic byte ordering between machines. SFS is also fully integrated with the Encina transaction manager. As a result, SFS provides transaction consistency and data recovery from transaction aborts. For reliability purposes, HPSS metadata stored in SFS is mirrored. A library of metadata manager convenience functions for retrieving, adding, updating, and deleting metadata for each of the HPSS components was developed.

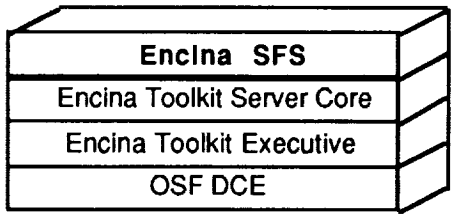


Figure 5 - Structured File Server (SFS)

Security

The security components of HPSS provide authentication, authorization, enforcement, and audit capabilities for the HPSS components. Authentication is responsible for guaranteeing that a principal is the entity that is claimed, and that information received from an entity is from that entity. Authorization is responsible for enabling an authenticated entity access to an allowed set of resources and objects. Authorization enables end user access to HPSS directories and bitfiles. Enforcement is responsible for guaranteeing that operations are

restricted to the authorized set of operations. Enforcement applies to end user access to bitfiles. Audit is responsible for generating a log of security relevant activity. HPSS security libraries utilize DCE and DCE security. The authentication service, which is part of DCE, is based on Kerberos v5. The following figure depicts how HPSS security fits with DCE and Kerberos.

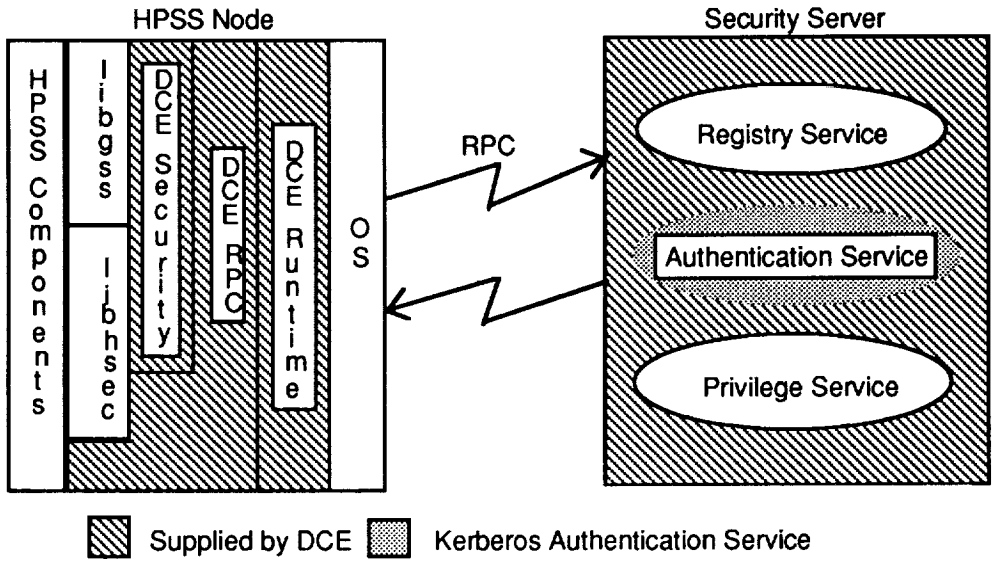


Figure 6 - HPSS Security

Communication

The control path communications between HPSS components is through DCE RPCs or Encina transaction RPCs. For data path communication, the HPSS Mover(s) currently utilize either Sockets or IPI-3 (over HIPPI) libraries. Future support is planned for IPI-3 and SCSI over Fibre Channel Standard and TCP/IP over ATM. A special parallel data transfer library has been developed. This library allows data to be transferred across many parallel data connections. The library transfers data headers that identify the data that follows. This allows data to be sent and arrive in any order on the parallel paths.

Logging

The HPSS logger is used to record alarms, events, requests, security audit records, accounting records, and trace information from the HPSS components. A central log is maintained which contains records from all HPSS components. A local log of activity from components on each HPSS node is also supported. When the central log fills, it will switch to a secondary log file. A configuration option allows the filled log to be automatically archived to HPSS. A delog function is provided to extract and format log records. Delog options support filtering by time interval, record type, server, and user.

64 Bit Arithmetic Libraries

HPSS supports file sizes up to 2**64 bytes. Many vendor platforms support only 32 bit integer arithmetic. In order to support large file sizes and large numbers of objects on 32 bit platforms, a library of 64 bit arithmetic functions has been developed. The functions support both big endian and little endian I/O architectures.

Interfaces

HPSS supports several high-level interfaces: currently Client API, FTP (both standard and parallel), and NFS, with DFS/AFS, DMIG, and VFS planned for future releases.

Client API

The HPSS Client file server API mirrors the POSIX file system interface specification where possible. The Client API also supports extensions to allow the programmer to take advantage of the specific features provided by HPSS (e.g., class-of-service, storage/access hints passed at file creation and support for parallel data transfers).

FTP (standard and parallel)

HPSS provides a standard FTP server interface to transfer files from HPSS to a local file system. Parallel FTP, an extension and superset of standard FTP, has been implemented to provide high performance data transfers to client systems. The standard FTP protocol supports third-party data transfer through separation of the data transfer and control paths, but it does not offer parallel data paths [11]. HPSS modified and augmented the standard client FTP file retrieval and storage functions to offer parallel data paths for HPSS data transfers. This approach provides high performance FTP transfers to the client while still supporting the FTP command set. Additional commands have been added to support parallel transfer. This work will be submitted to the Internet Engineering Task Force for standardization.

NFS

The NFS V2 Server interface for HPSS provides transparent access to HPSS name space objects and bitfile data for client systems from both the native HPSS and the Network File System V2 service. The NFS V2 Server translates standard NFS calls into HPSS control calls and provides data transfers for NFS read and write requests. The NFS V2 Server handles optimization of data movement requests by the caching of data and control information. If the server machine crashes, the NFS V2 Server is in charge of recovery of all cached data at the time of the crash. The NFS V2 Server will also recover when HPSS crashes. Before NFS clients can request NFS services, they must mount an exported HPSS directory by calling the Mount daemon mount API. Support for NFS V3 is planned for a future release.

Parallel File System

HPSS provides the capability to act as an external hierarchical file system to vendor Parallel File Systems (PFS). The first implementation supports the IBM SPx PIOFS. Early deployment is also planned for Intel Paragon and Meiko PFS integration with HPSS.

Name Server (NS)

The Name Server maps a file name to an HPSS object. The Name Server provides a POSIX view of the name space which is a hierarchical structure consisting of directories, files, and links. File names are human readable ASCII strings. Namable objects are any object identified by HPSS Storage Object IDs. The commonly named objects are bitfiles, directories, or links. In addition to mapping names to unique object identifiers, the Name Server provides access verification to objects. POSIX Access Control Lists (ACLs) are supported for the name space objects. A key requirement of the Name Server is to be able to scale to millions of directories and greater than a billion name space entries.

Bitfile Server (BFS)

The Bitfile Server provides the POSIX file abstraction to its clients. A logical bitfile is an uninterpreted bit string. HPSS supports bitfile sizes up to 2**64 bytes. A bitfile is identified by a Bitfile Server generated name called a bitfile-id. Mapping of a human readable name to the bitfile id is provided by a Name Server external to the Bitfile Server. Clients may reference portions of a bitfile by specifying the bitfile-id and a starting address and length. The writes and reads to a bitfile are random and the writes may leave holes where no data has been written. The Bitfile Server supports both sequential and parallel read and write of data to bitfiles. In conjunction with Storage Servers, the Bitfile Server maps logical portions of bitfiles onto physical storage devices.

Storage Server (SS)

The Storage Server provides a hierarchy of storage objects: logical storage segments, virtual volumes and physical volumes. All three layers of the Storage Server can be accessed by appropriately privileged clients. The server translates references to storage segments into references to virtual volume and finally into physical volume references. It also schedules the mounting and dismounting of removable media through the Physical Volume Library. The Storage Server in conjunction with the Mover have the main responsibility for orchestration of HPSS's parallel I/O operations.

The storage segment service is the conventional method for obtaining and accessing HPSS storage resources. The Storage Server maps an abstract storage space, the storage segment, onto a virtual volume, resolving segment addresses as required. The client is presented with a storage segment address space, with addresses from 0 to N-1, where N is the byte length of the segment. Segments can be opened, created, read, written, closed and deleted. Characteristics and information about segments can be retrieved and changed.

The virtual volume service is the method provided by the Storage Server to group physical storage volumes. The virtual volume service supports striped volumes today and mirrored volume in a future release. Thus, a virtual volume can span multiple physical volumes. The Storage Server maps the virtual volume address space onto the component physical volumes in a fashion appropriate to the grouping. The client is presented with a virtual

volume that can be addressed from 0 to N-1, where N is the byte length of the virtual volume. Virtual volumes can be mounted, created, read, written, unmounted and deleted. Characteristics of the volume can be retrieved and in some cases, changed.

The physical volume service is the method provided by the storage server to access the physical storage volumes in HPSS. Physical volumes can be mounted, created, read, written, unmounted and deleted. Characteristics of the volume can be retrieved and in some cases, changed.

Repack runs as a separate process. It provides defragmentation of physical volumes. Repack utilizes a Storage Server provided function which moves storage segments to a different virtual volume.

Mover (Mvr)

The Mover is responsible for transferring data from a source device(s) to a sink device(s). A device can be a standard I/O device with geometry (e.g., a tape or disk), or a device without geometry (e.g., network, memory). The Mover also performs a set of device control operations. Movers perform the control and transfer of both sequential and parallel data transfers.

The Mover consists of several major parts: Mover parent task, Mover listen task/request processing task, Data Movement, Device control, and System Management.

The Mover parent task performs Mover initialization functions, and spawns processes to handle the Mover's DCE communication, data transfer connections, as well as the Mover's functional interface. The Mover listen task listens on a well-known TCP port for incoming connections to the Mover, spawns request processing tasks, and monitors completion of those tasks. The request processing task performs initialization and return functions common to all Mover requests. Data movement supports client requests to transfer data to or from HPSS. Device control supports querying the current device read/write position, changing the current device read/write position, loading a physical volume into a drive, unloading a physical volume from a drive, flushing data to the media, writing a tape mark, loading a message to a device's display area, reading a media label, writing a media label, and zeroing a portion of disk. System management supports querying and altering device characteristics and overall Mover state.

Physical Volume Library (PVL)

The PVL manages all HPSS physical volumes. Clients can ask the PVL to mount and dismount sets of physical volumes. Clients can also query the status and characteristics of physical volumes. The PVL maintains a mapping of physical volume to cartridge and a mapping of cartridge to PVR. The PVL also controls all allocation of drives. When the PVL accepts client requests for volume mounts, the PVL allocates resources to satisfy the request. When all resources are available, the PVL issues commands to the PVR(s) to mount cartridges in drives. The client is notified when the mount has completed.

The Physical Volume Library consists of two major parts: Volume mount service and Storage system management service.

The volume mount service is provided to clients such as a Storage Server. Multiple physical volumes belonging to a virtual volume may be specified as part of a single request. All of the volumes will be mounted before the request is satisfied. All volume mount requests from all clients are handled by the PVL. This allows the PVL to prevent multiple clients from deadlocking when trying to mount intersecting sets of volumes. The standard mount interface is asynchronous. A notification is provided to the client when the entire set of volumes has been mounted. A synchronous mount interface is also provided. The synchronous interface can only be used to mount a single volume, not sets of volumes. The synchronous interface might be used by a non-HPSS process to mount cartridges which are in a tape library, but not part of the HPSS system.

The storage system management service is provided to allow a management client control over HPSS tape repositories. Interfaces are provided to import, export, and move volumes. When volumes are imported into HPSS, the PVL is responsible for writing a label to the volume. This label can be used to confirm the identity of the volume every time it is mounted. Management interfaces are also provided to query and set the status of all hardware managed by the PVL (volumes, drives, and repositories).

Physical Volume Repository (PVR)

The PVR manages all HPSS supported robotics devices and their media such as cartridges. Clients can ask the PVR to mount and dismount cartridges. Every cartridge in HPSS must be managed by exactly one PVR. Clients can also query the status and characteristics of cartridges.

The Physical Volume Repository consists of these major parts: Generic PVR service, and support for devices such as Ampex, STK, and 3494/3495 robot services, as well as an operator mounted device service.

The generic PVR service provides a common set of APIs to the client regardless of the type of robotic device being managed. Functions to mount, dismount, inject and eject cartridges are provided. Additional functions to query and set cartridge metadata are provided. The mount function is asynchronous. The PVR calls a well-known API in the client when the mount has completed. For certain devices, like operator mounted repositories, the PVR will not know when the mount has completed. In this case it is up to the client to determine when the mount has completed. The client may poll the devices or use some other method. When the client determines a mount has completed, the client should notify the PVR using one of the PVR's APIs. All other PVR functions are synchronous. The generic PVR maintains metadata for each cartridge managed by the PVR. The generic PVR interface calls robotics vendor supplied code to manage specific robotic devices.

The operator mounted device service manages a set of cartridges that are not under the control of a robotics device. These cartridges are mounted to a set of drives by operators. The Storage System Manager is used to inform the operators when mount operations are required.

Storage System Management (SSM)

The HPSS SSM architecture is based on the ISO managed object architecture [10,12]. The Storage System Manager (SSM) monitors and controls the available resources of the HPSS storage system in ways that conform to the particular management policies of a given site. Monitoring capabilities include the ability to query the values of important management

attributes of storage system resources as well as an ability to receive notifications of alarms and other significant system events. Controlling capabilities include the ability to set the values of management attributes of storage system resources and storage system policy parameters. Additionally, SSM can request that specific operations be performed on resources within the storage system, such as adding and deleting logical or physical resources. The operations performed by SSM are usually accomplished through standard HPSS server APIs.

SSM management roles cover a wide spectrum, including configuration aspects of installation, creating new volumes, initialization, operations, and termination tasks. SSM can provide management capabilities to a range of clients, including site administrators, systems administrators, operations personnel, complex graphical user interface (GUI) management environments, and independent management applications responsible for tasks such as purges, migration, and reclamation. Some of the functional areas of SSM include fault management, configuration management, security management, accounting management, and performance management.

SSM consists of these major parts: SSM Graphical User Interface (SAMMI GUI Displays), SAMMI Data Server, and System Manager.

The SSM Graphical User Interface allows operators, administrators, and users to interactively monitor and control the HPSS storage system. Kinesix's SAMMI product is used to provide the HPSS GUI services. SAMMI is built on X-windows and OSF's Motif. It provides mechanisms to simplify screen design and data management services for screen fields. Standard Motif widgets such as menus, scrollbar lists, and buttons are used. In addition SAMMI specific widgets such as dials, gauges, and bar charts are used for informational and statistical data.

The SAMMI Data Server is a client to the System Manager and a server to the SAMMI Runtime Display Manager. The SAMMI Data Server is the means by which data is acquired and fed to the SAMMI Displays.

The Storage System Manager is a client to the HPSS servers and a server to the SAMMI Data Server and other external clients wishing to perform management specific operations. It interfaces to the managed objects defined by the HPSS servers.

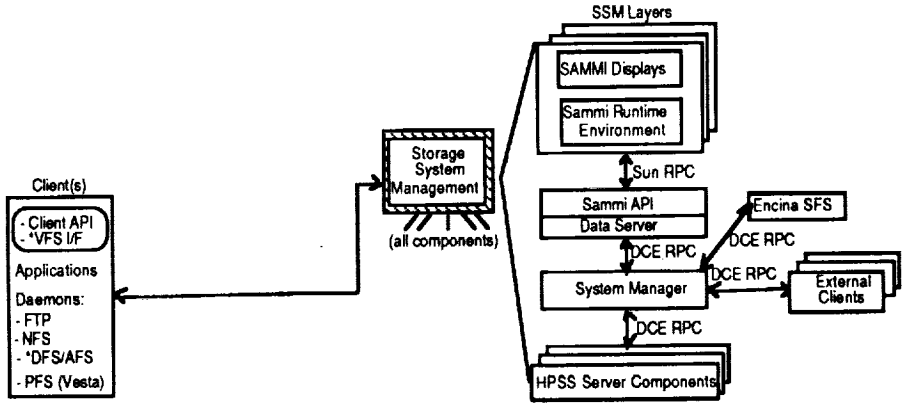


Figure 7 - Storage System Management

Migration - Purge

The Migration-Purge server provides hierarchical storage management for HPSS through migration and caching of data between devices. There are two types of migration and caching: disk migration and caching and tape migration and caching. Multiple storage hierarchies are supported by HPSS [2]. Data is cached to the highest level (fastest) device in a given hierarchy when accessed and migrated when inactive and space is required.

The main purpose of disk migration is to free up the disk storage. This type of migration contains two functions; migration and purge. Migration selects the qualified bitfiles and copies these bitfiles to the next storage level defined in the hierarchy. Purge later frees the original bitfiles from the disk storage.

The main purpose of tape migration is to free up tape volumes, and not just migrate bitfiles. The active bitfiles in the target virtual volumes are moved laterally to the free tape volumes in the same storage level. The inactive bitfiles in the target virtual volumes are migrated to the free tape volumes in the next storage level.

The HPSS component client APIs provide the vehicle for the Storage System Manger to request the server to start migration and purge whenever it is necessary. The migration-purge server is set up to run migration periodically with the time interval specified in the migration policy. In addition, the server will start the migration and purge to run automatically if the free space of a storage class is below the percentage specified in the migration-purge policy.

Other

Installation

Installation software is provided for system administrators to install/update HPSS, and perform the initial configuration of HPSS following installation. The full HPSS system is first installed to an installation node. Selected HPSS software components may then be installed (using the remote installation feature) from the installation node to the other nodes where HPSS components will be executed.

NSL-UniTree Migration

HPSS, through its support of parallel storage, provides significant improvements in I/O rates and storage capacity over existing storage systems software. In transitioning from existing systems, a migration path is required. The migration path should be transparent to end users of the storage system. The capability to migrate from NSL UniTree to HPSS is provided. The migration software handles both file metadata and actual data. Utilities convert the file metadata (e.g., storage maps, virtual volume data, physical volume data), and name space metadata from UniTree format to HPSS format. Actual data is not moved. The HPSS Mover software contains additional read logic to recognize NSL UniTree data formats when an NSL UniTree file is accessed. Utilities to support migration from other legacy storage systems will also be provided as required.

Accounting

HPSS provides interfaces to collect accounting information (initially storage space utilization). These interfaces may be used by site specific programs to charge for data storage. SSM provides user interfaces to run the accounting collection utility, change account numbers and change the account code assigned to storage objects.

Summary and Status

We have described the key objectives, features and components of the HPSS architecture. At the time this paper is being written, December 1994, HPSS Release 1 (R1) is in integration testing and planning for its early deployment at several sites has begun. R1 contains all the basic HPSS components and services and supports parallel tape. It is targeted at MPP environments with existing parallel disk services. Much of the coding for Release 2 (R2) has been completed also. R2 adds support for parallel disks, migration and caching between levels of the hierarchy and other functionality. R2 will be a complete stand-alone system and is targeted for third quarter 1995.

We demonstrated, HPSS at Supercomputing 1994 with R1 and early R2 capabilities of parallel disks, and tape access (Ampex D2, IBM NTP and 3490), to an IBM SP2, IBM RS 6000, PsiTech framebuffer, and Sony high-resolution monitor over a NSC HIPPI switch. HPSS R1 is on order 95K lines of executable source code and R2 is expected to add on another 50K lines of executable source code.

Our experience indicates that the architectural choices of basing the system on the IEEE Reference Model, use of an industry defacto standard infrastructure based on OSF DCE and Transarc Encina, and use of other industry standards such as POSIX, C, Unix, ISO managed object model for Storage System Management and standard communication protocols is sound. This foundation plus the software engineering methodology employed, we believe, positions HPSS for a long and useful life for both scientific and commercial high performance environments.

Acknowledgments

We wish to acknowledge the many discussions and shared design, implementation, and operation experiences with our colleagues in the National Storage Laboratory collaboration, the IEEE Mass Storage Systems and Technology Technical Committee, the IEEE Storage System Standards Working Group, and in the storage community. Specifically we wish to acknowledge the people on the HPSS Technical Committee and Development Teams. At the risk of leaving out a key colleague in this ever-growing collaboration, the authors wish to acknowledge Dwight Barrus, Ling-Ling Chen, Ron Christman, Danny Cook, Lynn Kluegel, Tyce McLarty, Christina Mercier, and Bart Parliman from LANL; Larry Berdahl, Jim Daveler, Dave Fisher, Mark Gary, Steve Louis, Donna Mecozzi, Jim Minton, and Norm Samuelson from LLNL; Marty Barnaby, Rena Haynes, Hilary Jones, Sue Kelly, and Bill Rahe from SNL; Randy Burris, Dan Million, Daryl Steinert, Vicky White, and John Wingenbach from ORNL; Donald Creig Humes, Juliet Pao, Travis Priest and Tim Starrin from NASA LaRC; Andy Hanushevsky, Lenny Silver, and Andrew Wyatt from Cornell; and Paul Chang, Jeff Deutsch, Kurt Everson, Rich Ruef, Tracy Tran, Terry Tyler, and Benny Wilbanks from IBM U.S. Federal and its contractors.

This work was, in part, performed by the Lawrence Livermore National Laboratory, Los Alamos National Laboratory, Oak Ridge National Laboratory, and Sandia National Laboratories, under auspices of the U.S. Department of Energy Cooperative Research and Development Agreements, by Cornell, Lewis Research Center and Langley Research Center under auspices of the National Aeronautics and Space Agency and by IBM U.S. Federal under Independent Research and Development and other internal funding.

References

1. Berdahl, L., ed., "Parallel Transport Protocol," draft proposal, available from Lawrence Livermore National Laboratory, Dec. 1994.

2. Buck, A. L., and R. A. Coyne, Jr., "Dynamic Hierarchies and Optimization in Distributed Storage System," Digest of Papers, Eleventh IEEE Symposium on Mass Storage Systems, Oct. 7-10, 1991, IEEE Computer Society Press, pp. 85-91.

3. Christensen, G. S., W. R. Franta, and W. A. Petersen, "Future Directions of High-speed Networks for Distributed Storage Environments," Digest of Papers, Eleventh IEEE Symposium on Mass Storage Systems, Oct. 7-10, 1991, IEEE Computer Society Press, pp. 145-148.

4. Collins, B., et al., "Los Alamos HPDS: High-Speed Data Transfer," Proc. Twelfth IEEE Symposium on Mass Storage Systems, Monterey, April 1993.

5. Coyne, R. A., H. Hulen, and R. W. Watson, "The High Performance Storage System," Proc. Supercomputing 93, Portland, IEEE Computer Society Press, Nov. 1993.

6. Coyne, R. A. and H. Hulen, "An Introduction to the Mass Storage System Reference Model, Version 5," Proc. Twelfth IEEE Symposium on Mass Storage Systems, Monterey, April 1993.

7. Coyne, R. A., H. Hulen, and R. W. Watson, "Storage Systems for National Information Assets," Proc. Supercomputing 92, Minneapolis, Nov. 1992, pp. 626-633.

8. Dietzen, Scott, Transarc Corporation, "Distributed Transaction Processing with Encina and the OSF/DCE", Sept. 1992, 22 pages.

9. IEEE Storage System Standards Working Group (SSSWG) (Project 1244), "Reference Model for Open Storage Systems Interconnection, Mass Storage Reference Model Version 5," Sept. 1994. Available from the IEEE SSSWG Technical Editor Richard Garrison, Martin Marietta (215) 532-6746

10. "Information Technology - Open Systems Interconnection - Structure of Management Information - Part 4: Guidelines for the Definition of Management Objects," ISO/IEC 10165-4, 1991.

11. Internet Standards. The official Internet standards are defined by RFC's (TCP protocol suite). RFC 783; TCP standard defined. RFC 959; FTP protocol standard. RFC 1068; FTP use in third-party transfers. RFC 1094; NFS standard defined. RFC 1057; RPC standard defined.

12. ISO/IEC DIS 10040 Information Processing Systems - Open Systems Interconnection - Systems Management Overview, 1991.

13. Katz, R. H., "High Performance Network and Channel-Based Storage," *Proceedings of the IEEE*, Vol. 80, No. 8, pp. 1238-1262, August 1992.

14. Lampson, B. W., M. Paul, and H. J. Siegert (eds.), "Distributed Systems - Architecture and Implementation," Berlin and New York: Springer-Verlag, 1981.

15. Morris, J. H., et al., "Andrew: A Distributed Personal Computing Environment," Comm. of the ACM, Vol. 29, No. 3, March 1986.

16. Nelson, M., et al., "The National Center for Atmospheric Research Mass Storage System," Digest of Papers, Eighth IEEE Symposium on Mass Storage Systems, May 1987, pp. 12-20.

17. Open Software Foundation, Distributed Computing Environment Version 1.0 Documentation Set. Open Software Foundation, Cambridge, Mass. 1992.

18. OSF, File Systems in a Distributed Computing Environment, *White Paper, Open Software Foundation*, Cambridge, MA, July 1991.

19. Sandberg, R., et al., "Design and Implementation of the SUN Network Filesystem," Proc. USENIX Summer Conf., June 1989, pp. 119-130.

20. Tolmie, D. E., "Local Area Gigabit Networking," Digest of Papers, Eleventh IEEE Symposium on Mass Storage Systems, Oct. 7-10, 1991, IEEE Computer Society Press, pp. 11-16.

21. Watson, R. W., R. A. Coyne, "The National Storage Laboratory: Overview and Status," Proc. Thirteenth IEEE Symposium on Mass Storage Systems, Annecy France, June 12-15, 1994, pp. 39-43.

22. Witte, L. D., "Computer Networks and Distributed Systems," IEEE Computer, Vol. 24, No. 9, Sept. 1991, pp. 67-77.

APPENDIX A

Application Programming Interfaces (APIs) to HPSS Components

HPSS provides an application client library containing file, directory, and client state operations.

The HPSS Client Library provides the following routines grouped by related functionality.

API	Clients	Description
hpss_Open	client	Optionally create and open an HPSS file
hpss_Close	client	Close a file
hpss_Umask	client	Set the file creation mask
hpss_Read	client	Read a contiguous section of an HPSS file, beginning at the current file offset into a client buffer
hpss_Write	client	Write data from a client buffer to a contiguous section of an HPSS file, beginning at the current file offset
hpss_Lseek	client	Reposition the read/write file offset
hpss_ReadList	client	Read data from an HPSS file, specifying lists for data sources and sinks
hpss_WriteList	client	Write data to an HPSS file, specifying lists for data sources and sinks
hpss_Stat	client	Get file status
hpss_Fstat	client	Get file status
hpss_Lstat	client	Get file status, returning status about a symbolic link if the named file is a symbolic link
hpss_FileGetAttributes	client	Get attributes for a file
hpss_FileSetAttributes	client	Alter file attribute values
hpss_Access	client	Check file accessibility
hpss_Chmod	client	Change the file mode of an HPSS file
hpss_Chown	client	Change owner and group of an HPSS file
hpss_Utime	client	Set access and modification times of an HPSS file
hpss_GetACL	client	Query the Access Control List of a file
hpss_DeleteACLEntry	client	Remove an entry from the Access Control List of a file
hpss_UpdateACLEntry	client	Update an entry in the Access Control List of a file
hpss_Truncate	client	Set the length of a file
hpss_Ftruncate	client	Set the length of a file

hpss_Fclear	client	Clear part of a file
hpss_Cache	client	Cache a piece of a file to a specified level in the storage hierarchy
hpss_Fcache	client	Cache a piece of a file to a specified level in the storage hierarchy
hpss_Purge	client	Purge a piece of a file from a specified level in the storage hierarchy
hpss_Fpurge	client	Purge a piece of a file from a specified level in the storage hierarchy
hpss_Migrate	client	Migrate a piece of a file from a specified level in the storage hierarchy
hpss_Fmigrate	client	Migrate a piece of a file from a specified level in the storage hierarchy
hpss_Link	client	Create a hard link to an existing HPSS file
hpss_Unlink	client	Remove an entry from an HPSS directory
hpss_Rename	client	Rename a file or directory
hpss_Symlink	client	Create a symbolic link
hpss_Readlink	client	Read the contents of a symbolic link (i.e., the data stored in the symbolic link)
hpss_Mkdir	client	Create a directory
hpss_Rmdir	client	Remove an HPSS directory
hpss_Opendir	client	Open an HPSS directory
hpss_Readdir	client	Read a directory entry
hpss_Rewinddir	client	Reset position of an open directory stream
hpss_Closedir	client	Close an open directory stream
hpss_Chdir	client	Change current working directory
hpss_Getcwd	client	Get current working directory
hpss_Chroot	client	Change the root directory for the current client
hpss_LoadThreadState	client	Updates the user credentials and file/directory creation mask for a thread's API state
hpss_ThreadCleanup	client	Cleans up a thread's Client API state
hpss_Statfs	client	Returns information about the HPSS file system
hpss_AccessHandle	client	Determines client accessibility to a file, given a Name Server object handle and file pathname
hpss_OpenBitfile	client	Opens and HPSS file, specified by bitfile ID
hpss_OpenHandle	client	Open an HPSS file, specified by Name Server object ID and, optionally, pathname

hpss_GetAttrHandle	client	Get attributes of an HPSS file, specified by Name Server object ID and, optionally, pathname
hpss_SetAttrHandle	client	Set attributes of an HPSS file, specified by Name Server object ID and, optionally, pathname
hpss_GetACLHandle	client	Query the Access Control List of a file
hpss_DeleteACLEntry-Handle	client	Remove an entry from the Access Control List of a file
hpss_UpdateACLEntry-Handle	client	Update an entry in the Access Control List of a file
hpss_LinkHandle	client	Create a hard link to an existing HPSS file, given the name space object handle of the existing object, and relative directory for the new link and the pathname of the new link
hpss_LookupHandle	client	Query the Name Server to obtain attributes, an access ticket and object handle for a specified name space entry
hpss_MkdirHandle	client	Create a new directory
hpss_RmdirHandle	client	Remove a directory
hpss_ReaddirHandle	client	Read directory entries
hpss_UnlinkHandle	client	Remove directory entry
hpss_RenameHandle	client	Rename a directory entry
hpss_SymlinkHandle	client	Create a symbolic link
hpss_ReadlinkHandle	client	Read the contents of a symbolic link
hpss_TruncateHandle	client	Set the length of a file
hpss_StageHandle	client	Stage a piece of a file to a specified level in the storage hierarchy
hpss_PurgeHandle	client	Purge a piece of a file from a specified level in the storage hierarchy
hpss_MigrateHandle	client	Migrate a piece of a file from a specified level in the storage hierarchy

The Name Server provides *APIs* for the following operations:

API	Clients	Description
ns_Insert	client	Insert a bitfile object into a directory
ns_Delete	client	Delete a name space object
ns_Rename	client	Rename a name space object
ns_MkLink	client	Create a hard link to file
ns_MkSymLink	client	Make a symbolic link
ns_ReadLink	client	Read data associated with a symbolic link
ns_GetName	client	Get path name for the specified bitfile
ns_GetACL	client	Get an ACL for the specified name server object
ns_SetACL	client	Set an ACL for the specified name server object
ns_DeleteACLEntry	client	Delete an entry from the ACL of the specified name server object
ns_UpdateACLEntry	client	Update an entry from the ACL of the specified name server object
ns_Mkdir	client	Create a directory
ns_ReadDir	client	Return a list of directory entries
ns_GetAttrs	SSM, client	Get Name Server handle and managed object attributes
ns_SetAttrs	SSM, client	Set Name Server managed object attributes

The Bitfile Server provides *APIs* for the following operations:

API	Client	Description
bfs_Create	client	Create a bitfile
bfs_Unlink	client	Unlink a bitfile
bfs_Open	client	Open a bitfile
bfs_Close	client	Close a bitfile
bfs_Read	client	Read data from a bitfile
bfs_Write	client	Write data to a bitfile
bfs_BitfileGetAttrs	SSM, client	Get bitfile managed object attributes
bfs_BitfileSetAttrs	SSM, client	Set bitfile managed object attributes
bfs_BitfileOpenGetAttrs	SSM, client	Get bitfile managed object attributes (for an open bitfile)
bfs_BitfileOpenSetAttrs	SSM, client	Set bitfile managed object attributes (for an open bitfile)
bfs_ServerGetAttrs	SSM, client	Get (common) server managed object attributes
bfs_ServerSetAttrs	SSM, client	Set (common) server managed object attributes
bfs_Copy	Migration, client	Copy storage segments for a bitfile to the next storage hierarchy level
bfs_Copy	Migration, client	Move storage segments for a bitfile to the next storage hierarchy level
bfs_Purge	Purge, client	Reclaim space (i.e., purge segments) occupied by a bitfile

The Storage Server provides *APIs* for the following operations:

API	Clients	Description
ss_BeginSession	BFS, SSM, client	Start a storage server session
ss_EndSession	BFS, SSM, client	End a storage server session
ss_SSCreate	BFS, client	Create a storage segment
ss_SSUnlink	BFS, client	Delete a storage segment
ss_SSRead	BFS, client	Read data from a storage segment
ss_SSWrite	BFS, client	Write data to a storage segment
ss_SSGetAttrs	BFS, client	Get storage segment managed object attributes
ss_SSSetAttrs	BFS, client	Set storage segment managed object attributes
ss_SSMount	Migrate, Repack, SSM, client	Mount a storage segment and assign it to a session
ss_SSUnmount	Migrate, Repack, SSM, client	Unmount a storage segment
ss_SSCopySegment	Migrate, SSM, client	Copy storage segment to new segment on different virtual volume
ss_SSMoveSegment	Migrate, Repack, SSM, client	Move storage segment to new virtual volume
ss_MapCreate	SSM, client	Create storage map for a virtual volume
ss_MapDelete	SSM, client	Delete storage map for a virtual volume
ss_MapGetAttrs	SSM, client	Get storage map managed object attributes
ss_MapSetAttrs	SSM, client	Set storage map managed object attributes
ss_VVCreate	SSM, client	Create a virtual volume
ss_VVDelete	SSM, client	Delete a virtual volume
ss_VVMount	SSM, client	Mount a virtual volume
ss_VVUnmount	SSM, client	Unmount a virtual volume
ss_VVRead	SSM, client	Read a virtual volume
ss_VVWrite	SSM, client	Write a virtual volume
ss_VVGetAttrs	SSM, client	Get virtual volume managed object attributes
ss_VVSetAttrs	SSM, client	Set virtual volume managed object attributes
ss_PVCreate	SSM, client	Create a physical volume
ss_PVDelete	SSM, client	Delete a physical volume
ss_PVMount	SSM, client	Mount a physical volume
ss_PVUnmount	SSM, client	Unmount a physical volume
ss_PVRead	SSM, client	Read a physical volume
ss_PVWrite	SSM, client	Write a physical volume
ss_PVGetAttrs	SSM, client	Get physical volume managed object attributes

ss_PVSetAttrs	SSM, client	Set physical volume managed object attributes
ss_SSrvGetAttrs	SSM, client	Get Storage Server specific managed object attributes
ss_SSrvSetAttrs	SSM, client	Set Storage Server specific managed object attributes
ss_ServerGetAttrs	SSM, client	Get (common) server managed object attributes
ss_ServerSetAttrs	SSM, client	Set (common) server managed object attributes

The Mover provides *APIs* for the following operations:

API	Clients	Description
mvr_Read	SS, PVL, client	Read data from a device or devices
mvr_Write	SS, PVL, client	Write data to a device or devices
mvr_DeviceSpec	SS, client	Load a physical volume
		Unload a physical volume
		Load message to device's display area
		Flush data to media
		Write tape mark
		Read media label
		Write media label
		Clear portion of disk
mvr_DeviceGetAttrs	SS, SSM, client	Get Mover device managed object attributes
mvr_DeviceSetAttrs	SS, SSM, client	Set Mover device managed object attributes
mvr_MvrGetAttrs	SSM, client	Get Mover specific managed object attributes
mvr_MvrSetAttrs	SSM, client	Set Mover specific managed object attributes
mvr_ServerGetAttrs	SSM, client	Get (common) server managed object attributes
mvr_ServerSetAttrs	SSM, client	Set (common) server managed object attributes

The Physical Volume Library provides *APIs* for the following operations:

API	Clients	Description
pvl_Mount	client	Synchronously mount a single volume
pvl_MountNew	SS, client	Begin creating a set of volumes to automatically mount
pvl_MountAdd	SS, client	Add a volume to the set of volumes to be mounted
pvl_MountCommit	SS, client	Mount a set of volumes
pvl_MountCompleted	PVR	Notify the PVL a pending mount has completed
pvl_CancelAllJobs	SS, SSM, client	Cancel all jobs associated with a connection handle
pvl_DismountJobId	SS, SSM, client	Dismount all volumes associated with a specific job
pvl_DismountVolume	SS, SSM, client	Dismounts a single volume
pvl_DismountDrive	SSM, client	Forces the dismount of a specified drive
pvl_Import	SSM, client	Imports a new cartridge into HPSS
pvl_Export	SSM, client	Exports a cartridge from HPSS
pvl_Move	SSM, client	Move a cartridge from one PVR to another
pvl_NotifyCartridge	PVR	Notify the PVL that a cartridge has been check in or out of a PVR
pvl_WriteVolumeLabel	SS, SSM, client	Rewrite the internal label of a specified volume
pvl_AllocateVol	SS, SSM, client	Allocate a volume to a particular client
pvl_ScratchVol	SS, SSM, client	Return a volume to the scratch pool
pvl_DriveGetAttrs	SSM, client	Get drive managed object attributes
pvl_DriveSetAttrs	SSM, client	Set drive managed object attributes
pvl_VolumeGetAttrs	SSM, client	Get volume managed object attributes
pvl_VolumeSetAttrs	SSM, client	Set volume managed object attributes
pvl_QueueGetAttrs	SSM, client	Get PVL request queue managed object attributes
pvl_QueueSetAttrs	SSM, client	Set PVL request queue managed object attributes
pvl_RequestGetAttrs	SSM, client	Get PVL request queue entry managed object attributes
pvl_RequestSetAttrs	SSM, client	Set PVL request queue entry managed object attributes
pvl_PVLGetAttrs	SSM, client	Get PVL specific managed object attributes
pvl_PVLSetAttrs	SSM, client	Set PVL specific managed object attributes
pvl_ServerGetAttrs	SSM, client	Get (common) server managed object attributes
pvl_ServerSetAttrs	SSM, client	Set (common) server managed object attributes

The Physical Volume Repository provides *APIs* for the following operations:

API	Clients	Description
pvr_Mount	PVL, client	Asynchronously mount a single volume
pvr_MountComplete	PVL, client	Notify PVL a requested mount has completed
pvr_DismountCart	PVL, client	Dismount a single cartridge
pvr_DismountDrive	PVL, client	Dismount the cartridge in a given drive
pvr_Inject	PVL, SSM, client	Accept a new cartridge into the PVR
pvr_Eject	PVL, SSM, client	Eject a cartridge from the PVR
pvr_Audit	SSM, client	Audit all or part of a repository checking external cartridge labels when possible
pvr_LocateCartridge	PVL, client	Verify whether or not a PVR manages a cartridge
pvr_SetDrive	PVL, client	Takes drives in the PVR on-line or off-line
pvr_CartridgeGetAttrs	SSM, client	Get a cartridge managed object attributes
pvr_CartridgeSetAttrs	SSM, client	Set a cartridge managed object attributes
pvr_PVRGetAttrs	SSM, client	Get PVR specific managed object attributes
pvr_PVRSetAttrs	SSM, client	Set PVR specific managed object attributes
pvr_ServerGetAttrs	SSM, client	Get (common) server managed object attributes
pvr_ServerSetAttrs	SSM, client	Set (common) server managed object attributes
pvr_ListPendingMounts	SSM, client	List all currently pending mounts for the PVR

The Storage System Manager provides *APIs* for the following operations:

API	Clients	Description
ssm_Adm	client	Perform administrative request on one or more servers (shut down, halt, mark down, reinitialize, start)
ssm_AttrGet	client	Get managed object attributes
ssm_AttrReg	client	Register an SSM client to receive notifications of data change in managed objects
ssm_AttrSet	client	Set managed object attributes
ssm_Checkin	client	Accept checkins from data server clients
ssm_Checkout	client	Accept checkouts from data server clients
ssm_ConfigAdd	client	Add a new entry to a configuration files
ssm_ConfigDelete	client	Delete an entry from a configuration file
ssm_ConfigUpdate	client	Update a configuration file entry
ssm_Delog	client	Allow accept to the delog command

ssm_DriveDismount	client	Dismount a drive
ssm_JobCancel	client	Cancel a Physical Volume Library job
ssm_CartImport	client	Import cartridges into the Physical Volume Library
ssm_CartExport	client	Export cartridges from the Physical Volume Library
ssm_ResourceCreate	client	Create resources (physical volume, virtual volume, and storage map) in the Storage Server
ssm_ResourceDelete	client	Delete resources (physical volume, virtual volume, and storage map) from the Storage Server
ssm_AlarmNotify	Logging	Receive notifications of alarms
ssm_EventNotify	Logging	Receive notifications of events
ssm_MountNotify	PVL	Receive notifications of tape mounts and dismounts
ssm_BitfileNotify	BFS	Receive bitfile data change notifications
ssm_CartNotify	PVR	Receive cartridge data change notifications
ssm_DeviceNotify	PVL	Receive device data change notifications
ssm_DriveNotify	PVL	Receive drive data change notifications
ssm_LogfileNotify	Logging	Receive log file data change notifications
ssm_MVRNotify	Mvr	Receive Mover specific data change notifications
ssm_MapNotify	SS	Receive storage map data change notifications
ssm_NSNotify	NS	Receive Name Server specific data change notifications
ssm_PVNotify	SS	Receive physical volume data change notifications
ssm_PVRNotify	PVR	Receive PVR specific data change notifications
ssm_QueueNotify	PVL	Receive PVL queue data change notifications
ssm_RequestNotify	PVL	Receive PVL request entry data change notifications
ssm_SFSNotify	Metadata Manager	Receive SFS data change notifications
ssm_SSNotify	SS	Receive storage segment data change notifications
ssm_ServerNotify	NS, BFS, SS, Mvr, PVL, PVR, Logging	Receive common server data change notifications
ssm_SsrvNotify	SS	Receive Storage Server specific data change notifications
ssm_VVNotify	SS	Receive virtual volume data change notifications
ssm_VolNotify	PVL	Receive volume data change notifications
ssm_Migrate	client	Move storage segments for a bitfile to the next storage hierarchy level
ssm_Purge	client	Reclaim space occupied by bitfiles

ssm_Repack	client	Perform defragmentation of physical volumes
ssm_MoveCart	client	Move a cartridge from one PVR to another
client_notify	client	Notify clients of alarms, events, mount requests, managed object data changes, and special System Manager requests

The following managed objects have attributes which may be queried (and set) by SSM:

Name Server	Volume
Bitfiles	Physical Volume Library queue
Bitfile Server (common)	Physical Volume Library request entry
Storage segments	Physical Volume Library server specific
Storage maps	Physical Volume Library Server (common)
Virtual volumes	Cartridge
Physical volumes	Physical Volume Repository server specific
Storage Server specific	Physical Volume Repository Server (common)
Storage Server (common)	Security server
Mover device	Log Daemon server (common)
Mover server specific	Log Client server (common)
Mover server (common)	Structured File Server
Drive	

The Storage System Manager also receives the following type of notifications from the HPSS server components:

Alarms	Tape mounts
Events	Data changes for registered object attributes

Some of the more important management operations which may be performed by the Storage System Manager include:

Import/create resources	Repack
Import cartridges	Delog
Export cartridges	Set devices online/offline
Move cartridges	Dismount drive
(from one PVR to another)	
Audit PVR	Start/stop/reinitialize/halt servers
Migrate	Configure servers
Purge	Define/modify ACLs

Migration/Purge provides *APIs* for the following operations:

API	Clients	Description
migr_StartMigration	SSM, client	Start migration for a particular storage class
migr_StartPurge	SSM, client	Start purge for a particular storage class
migr_MPSGetAttrs	SSM, client	Get the migration-purge server attributes
migr_MPSSetAttrs	SSM, client	Set the migration-purge server attributes
migr_ServerGetAttrs	SSM, client	Get (common) server managed object attributes
migr_ServerSetAttrs	SSM, client	Set (common) server managed object attributes

A 500 MegaByte/Second Disk Array

Thomas M. Ruwart and Matthew T. O'Keefe
University of Minnesota
Army High Performance Computing Research Center
Graphics and Visualization Laboratory
1100 Washington Avenue South
Minneapolis, MN 55415
+1-612-626-8091
+1-612-625-4583 (fax)
tmr@ahpcrc.umn.edu
okeefe@everest.ee.umn.edu

Abstract

Applications at the Army High Performance Computing Research Center's (AHPCRC) Graphics and Visualization Laboratory (GVL) at the University of Minnesota require a tremendous amount of I/O bandwidth and this appetite for data is growing. Silicon Graphics workstation are used to perform the post-processing, visualization, and animation of multi-terabyte size datasets produced by scientific simulations performed on AHPCRC supercomputers. The M.A.X. (Maximum Achievable Xfer) was designed to find the maximum achievable I/O performance of the Silicon Graphics CHALLENGE/Onyx-class machines that run these applications. Running a fully configured Onyx machine with 12 - 150MHz R4400 processors, 512MB of 8-way interleaved memory, 31 fast/wide SCSI-2 channels each with a Ciprico disk array controller we were able to achieve a maximum sustained transfer rate of 509.8 megabytes per second. However, after analyzing the results it became clear that the true maximum transfer rate is somewhat beyond this figure and we will need to do further testing with more disk array controllers in order to find the true maximum.

Introduction

The Silicon Graphics CHALLENGE/Onyx computer system has an enormous I/O bandwidth that, to our knowledge, has not been fully explored. Researchers at the AHPCRC are working on projects that require significant I/O bandwidth from these computer systems [Woodward93]. We performed several experiments to find the total sustainable I/O bandwidth of the CHALLENGE/Onyx computer systems that are key to these projects. These high-end workstations are now achieving transfer rates that are competitive with mainframe architectures and given their attractive price/performance may potentially become the primary data servers in future high performance computing

environments. Our goal was to find the I/O performance limits for large sequential transfers on the SGI CHALLENGE/Onyx workstation.

The cost of putting together enough high-speed disk subsystems to push the limits of the I/O bandwidth was expensive and remains so to this day. A fully configured CHALLENGE/Onyx computer system could support 32 fast-wide SCSI-2 channels each with 20 MBytes/second[1] of I/O bandwidth. Each SCSI channel would require a minimum of 5 high performance disk drives to saturate the 32 SCSI channels sufficiently to find the maximum I/O bandwidth. This would require a total of 160 disks which implies a great deal of device management and bus contention if these devices are not managed properly.

Instead of using individual disk drives, we connected a single high-speed disk array controller to each of 31 SCSI channels[2] on the Onyx system. These disk array controllers are much easier to obtain than disks and fewer of them are needed due to their individual high bandwidth. Furthermore, each disk array controller can easily saturate a single fast/wide SCSI-2 channel so fewer devices are needed (one per channel) resulting in less device management overhead.

Experimental Setup

Software

- IRIX Version 5.2, a UNIX SystemV Release 4 derivative
- lv - The Silicon Graphics Logical Volume Device Driver

Hardware

Onyx System Configuration
The system used in this experiment was a Silicon Graphics Onyx machine with the following configuration:

- 20 150 MHz R4400 Processors (12 Processors for 8-way interleaved memory configuration)
- CPU: MIPS R4400 Processor Chip Revision: 5.0
- FPU: MIPS R4010 Floating Point Chip Revision: 0.0
- Data cache size: 16 Kbytes
- Instruction cache size: 16 Kbytes
- Secondary unified instruction/data cache size: 1 Mbyte
- Main memory size: 512 Mbytes, 4- and 8-way interleaved
- 4 IO4 Power Channels
- 32 Fast-Wide Differential SCSI-2 channels
- 2GB System disk on SCSI channel 1

An Onyx system is basically a CHALLENGE with a graphics engine. Since this experiment did not make use of the graphics engine in the Onyx at any time, these results can be considered equally valid for a CHALLENGE.

---

[1]MBytes/second = 1,000,000 bytes per second.
[2]Only 23 of the 24 available channels were used due to a minor cabling oversight on the part of the experimenters.

Ciprico Disk Array and Diskless Array Description
The disk devices used in this experiment were Ciprico RF6710 disk arrays. Each RF6710 disk array is a RAID-3 device made up of 8 data drives plus 1 parity drive[Ciprico 93][Patterson89]. The number and type of disk arrays used were:

- 8 real disk arrays populated with Seagate ST12400N 2.5GB 3.5-inch disks.
- 23 diskless arrays populated with simulated Seagate Barracuda-2 2.5GB 3.5-inch disks.

Because the number of disks required to populate 31 disk array controllers was more than we could purchase or borrow, there were no disks on 23 of the 31 disk array controllers. Instead, they were programmed to act like *real* disk arrays when accessed. The *diskless* array controllers read and wrote data as any disk array would with the exception that data written to the diskless arrays was thrown away and data read was always zero. Consequently, no file system testing was possible and all testing was performed on *raw* devices.

The diskless array controllers have geometry characteristics based on the ST12400N disks but performance characteristics based on an array populated with Seagate Barracuda-2 disks, the higher performance version of the ST12400N disk. The data read from the array is always zero with the exception of the first 512-byte block on the array which will be kept in the controller memory and contains the volume header information.

The performance of the diskless array controllers depends on the type of access. For purely sequential access the seek and rotational latencies are zero. This is because on array controllers with real disks, sequential read and write operations make effective use of the data caches on the individual disks thus hiding rotational and seek delays. For any other access that involves a seek, an appropriate delay was inserted in the command processing to simulate the seek and rotational latencies. The seek time is estimated to be proportional to the seek distance and the rotational delay is set to half a revolution (4.1 milliseconds in this case). The disk drive being modeled is a Seagate Barracuda-2. The seek simulation feature was used for a different set of experiments but was not used in the M.A.X. experiment.

For sequential read operations, the performance of the diskless arrays was only 4% higher than an array disk real disks at moderate to large request sizes (Figure 1). Sequential write operations on the diskless arrays performed nearly identically to the read operations. It should be noted that the objective of this experiment was not to simulate a disk array but rather to saturate the I/O subsystem. Therefore, these performance differences are more of a benefit than a detriment.

Finally, the read operations on the real disk arrays perform better than write operations on real disks even when the write caches are used (Figure 2). However, this difference seems to be reasonably constant for small request sizes and becomes less significant at larger request sizes.

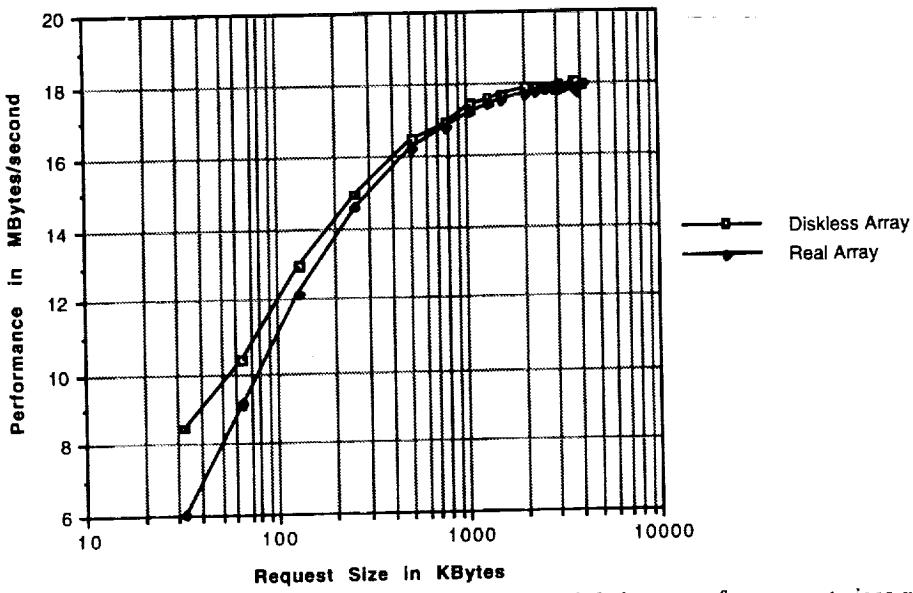


Figure 1. *Performance of read operations for diskless and real disk arrays for request sizes ranging from 32KBytes to 4096KBytes. At the lower request sizes, the diskless arrays are considerably faster than the real disk arrays. However, the performance curves converge at request sizes of 512KBytes and higher.*

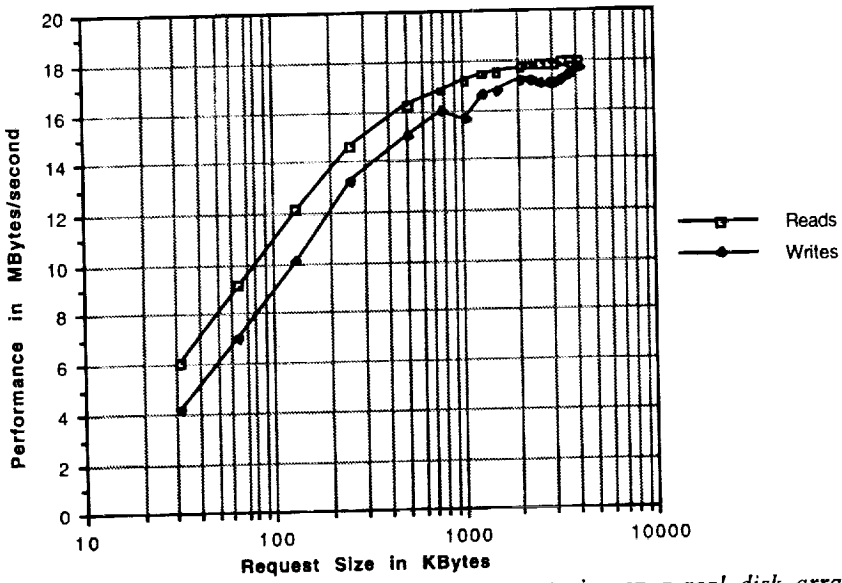


Figure 2. *Performance of a reads and writes versus request size on a real disk array. The write operations are cached on each of the individual disk drives within the array.*

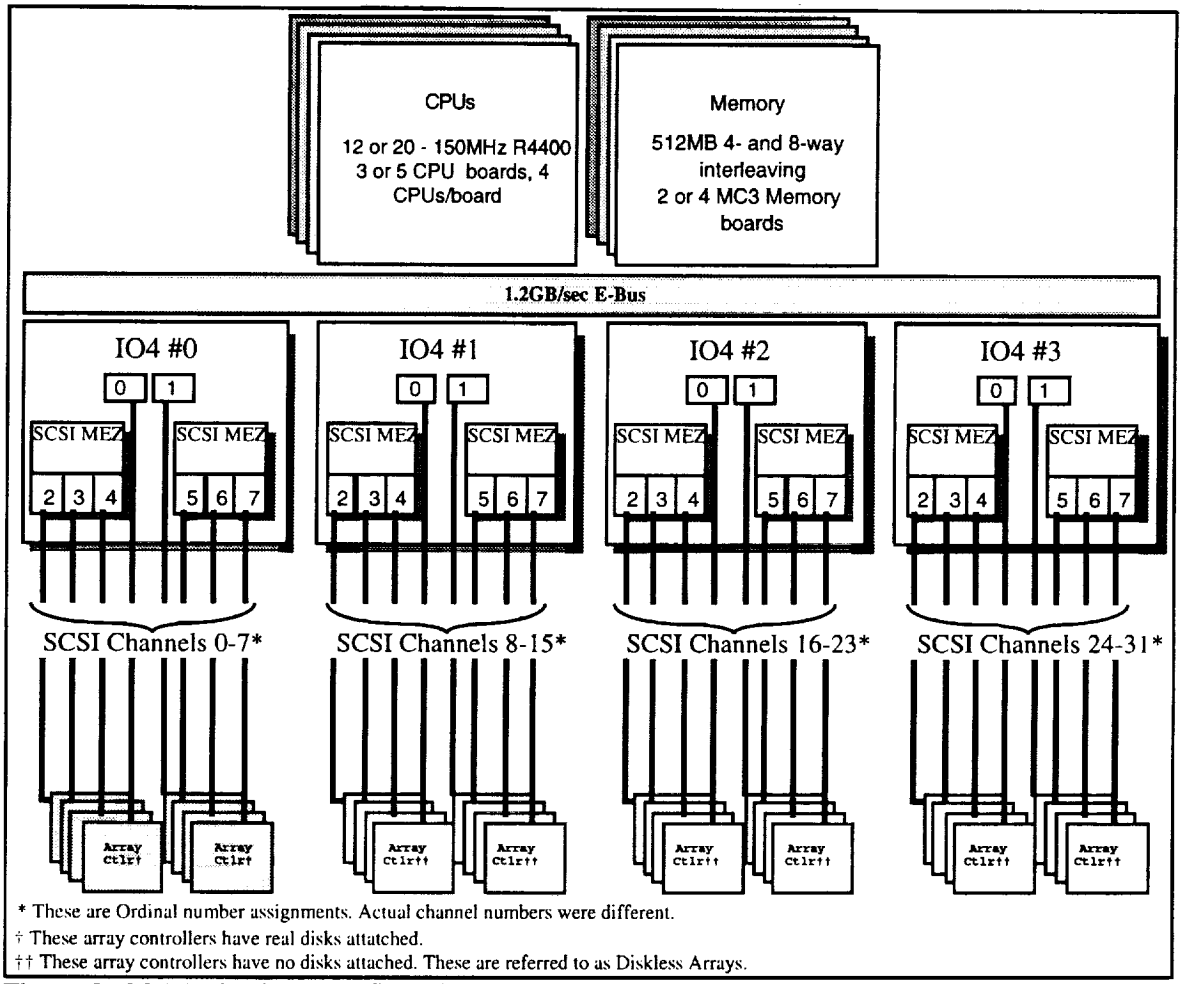


Figure 3. M.A.X. hardware configuration diagram.

Performance Evaluation Program

xdd - An I/O performance measurement tool

xdd is a program developed to measure I/O performance by reading or writing large amounts of data sequentially from a file or raw device. This program is intended to find the upper limit of performance of an I/O subsystem under specific, well-controlled operating parameters. *xdd* takes as command line parameters the target device to operate on, the operation to perform (read or write), the request size to use for each read/write operation, the number of read/write requests to perform, and the number of times to repeat the test in order to obtain a good statistical average. Furthermore, *xdd* can be instructed to limit the time to run each test in order to make the runtime more deterministic.

xdd provides three measures of I/O performance: (1) an aggregate transfer rate, (2) a table of time stamps detailing each request, and (3) the number of I/O operations completed during the test. Upon completion, *xdd* prints a single line of values indicating the request size (in 1024-byte blocks) , the average, high, and low I/O performance in units of 10^6-bytes per second, the number of I/O operations, the average, maximum, and minimum number of seconds to complete the specified number of requests, and the number of errors that occurred during the test.

The first set of performance values is the aggregate transfer rate and can be affected erroneously by individual I/O operations that may have "stalled" due to some outside influence. To help identify these outlying values a collection of high resolution time stamps are recorded in a file for further analysis. Before each I/O operation has been initiated, a time stamp is recorded in an internal memory array. This array is pre-allocated and page locked in order to avoid any paging interference that may negatively affect these values. After *xdd* has completed all passes of the requested test, the time stamp values are written to a file with header that contains the request size in 1024-byte blocks, the resolution of the time stamp values, and the number of time stamp entries.

In an attempt to minimize the impact of virtual memory management and process scheduling, the *xdd* text and data areas, the I/O buffer, and the time stamp table are page locked during initialization to avoid any page faults or program swapping during the performance test. The program also sets itself to a non-degrading, high priority in order to reduce scheduling side effects on the measurements.

xdd uses a single page-aligned memory buffer large enough to handle a single request. An I/O request to a single disk can range in size from 512-bytes up to a system defined maximum. Currently, this maximum is set to 4 MBytes (4*1024*1024 bytes), or more appropriately, 1024 pages[3]. The IRIX operating system allocates 1024 page mapping registers for each I/O request but in order to map any arbitrary 4MB I/O request, 1025 page mapping registers are required to map requests that do not start on page boundaries. Therefore, in order to issue an I/O request of 4MB it is necessary to page align the buffer to insure it can be mapped in 1024 page mapping registers.[4]

The Experiment

First, a test utilizing eight fast/wide SCSI-2 channels on a single IO4[5] was run to determine if the IO4 imposed any bandwidth limitations on the eight channels. The aggregate performance scaled linearly as the number of independently fully utilized channels was increased from 1 to 8. Hence, there are no bandwidth limitations within an IO4.

The principle testing involved three basic *access methods*. The first access method was the simultaneous *independent* access of 1 to 31 disk array controllers. The second access method used the Silicon Graphics Logical Volume (lv) striping device driver to access 2 to 31 devices as a single logical device. The third access method was a variation of the first whereby half the disk arrays would be reading data into memory while the other half would be writing data from memory to disk. This last test was intended to measure any bi-directional interference.

Each of these tests were performed using 4- and 8-way memory interleaving. The greater the interleaving, the higher the effective bandwidth into memory. Figure 4 describes the overall experimental test layout..

---

[3]The page size in IRIX 5.x is fixed at 4096-bytes.

[4]This problem with one to few page mapping registers exists in IRIX 5.2 but may not exist in later releases.

[5] The IO4 card has 4 Fast/Wide SCSI-2 channels.

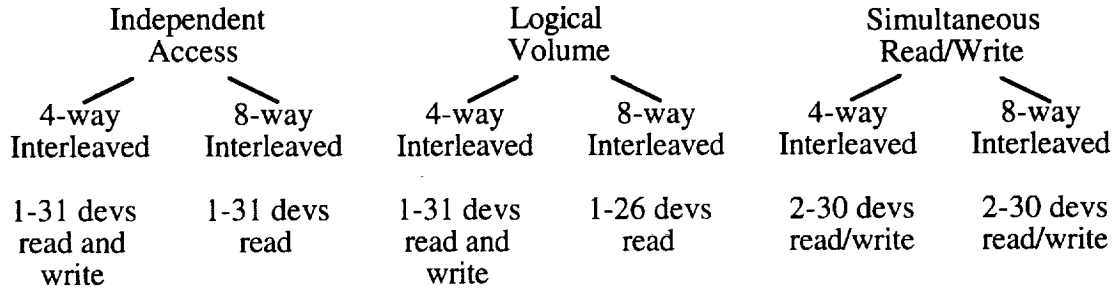


Figure 4. *The access methods and system memory configurations.*

Due to time constraints, write operations were not tested for the 8-way interleaved Independent Access and Logical Volume tests. However, it was observed in the 4-way interleaved memory testing that the overall write performance tended to be slightly better than the read performance. It is believed that this characteristic holds true for the 8-way interleaved memory as well although it still needs to be verified.

Caveats

- In order to accommodate a shorter than expected testing schedule the 2-way interleaved memory testing was removed.
- The fully configured Onyx with 4-way interleaved memory was able to accommodate 20 processors (5 processor boards). However, the 8-way interleaved memory configuration required 2 extra memory boards that displaced 2 processor boards reducing the number of CPUs to 12 for this configuration. However, it should be noted that this would not be necessary on a CHALLENGE server which can be configured with 36 CPUs, 8-way interleaved memory, and 4 IO4s simultaneously.
- The diskless array controllers were measured to be about 4% faster than the real disk arrays at the top end of their performance curve (18.1 MBytes/sec versus 17.85 MBytes/sec).
- It is interesting to note that even with only 12 CPUs on the 8-way interleaved memory configuration, the I/O rate did not appear to be limited by the CPU performance.

Results

The results are presented by access method as described in figure 4. First the Independent Access results are presented (figures 5-9) followed by the Logical Volume results (figures 10-16) and finally the Simultaneous Read/Write results are presented (figure 17).

Independent Access Results

The total bandwidth of the 4-way interleaved memory configuration was tested by increasing the number of independently accessed arrays from 1 to 31 over request sizes ranging from 64KBytes[6] to 4096KBytes. Disk array controllers were added one at a time incrementing monotonically through each IO4 until all channels were running. This procedure was repeated for the 8-way interleaved memory configuration.

[6] 1 KByte = 1024 bytes.

This access method yielded the best overall performance when compared to the logical volume and simultaneous read/write access methods. The 4-way interleaved memory configuration peaked at 392 MBytes/second accessing 27 devices with a request size of 768KBytes, dropping to 310 MBytes/second as more devices were added (figures 5-6). The 8-way interleaved memory configuration performance was measured at 509.8 MBytes/second accessing 31 devices with a request size of 2048KBytes (figures 7-8). Request size has a definite effect on the performance with request sizes larger than 512KBytes performing the best (figure 9).

Due to time constraints, testing was limited to read operations only.

Logical Volume Read and Write Tests

This series of tests were run to measure the read and write performance of logical volumes composed of 9 to 30 devices. Since a previous study [Ruwart93] characterized the read performance of logical volumes composed of 2 to 8 devices it was decided to start where that study left off in the interest of time.

The results of these tests are reported as Performance as a function of number of devices at two different *step sizes*. The step size of a logical volume is the maximum amount of data read off a single disk array in a single request. Thus, from the disk array's perspective, the step size is equivalent to a request size because this is what the disk array sees as a request from the host. The amount of data the *xdd* application actually requests from the logical volume was intentionally set to the step size times the number of devices in the logical volume in order to insure that all devices in the logical volume would be accessed for each application I/O request in the most optimal manner.

As expected, the larger step size of 1024KBytes performed better than the smaller step size of 256Kbytes (figures 10-15). However, the performance did not seem to depend on the type of operation (figures 12 and 15) and only slightly on the memory interleaving (figure 16). The peak performance of the logical volume access method was about 240 MBytes/second.

Simultaneous Read/Write Tests

The simultaneous read/write tests were run to measure any bi-directional interference when transferring data to and from different groups of I/O devices simultaneously. The motivation behind this testing has to do with large multi-media servers that must sustain a large bandwidth in *and* out of a system.

The results show a peak performance of 482 MBytes/second accessing a total of 30 disk arrays: 15 reading plus 15 writing using a request size of 1536KBytes and 8-way interleaved memory (figure 17). This is 97% of the straight read performance of 30 independent disk arrays. The 3% difference is attributed to the slightly lower performance of the individual disk array write operations (see figure 2). Since 15 of the 30 devices were writing data in the simultaneous read/write case, the aggregate performance of all 30 disk arrays is less than if all 30 devices were reading.

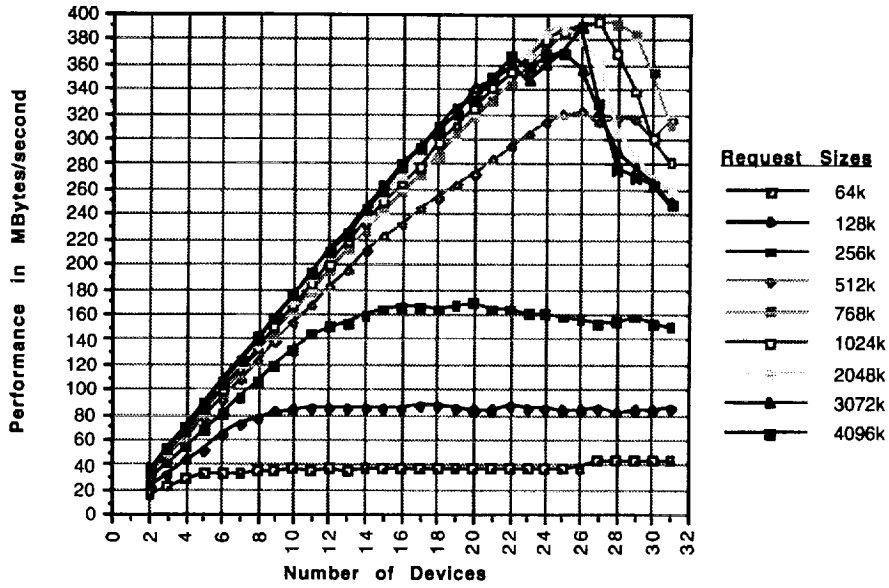


Figure 5. *The performance curves for read operations using request sizes 64-4096-KBytes using 4-way interleaved memory. The performance peaked at 393 MBytes/second using 27 devices with a request size of 768-KBytes.*

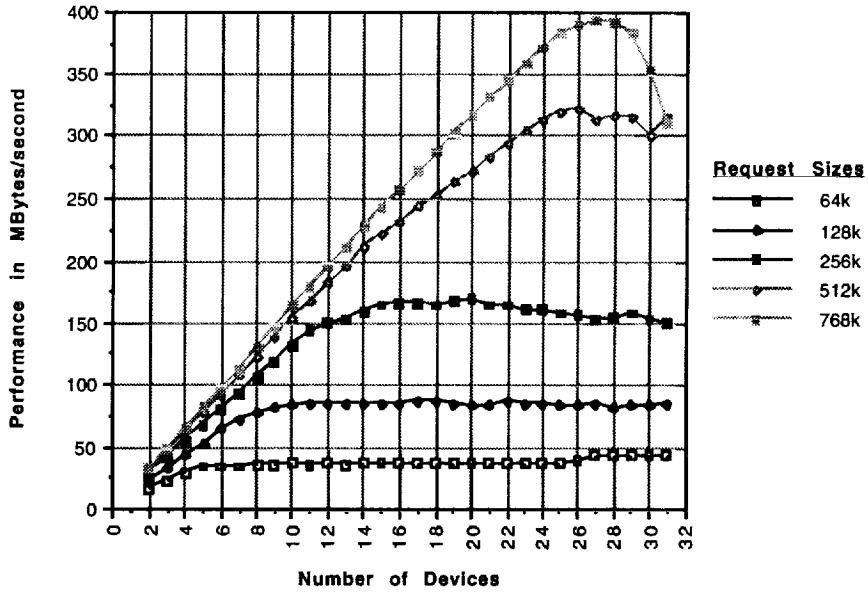


Figure 6. *The performance curves for read operations using request sizes 64-768-KBytes using 4-way interleaved memory. The performance peaked at 393 MBytes/second using 31 devices with a request size of 768-KBytes.*

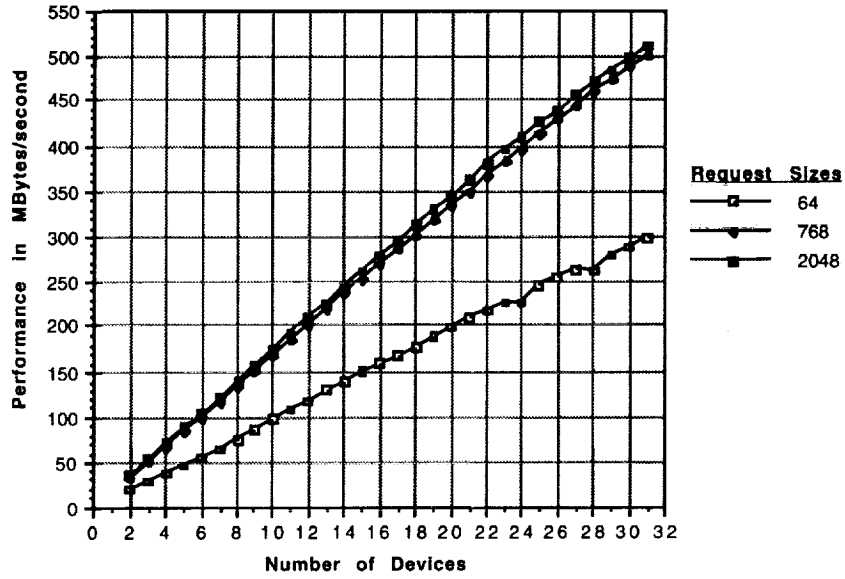


Figure 7. *The performance curves for read operations using request sizes 64, 768, and 2048-KBytes using 8-way interleaved memory. The performance peaked at 509.8 MBytes/second using 31 devices with a request size of 2048-KBytes.*

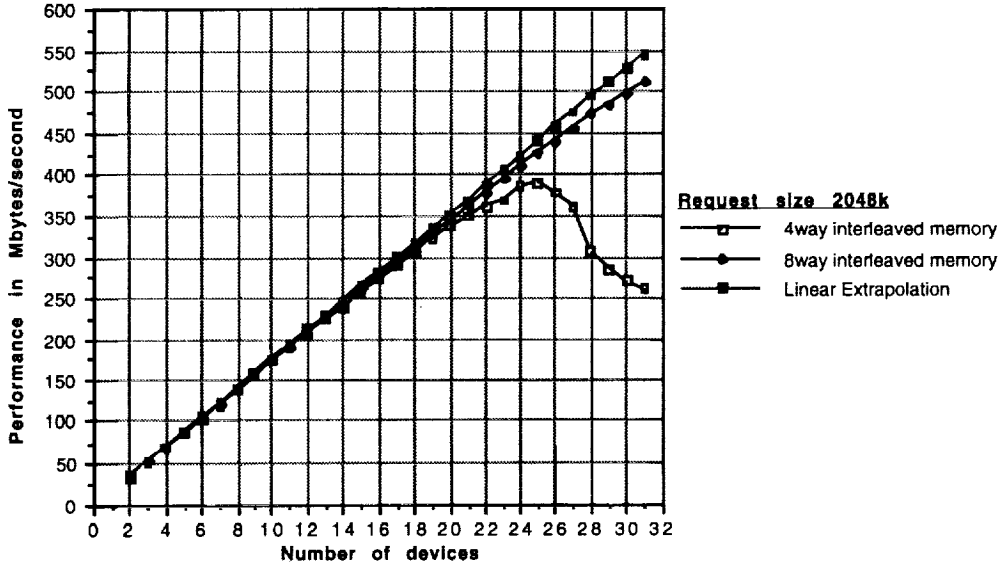


Figure 8. *The performance curves for read operations using a request size of 2048 KBytes, 8-way versus 4-way interleaved memory, for independent processes accessing 2 to 31 disk arrays. The performance using 4-way interleaved memory tracks the 8-way performance curve up to 390MBytes per second where it drops off noticeably while the 8-way performance curve continues with no signs of tapering off.*

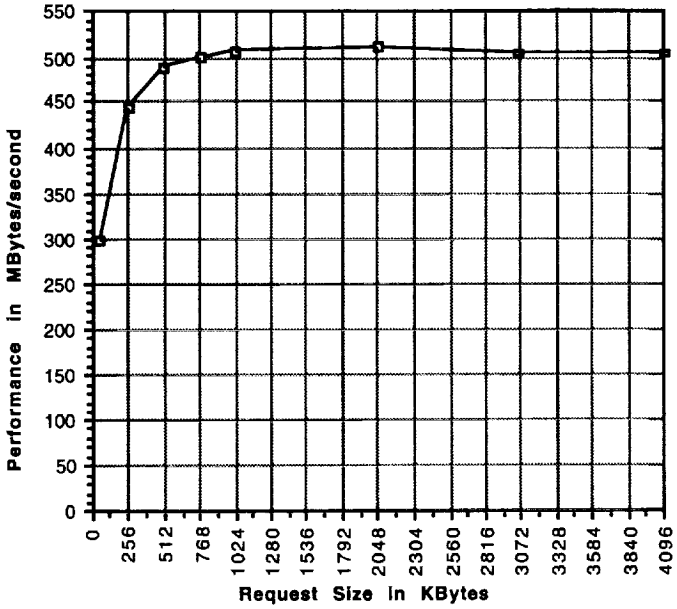


Figure 9. *The performance curve for read operations using request sizes 64-4096-KBytes using 8-way interleaved memory. The performance peaked at 509.8 MBytes/second using 31 devices with a request size of 2048-KBytes*

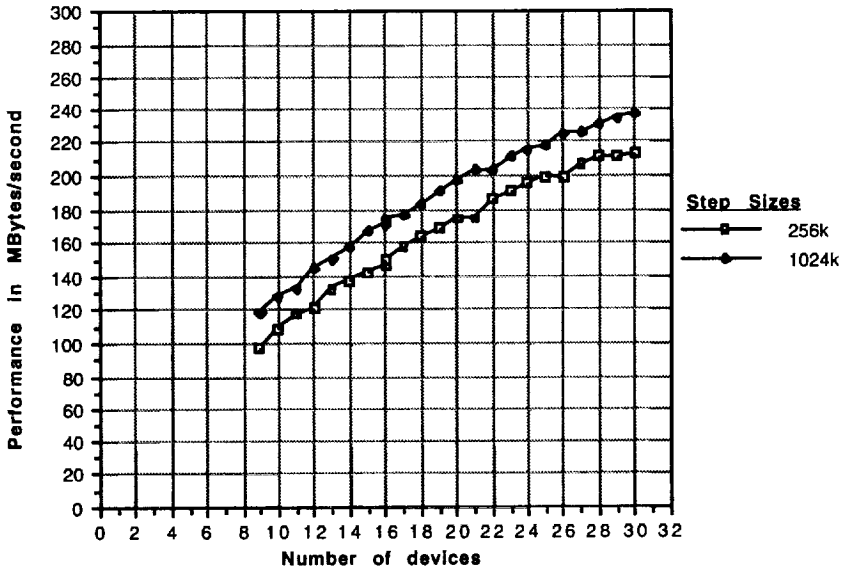


Figure 10. *The performance curves for read operations using step sizes 256 KBytes and 1024 KBytes, 4-way interleaved memory, and a single logical volume consisting of 9 to 30 disk arrays. The performance peaked at 236.9 MBytes/second using 30 devices with a step size of 1024-KBytes.*

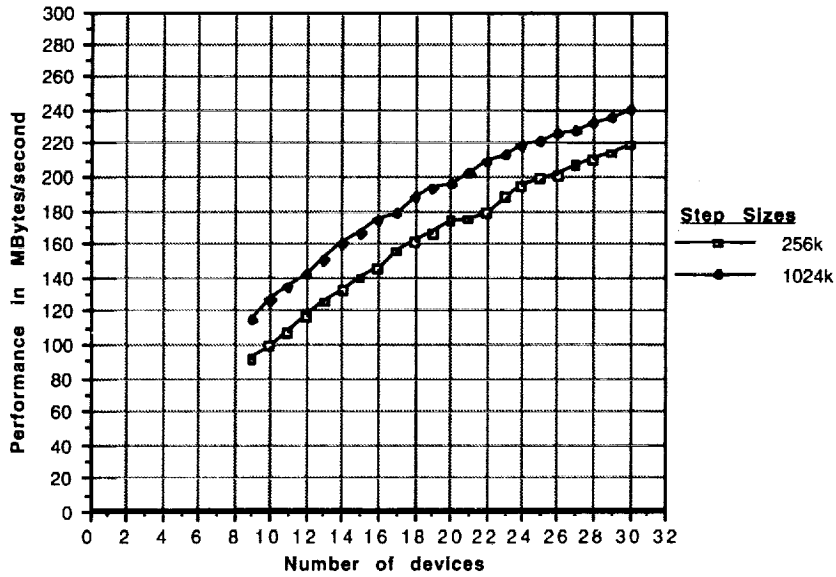


Figure 11. *The performance curves for write operations using step sizes 256 KBytes and 1024 KBytes, 4-way interleaved memory, and a single logical volume consisting of 9 to 30 disk arrays. The performance peaked at 241 MBytes/second using 30 devices with a step size of 1024-KBytes*

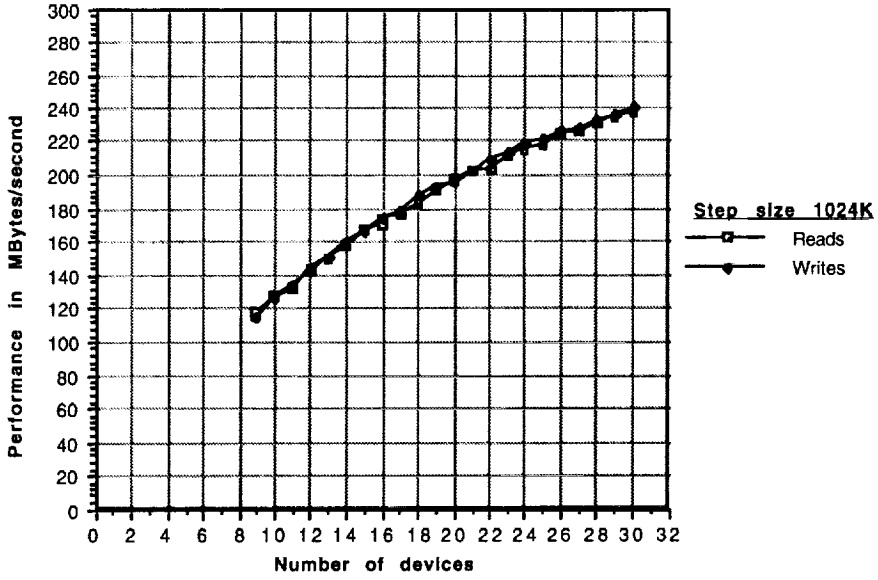


Figure 12. *The performance curves for read and write operations using a step size of 1024 KBytes, 4-way interleaved memory, and a single logical volume consisting of 9 to 30 disk arrays. The performance of the write operations was slightly better than the read operations.*

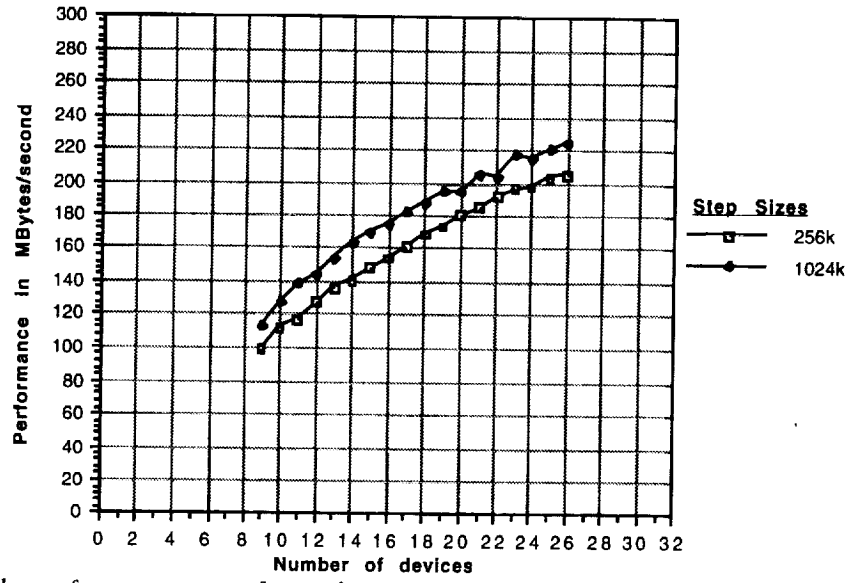


Figure 13. *The performance curves for read operations using step sizes 256 KBytes and 1024 KBytes, 8-way interleaved memory, and a single logical volume consisting of 9 to 26 disk arrays. The performance peaked at 223.9 MBytes/second using 26 devices with a step size of 1024-KBytes.*

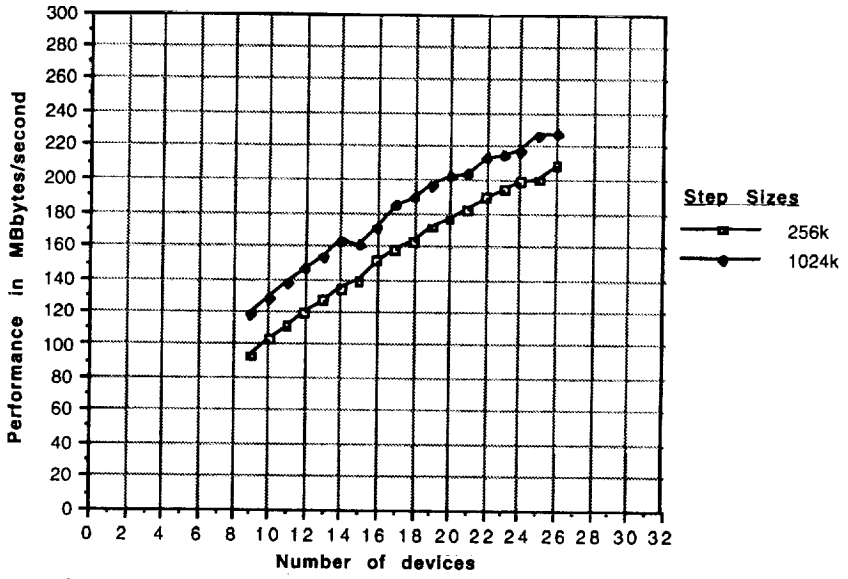


Figure 14. *The performance curves for read operations using step sizes 256 KBytes and 1024 KBytes, 8-way interleaved memory, and a single logical volume consisting of 9 to 26 disk arrays. The performance peaked at 228.2 MBytes/second using 26 devices with a step size of 1024-KBytes.*

C-2

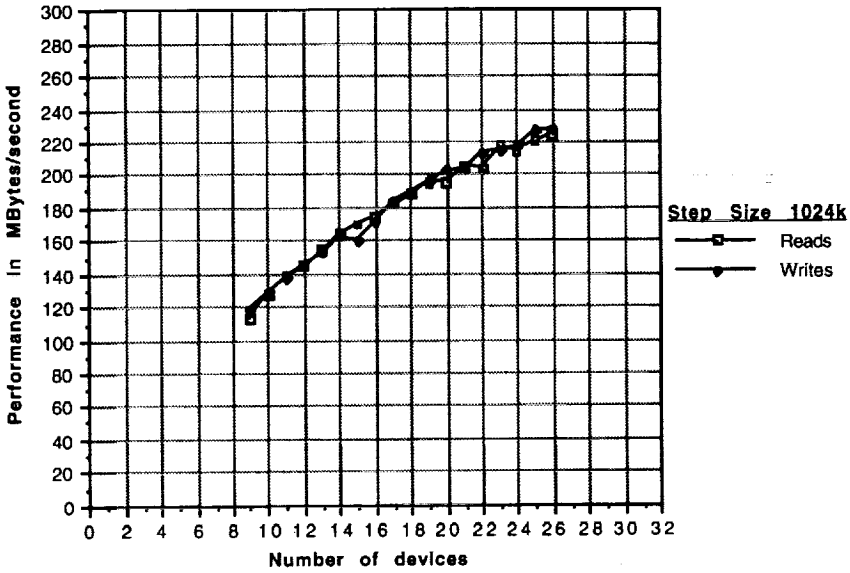


Figure 15. *The performance curves for read and write operations using a step size of 1024 KBytes, 8-way interleaved memory, and a single logical volume consisting of 9 to 26 disk arrays. The performance of the write operations was slightly better than the read operations in most cases but are still very close*

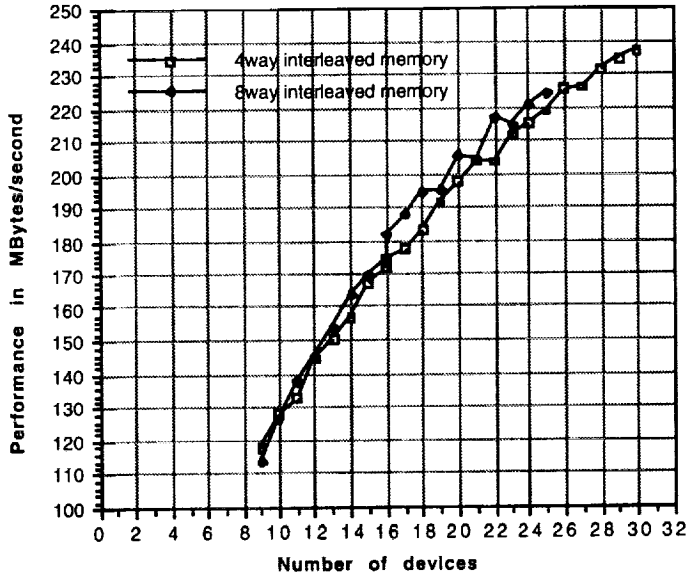


Figure 16. *The performance curves for read operations using a step size of 1024 KBytes, 8-way versus 4-way interleaved memory, and a single logical volume consisting of 9 to 30 disk arrays (26 for the 8-way case). The performance using 8-way interleaved memory was slightly better than the 4-way interleaved memory configuration.*

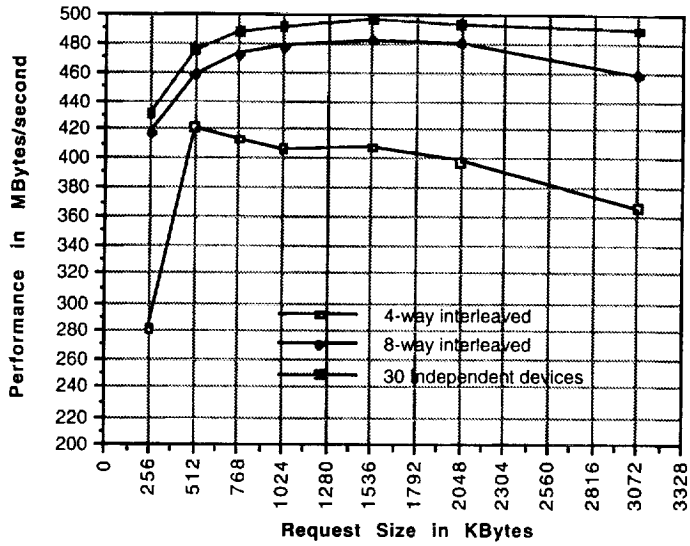


Figure 17. *The performance curves for simultaneous read and write operations using a request sizes ranging from 128KBytes to 3072KBytes, 8-way versus 4-way interleaved memory, reading from 15 disk arrays while writing to 15 other disk arrays. The performance using 8-way interleaved memory was consistently better than the 4-way interleaved memory configuration. The top curve represents read operations on 30 independent devices.*

Conclusions

The M.A.X. experiment demonstrated a sustained performance of 509.8 MBytes/second reading data from 31 independent disk arrays simultaneously into an 8-way interleaved memory subsystem on the CHALLENGE/Onyx system. However, the *maximum* achievable transfer rate was not observed because 31 disk arrays were not enough to saturate the I/O subsystem. This statement is based on the results for the 4-way interleaved memory configuration whereby the performance hits a maximum and degrades as more devices are added. This effect was not observed for the 8-way interleaved memory configuration. Therefore, we believe that the actual *maximum* I/O performance of the CHALLENGE/Onyx is greater than 510 MBytes/second.

The logical volume testing showed a maximum transfer rate of approximately 240 MBytes/second for reading or writing. The memory configuration did not have any effect on the overall performance of any logical volume configuration.

Finally, the simultaneous read/write tests demonstrated a maximum performance of 482 MBytes/second using 30 disk arrays: reading from 15 while simultaneously writing to 15 others. Since this performance is measured over 30 devices, it is estimated that 31 devices would provide an additional 16 MBytes/second for a total sustained performance of 498 MBytes/second.

The M.A.X. experiment was a success and exceeded our expectations inasmuch as we expected to observe a peak performance less than 500 MBytes/second. Had we known that the peak would have been higher we would have designed the experiment to utilize far more disk array controllers and SCSI-2 channels. The Silicon Graphics CHALLENGE/Onyx system architecture has proven to have a very efficient I/O subsystem that has a tremendous usable bandwidth.

Future Work / Related Work

- *Perform 8-way interleaved memory testing on a CHALLENGE and more processors, 6 IO4's, and 48 fast/wide SCSI-2 channels with a theoretical peak bandwidth of 960MBytes/second.*
- *File System Testing with 160 Real Disks and/or 32 Real Disk Arrays*
- *Testing with Multiple 100-MByte/second HiPPI and/or Fibre Channel Devices*
- *Bit rate consistency testing for multimedia applications*

Acknowledgments

We would like to acknowledge Silicon Graphics, Inc. for providing the hardware required to attach the disk arrays to the Onyx machine and Ciprico, Inc. for providing the disk array controllers and engineering that went into making them believe they had real disks attached.We thank Jeff Stromberg and Steve Soltis for their hard work in taking the measurements. This work was supported by the U.S. Army and by grant no. 5555-23 from the University Space Research Association which is administered by NASA's Center for Excellence in Space Data and Information Sciences (CESDIS) at the NASA Goddard Space Flight Center.

References

[Ciprico93] ``RF6700 Controller Board Reference Manual," Publication Number 21020236 A, Ciprico, Inc., Plymouth, MN, August 1993.

[Paterson89] D.A. Paterson, P.M. Chen, G. Gibson, and R.H. Katz, ``Introduction to redundant arrays of inexpensive disks (raid)," Proc. IEEE Compcon, Spring 1989.

[Ruwart93] T.M.Ruwart and M.T. O'Keefe, ``Performance Characteristics of a 100MB/second Disk Array," Army High Performance Computing Research Center Preprint Series, no. 93-123, 1993.

[Woodward93] P.R. Woodward, ``Interactive Scientific Visualization of Fluid Flow,"IEEE Computer, vol. 6, no. 10, pp. 13-26, October 1993.

New Architectural Paradigms for Multi-PetaByte Distributed Storage Systems

Richard R. Lee
Data Storage Technologies, Inc.
Post Office Box 1293
Ridgewood, New Jersey USA 07451-1293
Tel: (201)-670-6620
FAX:(201)-670-7814
e-mail: rrl@dst.com

Abstract
In the not too distant future, programs such as NASA's Earth Observing System, NSF/ARPA/NASA's Digital Libraries Initiative and the Intelligence Community's (NSA, CIA, NRO, etc.) mass storage system upgrades will all require multi-petabyte (or larger) distributed storage solutions. None of these requirements, as currently defined, will meet their objectives utilizing either today's' architectural paradigms or storage solutions. Radically new approaches will be required to not only store and manage these veritable "mountain ranges of data", but to make the cost of ownership affordable, much less practical in today's (and certainly the future's) austere budget environment!

Within this paper we will explore new architectural paradigms and project systems performance benefits and $/PB of information stored. We will discuss essential "top down" approaches to achieving an overall systems level performance capability sufficient to meet the challenges of these major programs.

Foreword

Today's data center is growing at a rate of per year of 40% CAGR, without even factoring in the impact of new multi-media and imagery-on-demand applications. This means that someone with a 10 TB problem today will have a 100 TB problem in 2-3 years and a multi-Petabyte problem in 7-10 years. Many of the large data centers found today have multi-PetaByte problems already. Based on this growth new exponential factors must be defined in order to understand the magnitude of the problem. Based on new exponent prefixes defined in the past two years, we have compiled a listing for reference throughout our discussions.

TeraByte: 1012 Bytes of bitfile data
PetaByte: 1015 Bytes of bitfile data
ExaByte: 1018 Bytes of bitfile data
ZettaByte: 1021 Bytes of bitfile data
YottaByte: 1024 Bytes of bitfile data

Near-Term Programs with Storage Requirements in Excess of 1 PetaByte

Within the federal end-user community today there are a number of requirements for multi-PetaByte archival systems already. A number of these will be based on years and years of data gathering by numerous Earth resources and imagery satellites producing warehouses of bitfiles which will be made available to thousands of researchers worldwide. For purposes of our discussion we will profile a sampling of the more visible ones.

NASA EOSDIS: Part of NASA's "Mission to Planet Earth", EOSDIS is a 13 site (8 directly associated with the program and 5 affiliated) distributed archive and data center for earth science data. This program has a data ingest, product generation & data distribution rate in excess of 1000 GB per day, with a 15+ year life span i.e. 11 PetaBytes anticipated over the program's life.

NASA EDOS: All incoming Level 0 data from EOS and International Partner satellite platforms is collected at this site in WV for archiving and processing into higher order data products. It is then distributed to the 13 EOSDIS sites (as well as IP sites upon request). Level 0 and higher order data products in excess of 1 TB per day will be archived, processed and distributed from this site over the 15+ life span of the program. Total archive capacity will exceed 1 PB during this time.

NSF/ARPA/NASA Digital Libraries Initiative: Envisioned as the "Data Malls on the Information Superhighway" these distributed information infrastructure servers will provide fast access to thousands of TB's of data, and will open the infrastructure up to the general public. They are intended to capture, store, distribute and provide access to every type of bitfile data available from public and private sources. Given the scope of this plan it is envisioned that this will comprise hundreds to thousands of PB's over its useful life.

NII - "High Resolution Video on Demand Services": As one of the most visible components of the National Information Infrastructure concept, this application has been embraced by the entire telecommunications and computer industries as well as capturing a significant share of the NII federal funding dollars available, and the public's mindshare as well.

Using the most advanced image compression techniques available today can only reduce the large size of a "digital movie" to 10's of GB's (assuming higher resolutions than found in conventional broadcast today). This nets out to a requirement of multiple PB's in key VOD locations serving major metropolitan areas across the country (each Blockbuster Video location currently houses in excess of 10,000 feature length movie titles).

The Intelligence Community's Consolidation of Disparate Archives: Hard to describe in any other terms, the United States' Intelligence Community (CIA, NSA, NRO, DIA, etc.) is faced with dilemma of providing higher and faster levels of service to its end-users with less capital to work with (dollars and personnel). In total, the IC ingests over 4 TB per day from classified sources alone (1 TB+ per day from images that are approximately 1 GB each), not to mention the thousands of unclassified sources worldwide that are routinely accessed. In trying to meet the needs of their end-users they must respond to numerous real-time queries across disparate resources. All combined, the IC has in excess of 10 PB of data already archived, with this growing at a much higher rate than that of the rest of end-user community (60+% CAGR).

Current Storage System Architectural Paradigms i.e.*"Multi-TeraByte Class"* Types

- Direct Connected Peripherals i.e. "The Mainframe Era"
- Stand-alone Data Servers i.e. "The Client/Server Mantra
- Network Attached Peripherals i.e. "The NSL Approach"
- IEEE Mass Storage Systems Reference Model i.e. "The Open Systems Standard"

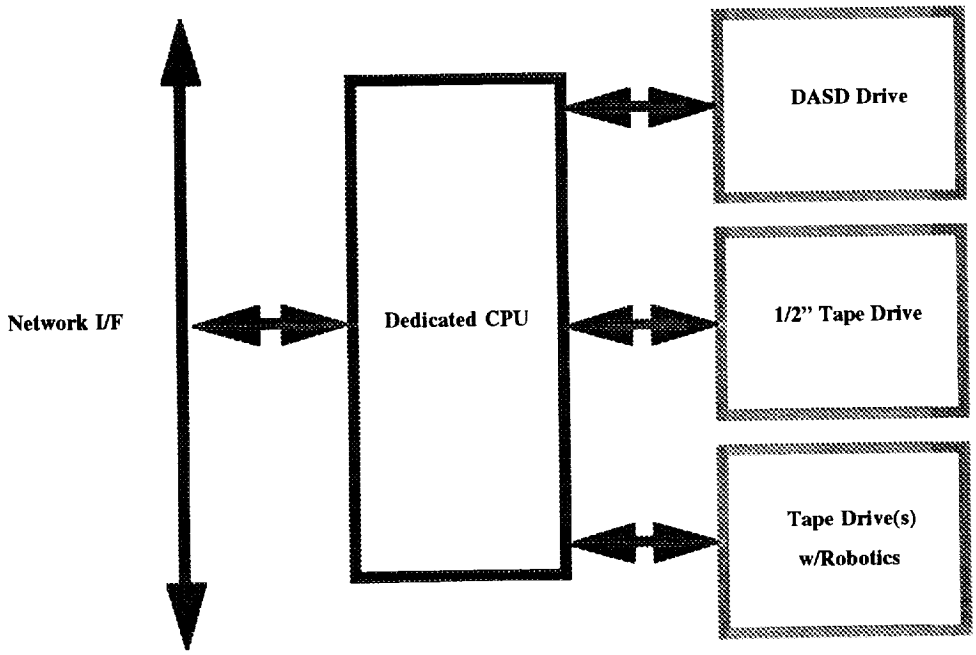


Figure 1: "TeraByte Class" Data Server

- **Overview**

Systems conforming to the first two of these types of architectural paradigms (Mainframe attached and Client/Server) are essentially CPU Centric and appear as a centralized repository of bitfiles to the outside world. Bitfiles may be distributed out over a network to various clients, but still originate from a centralized location. The IEEE Mass Storage Reference Model promises to break this scheme into either a distributed or quasi-distributed one, but all implementations fielded to date behave in a centralized manner and will potentially fall apart when distributed.

In short, these systems all suffer from the same type of performance limitation; that of acting as a single point of access for all classes of service. The controlling/serving CPU can only maintain one connection/DMA access at a time in practical terms and even through the use of multiple CPU's and multi-threaded OS's one can only maintain a small number of transfers simultaneously (mostly due to shared memory and operating system software limitations) appearing on an effective basis as a single point of access to the network.

Systems based on the NSL/HPSS paradigm (network attached peripherals) are designed to support high-speed transfers of large bitfiles, but do not translate to a distributed environment and are far too costly for the mainstream of the end-user community. For this reason we have classified them as part of the TeraByte class.

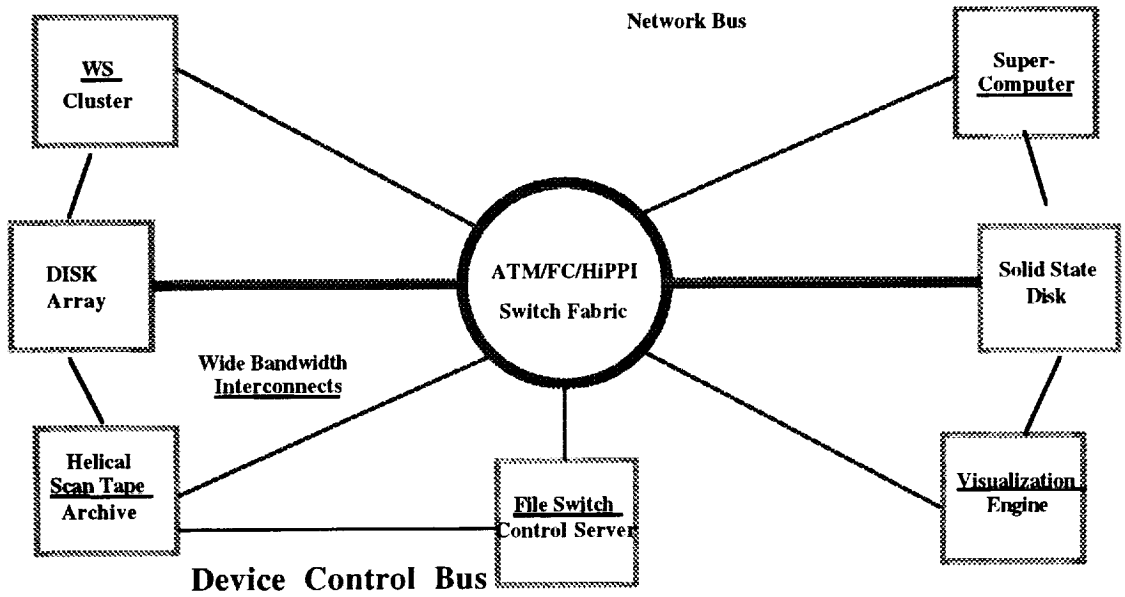


Figure 2: "The NSL/HPSS Paradigm"

Concepts for Future Storage System Architectural Paradigms: "Multi-PetaByte and Beyond"

In order to meet the challenges of managing multi-PetaByte distributed archives we need to think beyond the current COTS mindset and explore new approaches altogether, some based on concepts being used in parallel computing today (using however, COTS components where practical). We feel that a parallel architecture eliminates much of the problem encountered with "single point of access" found in traditional architectures of the day. Much of what we will present is still in the early stages of development, but does represent a logical approach to the problem at hand.

- **Distributed Cluster-type**

This architecture envisions an environment where a clustered array of servers are interconnected via a LAN to a series of data repositories. These servers are in turn connected to a WAN and serve clients and other servers distributed throughout the enterprise. Each repository contains multiple peripherals and robotics assemblies for contention free search and access of bitfiles. Using fast packet technology, the system is capable of storing and retrieving bitfiles within the repositories at very high packet rates, but at a relatively low cost. Utilizing this type of architecture allows for many points of access, while retaining the benefits of using commodity type technologies.

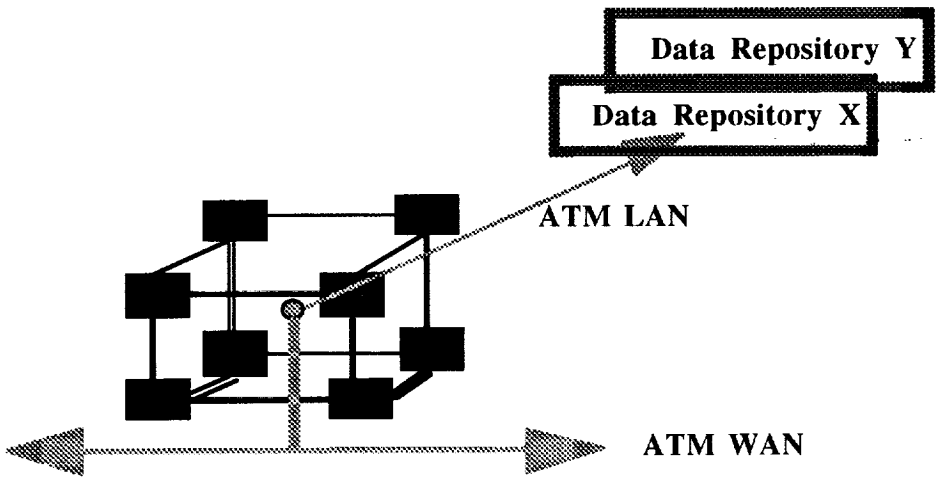


Figure 3: - A Distributed "Cluster -type" Query/Bitfile Server

- **Scalable Parallel**

This architectural approach borrows much from today's scalable processors i.e. shared memory parallelism. The system is essentially demand driven and each process automatically adapts itself to the number of resources (CPU's and peripherals) available to the user at the time of the request. This architectural approach is totally scalable and higher levels of performance can be obtained by merely adding more CPU's and peripherals i.e. forward extensibility without obsolescence.

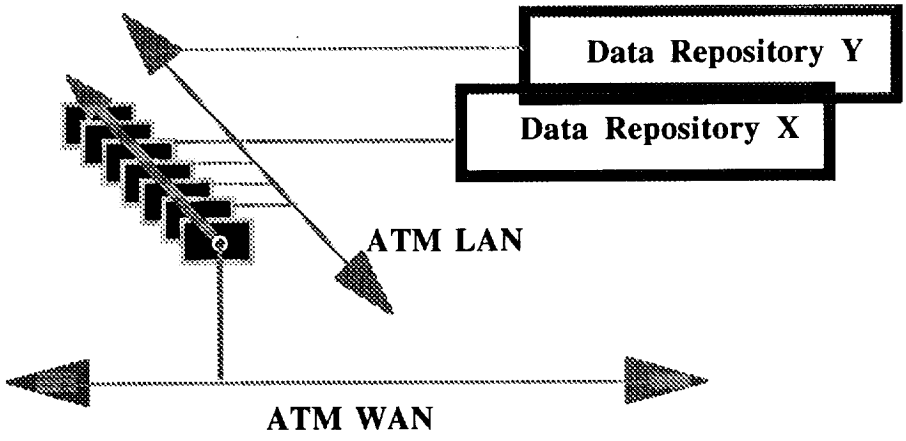


Figure 4: - "A Scalable Parallel Query/Bitfile Server"

• **Dynamically Configurable**

As implied by its name, this architectural approach is the most flexible in meeting "data on demand" requirements. The system configures itself dynamically depending upon end-user demand and resources available. During times of extremely high demand the system configures itself as highly parallel, while during periods of light-medium demand it acts as a clustered resource. The benefits of this approach are that it eliminates single-node bottlenecks (the slowest component of a distributed system throttles the performance of the entire system) and acts as a high-availability resource under all load conditions.

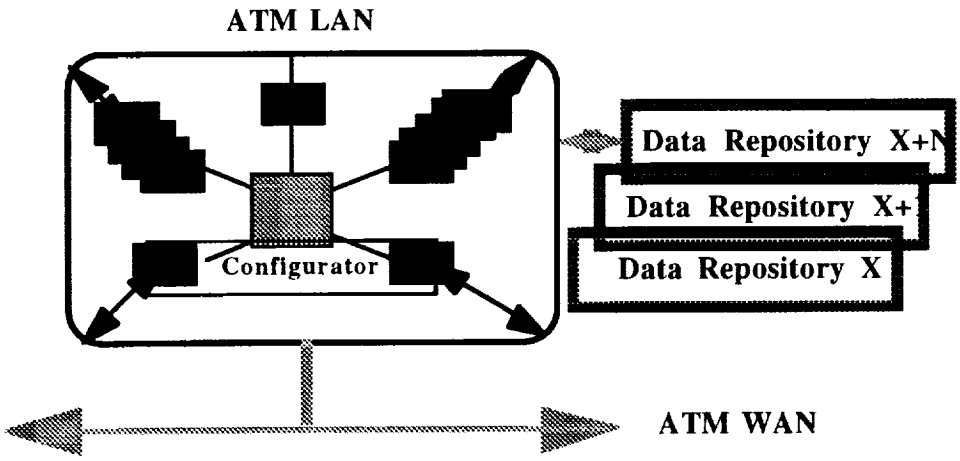


Figure 5: - "A Dynamically Configurable Query/Bitfile Server"

The concepts that we have discussed here are by no means new or all encompassing. Rather, they are shown as examples of wide departures from the status quo which seems to pervade the mindset of today's systems planners and developers as the only approach available to meet the challenges set forth. We expect that as everyone's eyes are opened wider to both the scope of the challenge as well as the tools available to respond to it, that new mindsets will develop.

Additional Considerations

Adoption of new hardware architectural paradigms alone will not suffice to meet the challenges of these ever increasing requirements. We will need to accomplish the following in parallel with these developments;

- Adopt Object Driven files systems for faster query, search and access to bitfiles
- Continue to develop "bandwidth on demand" driven internetworks and storage peripherals
- Eliminate all "single point of access" failures and bottlenecks
- Utilize distributed Metadata and Browse data db's
- Migration to higher order data transfer and communications protocols
- Achieve continuing incremental reductions in Unit Storage Costs with attendant increases in Capacity-per-physical unit and vastly improved data reliability.

• Achieve continuing incremental reductions in Unit Storage Costs with attendant increases in Capacity-per-physical unit and vastly improved data reliability.

Cost Projections & Realities:

Based on the use of conventional architectures and components, we project that most end-users are looking at fielded system costs of $40-60M per PetaByte, with the majority of these costs being centered around expensive CPU's, network fabrics and high-end peripherals. This level of cost is far too high for most, if not all budgets today and does not include the manpower or materials necessary to operate and maintain these systems over their useful life (a major component of total cost).

We believe that in order for the key programs discussed earlier to be achievable, that costs in the $10-20$/PetaByte range must be achieved. This can only be realized by embracing radical new approaches similar to what we have outlined.

Conclusions and Recommendations:

Current architectural approaches "bottom out" when tasked at multi-PetaByte levels (access, bandwidth, file management, cost, etc.).

Scalable and dynamically Configurable hardware architectures off significant promise in overcoming many of these limitations.

In addition, exponential increases in hardware, software and protocol efficiencies are mandated to meet this challenge as well.

In short, "The ways of the past must give way to the needs of the future" i.e. the familiar and comfortable path of the present will not suffice.

References;

[1] Lee, R. and Dan Mintz, "Grand Challenges in Mass Storage - A Systems Integrators Perspective", Second NASA Goddard Conference on Mass Storage Systems and Technologies, Greenbelt, MD, September 1992

[2] Lee, R., "The Future of Mass Storage", THIC Winter Meeting, San Diego, CA, January 1993

[3] Lee, R., "New Architectural Paradigms for Multi-PetaByte Distributed Storage Systems", Massive Digital Data Systems Workshop, Reston, VA, February 1994/Supercomputing '94, Washington, D.C., November 1994

[4] Kuhn, T., *"The Structure of Scientific Revolution"*, University of Chicago Press, Chicago, IL 1970

[5] Lee, R., 19mm Helical Scan Recording Technology for Data Intensive Computing Environments", 10th IEEE Symposium on Mass Storage Systems (vendor poster session), Monterey, CA, May 1990

[6] Coleman, S. and R.W. Watson, "The Emerging Paradigm Shift in Storage System Architectures", review copy for Proceedings of the IEEE, April 1993

[7] Coyne, R. , H. Hulen and R. Watson, "Storage Systems for National Information Assets", Proceedings- Supercomputing '92, Minneapolis, MN, November 1992

[8] Lee, R., "Mass Storage - the key to success in high performance computing" , Convex File Server Seminars, Milan/Rome, Italy, February 1993/Third NASA Goddard Conference on Mass Storage Systems and Technologies, College Park, MD, October 1993

[9] Lee, R., "19mm Data Storage Applications", THIC Fall Meeting, Annapolis, MD, October 1990

[10] Panel Discussions, Mass Storage Roundtable, Supercomputing '94, Washington, D.C., November 1994

[11] EOSDIS Core System Science Information Architecture "White Paper" Doc #FB9401V2, Hughes AIS, Inc., Landover Maryland, March 1994

[12] Dixon, Dick, "Statement of Requirements of the European Mass-Storage Specification Working Group Working Version 1.1" , European Weather Centre, June, 1994

[13] Teaff, Danny, "The High Performance Storage System", IBM U.S. Federal Publication

[14] IEEE Storage Systems Standards Working Group, "Mass Storage Systems Reference Model Version 5", IEEE Computer Society Mass Storage Systems and Technology Committee, Balloting Draft, July 1994

[15] "National Science Foundation's MetaCenter", NSF Division of Advanced Scientific Computing, NSF Publications, Arlington, VA., 1994

[16] "Program Guideline/Program Briefing", ARPA/NASA/NSF Research on Digital Libraries Initiative, Arlington, VA, September/December '93

[17] Convex Exemplar System Overview, DOC 080-002293-000 V1.1, Convex Computer Corporation, Richardson, Texas, 1994

Optimizing Raid Performance With Cache

Alex Bouzari, President
Mega Drive Systems, Inc.
489 S. Robertson Boulevard
Beverly Hills, CA 90211
phone: (310) 842-9616 fax: (310) 247-0006
e-mail: abouzari@uu1201.megadrive.com

We live in a world of increasingly complex applications and operating systems. Information is increasing at a mind-boggling rate. The consolidation of text, voice, and imaging represents an even greater challenge for our information systems. Which forces us to address three important questions: Where do we store all this information? How do we access it? And, how do we protect it against the threat of loss or damage?

Introduced in the 1980s, RAID (Redundant Arrays of Independent Disks) represents a cost-effective solution to the needs of the information age. While fulfilling expectations for high storage, and reliability, RAID is sometimes subject to criticisms in the area of performance. However, there are design elements that can significantly enhance performance. They can be subdivided into two areas: 1) RAID levels or basic architecture. And, 2) enhancement schemes such as intelligent caching, support of tagged command queuing, and use of SCSI-2 Fast and Wide features.

Host-independent hardware-based RAID

There are three types of disk arrays: 1) hardware-based, host-independent; 2) hardware-based, host-dependent; and, 3) software-based, host-dependent. Software- based disk arrays are very taxing on the CPU because most of the processing is done in the host computer. On the other hand, hardware-based, host-dependent RAID systems fall short by foregoing the host-side benefits of SCSI.

Therefore, this article will focus on host-independent, hardware-based disk arrays as they typically provide significantly improved overall performance (as measured by throughput and I/Os per second).

RAID levels

RAID 0 uses disk striping to distribute data evenly across all the disks in the array. There is no redundancy or duplication of any data, therefore data-security in minimized. The upside of this scheme is that it provides very high data transfer, and high I/0 rates for both read and write. Supplemented with a well implemented data integrity scheme, RAID 0 can significantly enhance performance in most general applications.

RAID 3 subdivides and distributes each data sector across all data disks, with redundant information stored on a dedicated parity disk. Data can be accessed on different drives concurrently, thereby offering very high data transfer rates, but no gains in I/O rates. RAID 3 is a great performance enhancer for large blocks of data such as video and multimedia type applications.

RAID 5 distributes data sectors as with disk striping, with additional independently computed redundant information. It significantly enhances data transfers and I/O reads, but penalizes writes. RAID 5 offers great performance in most business and database applications.

Other RAID levels that have been proposed to enhance performance, but they typically rely on complex and costly proprietary structures which have not gained broad market and industry acceptance.

Tools Available in RAID Systems to Enhance Performance
A well constructed caching algorithm is essential to a high performance RAID system. This article will cover methods to get the most out of caching.

Intelligent caching
Data transfers can be greatly improved by using adaptive techniques to allocate the optimal amount of cache memory to various read and write command blocks. Varying these segment block sizes will improve cache performance.

Caching and look-ahead
A look-ahead scheme ensures that when the host CPU requests data, the RAID caching algorithm provides the requested data. Look-ahead goes one step further and reads sequential data immediately following the request. That sequential data is written to a cache block on the array controller. If the host CPU requests that subsequent data, it can retrieve it from the cache nearly 1000 times faster than it normally would.

Studies have shown that 55 to 70 percent of all disk requests are sequential. As a result, well designed cache look-ahead schemes keep track of the sequential nature of data and continuously fill the cache with new data based on sequential patterns encountered in user storage activity.

Look-ahead caching eliminates the seek time and latency associated with non-cache transactions and keeps track of the type of drive activity (sequential versus non sequential) as well as the length of time a block of data resides in cache without being requested (FIFO implementation).

Effective array level caches typically range from 8 MB to 128 MB with the cost/performance ratio being optimized for most broad based applications in the 16 to 32 MB range.

It is worthy to note that the disk array's cache complements and greatly enhances the less sophisticated system level cache and smaller drive level caches found in most current hard disk drives.

Caching caveats
Caching offers significant performance gains in a disk array architecture. However, in order to maintain the RAID system's data integrity and fault tolerance requirements, it is critical that the data present in the cache during a system failure be gracefully recoverable.

This can be done by incorporating a UPS (uninterruptible power supply) in the RAID system and providing adequate firmware to flush the cache and carry through the rebuild process without data loss.

Multi-tasking environments
In a multi-tasking environment such as UNIX, where a disk array typically services multiple CPU operations, the array must divide the available time among all operations, even though each might be requesting data sequentially from the disk.

Without an adequate caching implementation, the read/write heads in the array will typically seek from one location to another in order to service multiple data requests. With caching, the number of seeks required will be significantly reduced because of segmented cache. After the first seek and read has been performed for each cache, the disk array's cache on board typically takes over and transfers the data directly from the various segments of cache memory.

Tagged Command Queuing

Tagged command queuing (TCQ) allows the host to send multiple commands (from 8 to 64 depending on the implementation) to the disk array for processing. These commands are then tagged and can be reordered in the queue to reduce the time it takes for drives in the array to 1) access specific blocks of data (minimize latency); 2) optimize the use of sequential data; and, 3) increase the number of cache hits, and optimize the execution of a command stream. TCQ is most beneficial in environments which support that feature (such as Unix).

Handling large blocks of data

Another bottleneck in disk array performance has to do with the transmission of large quantities of data. Typical examples are emerging multimedia and related full motion video and video-on-demand applications, as well as traditional multi-tasking and LAN based database and general server applications.

An intelligent way to address these challenges is to eliminate the REQ/INIT/ACK CPU intensive steps usually present between blocks of data by implementing intelligent DMA (direct memory access) techniques.

Similarly, it is possible to reduce the number of interrupts handled during the processing of an I/O request, freeing valuable CPU time. The result is faster throughput to the system, especially for large blocks of data, such as those described above.

SCSI as a bottleneck

The SCSI standard alone can be a potential bottleneck to the RAID disk array (SCSI Fast is only 10 MB/sec. versus SCSI-2 Fast and Wide at 20 MB/sec.) As previously explained, the best disk array performance can usually be obtained in hardware-based, host independent implementations.

Most of the methods we have discussed offer significant enhancements and overcome the performance penalties inherent to the other two aspects of RAID: better data integrity and lower cost. With these tools, a hardware-based subsystem can come very close to the sustained throughput limit of SCSI-2 Fast and Wide (ie 20 MB/sec.).

The future

SCSI is becoming a limiting factor in our performance requirements. Full-motion video, multimedia, and increasingly complex business applications being right-sized from the glass room to personal computers and workstations--will need significantly more power than SCSI can harness.

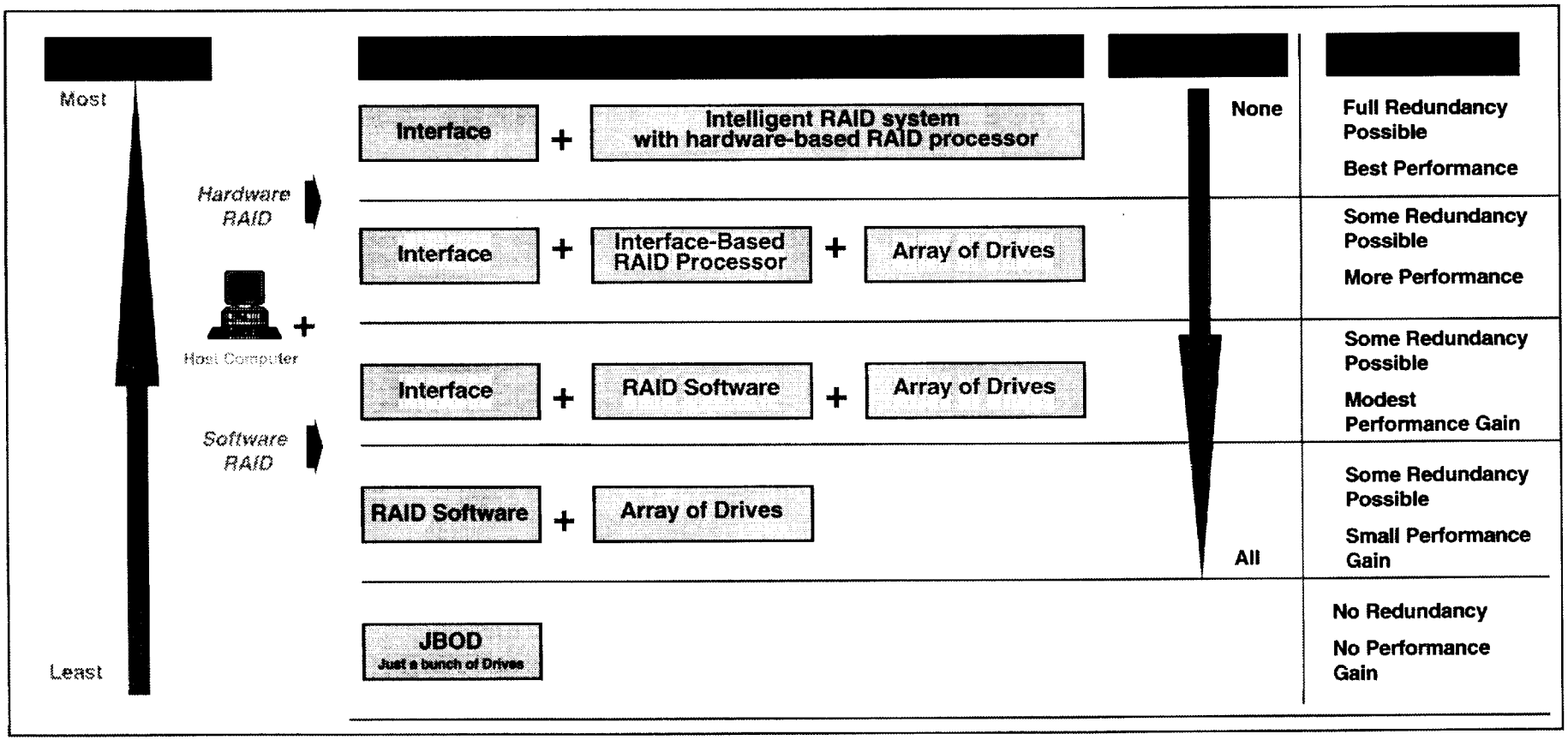
Intel's P-6 and Motorola's future PowerPC chips will provide the needed processor power. RAID can and will provide high bandwidth storage. What is missing is a faster, more flexible and cost effective interface standard.

Faster Interface Standards
Fiber channel's Gigabit/sec. throughput, under any of its current four or five proposed implementations, shows every sign of fulfilling this promise in the next few years.

Fiber will tie in processor and RAID storage under one high power interface standard and provide us with the high-speed highway needed to support our exponentially growing information needs.

RAID technology is capable of offering the high performance needed to access and process large amount of information, when properly implemented. There are many factors that contribute to RAID performance. The key is to assess the specific storage and application requirements, and select the most appropriate RAID scheme. Once this is done, the RAID system can offer significant performance gains over JBOD (Just A Bunch of Drives) by using the tools such as the ones discussed here.

Hardware vs. Software RAID



MEGADRIVE.
The Power of Critical Mass Storage

Document Image Archive Transfer from DOS to UNIX

Susan E. Hauser, Michael J. Gill, George R. Thoma
Lister Hill National Center for Biomedical Communications
National Library of Medicine
Bethesda, Maryland 20894
hauser@nlm.nih.gov
Tel: 301-496-4496
Fax: 301-402-0341

Abstract

An R&D division of the National Library of Medicine has developed a prototype system for automated document image delivery as an adjunct to the labor-intensive manual interlibrary loan service of the library. The document image archive is implemented by a PC controlled bank of optical disk drives which use 12" WORM platters containing bitmapped images of over 200,000 pages of medical journals. Following three years of routine operation which resulted in serving patrons with articles both by mail and fax, an effort is underway to relocate the storage environment from the DOS-based system to a UNIX-based jukebox whose magneto-optical erasable 5 1/4" platters hold the images. This paper describes the deficiencies of the current storage system, the design issues of modifying several modules in the system, the alternatives proposed and the tradeoffs involved.

Background

The Lister Hill National Center for Biomedical Communications, an R&D division of the National Library of Medicine, has developed a prototype system for the automated retrieval and delivery of document images as an adjunct to the manual interlibrary loan service of the library. The system is integrated with the library's existing interlibrary loan system and is transparent to the requester. Since April of 1991, the system has retrieved from optical disk storage and delivered to patrons the images of over 27,000 articles by fax and mail. While the current operation has been scaled down, the system continues to deliver about 450 articles per month and about 550 page images are added to the image archive per month.

The prototype system [1] consists of several DOS-based workstations connected to a LAN and supported by a Netware 3.11 file server. The workstation functions include document capture, image quality control, document tagging, document image archive, communications gateway and document delivery. Most of the software to support these functions was developed in house. The file server serves as a temporary image store until captured images have passed quality control, and it stores the several databases that the system uses to track images and requests.

The image archive is implemented by a bank of four 12" WORM optical disk drives connected via SCSI-1 to a PC. The vendor-supplied software that mediates the operation of the drives configures the workstation as an optical disk server that communicates with other PCs on the network via the IPX protocol used by Netware. Thus, by logging into the optical disk server, other PCs on the network can write and read image files directly to and from the optical platters. All of the files on the optical disk server appear to the PC to be located at a single drive letter. The archive currently holds over 200,000 image files on 15 12" platters, for a total archive of approximately 15 Gigabytes. Because there are more active platters than there are drives, software has been written to effect a "human jukebox" for manual platter exchange.

Optical Disk Server Problems

The four WORM drives of the archive workstation range from 2 to 9 years old and all have been in continuous operation since delivery. These aging drives are no longer supported by the manufacturer. Although maintenance, troubleshooting and some repair and replacement are performed by in-house technicians, parts and high-level repair must be obtained from a third party. Compatible and reliable media are also becoming difficult to obtain. In addition, the frequent manual exchange of platters is taking its toll on both drives and media.

At the time that the optical disk server software was purchased, there were few commercial options for network access to 12" WORM drives from PCs. The optical disk server software was selected because it met our minimum requirements for remote access to optical platters and included a small set of C-callable functions that our in-house programs could use to obtain information about the status of the drives and platters. However, this DOS-based software has not proven to be robust when handling multiple requests and error recovery is generally inadequate, requiring frequent intervention by the technical staff. The original manufacturer of the optical disk server software sold their license to a company overseas with no support staff in this country. The new company has not addressed the reliability and error recovery issues, and their new version of the software cannot write to platters written to by earlier versions.

The optical disk server continues to function adequately at its current low usage level, but at the cost of several man-hours of labor per week. There is also the threat of irreparable breakdown of one or more of the aging, irreplaceable optical disk drives. For these reasons, we are exploring the transfer of the image archive to a more reliable, flexible optical disk server employing current technology.

Rationale

The degree to which images in the archive are accessed is a function of their age, the probability of more recently published documents being accessed being higher than for older documents. One approach to solving the archive problem is to permanently retire disks containing older documents, and have only three or four platters permanently placed in the drives. These would then contain those documents that have the highest probability of being requested, thus reducing wear on drives and media from manual platter exchange. This approach might extend the life of the system for a short time but is not likely to significantly reduce the amount of staff labor needed to maintain the system.

There are good reasons to preserve the entire image archive. These images represent a large investment in equipment and labor. Although the development and operation of the prototype system largely answered the original research questions regarding cost, performance and image quality, the database of document images has potential value for future research. The archive could be used in projects addressing document image processing, image compression, file format conversion, image transfer, image access, or mass storage. It could also prove useful in testing components of improved document image delivery systems.

For these reasons, an effort has begun to relocate the entire image database from the DOS-based system to another system of optical media in which media are automatically exchanged when necessary and multiple network communications are handled reliably.

New Image Storage Requirements

In the new image store, all active images should be accessible from the current document delivery system without manual intervention. To be available to the widest number of future projects, the image database should be accessible from UNIX platforms, which normally communicate via TCP/IP, as well as from the many Netware-based PCs in the division. Internet access to the database would make it available to collaborators at other sites as well. These requirements are met by the division's HP 100 optical disk jukebox [2] in conjunction with the Netware NFS Gateway software [3]. The four-drive jukebox has a current near-line capacity of 93 Gigabytes, expandable to 186 Gigabytes. It is connected to a Sun 670MP and controlled by software from Alphatronix. Each platter side appears as a UNIX file system and is directly available to any computer to which it is exported. Netware NFS Gateway software supports NFS mounting of UNIX file systems to Netware servers, where the file system is available to Netware users as a Netware volume.

The jukebox also supports other projects. Should insufficient space be available for the image database, other commercial solutions are appearing. It is expected [4] that expandable network storage products will soon emerge that will connect directly to the network and will offer storage that is independent of the operating system. There is one

optical disk jukebox system available now that connects directly to the network and supports both TCP/IP and IPX/SPX communications [5].

Software Requirements

In an ideal world, the image database could be moved to a new image store with no effect on the operation of the current document delivery system. However, because much of the in-house-developed software is tightly integrated with the current optical disk server software and the operation of the "human jukebox", no simple substitution is possible. Any change in image store will require modifications to several of the modules that comprise the system. Software modification is not a casual matter. Several of these modules are written for a C compiler that is no longer supported, while others are written for an older version of Microsoft's C compiler. All these modules use a no-longer-supported library of routines to interact with the databases that resolve the location of image files corresponding to journal articles.

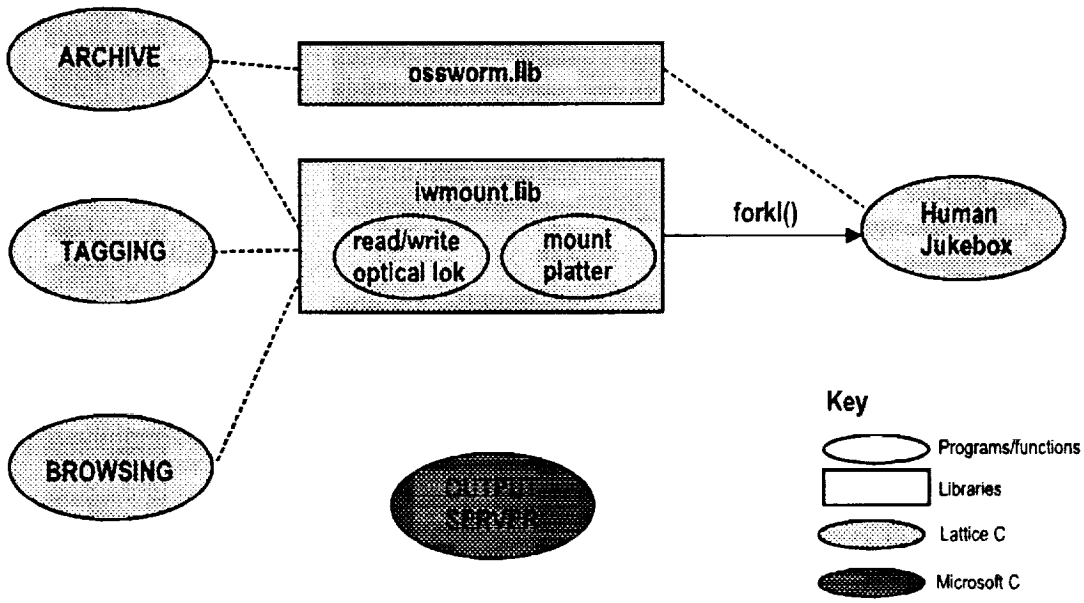


Figure 1. Modules of the current system that access the image store.

Figure 1 illustrates the software modules of the current system that interact with the image store and the libraries that are used to facilitate use of the optical disk server. The archive module moves the page images of a journal issue from the temporary store on the Netware server to permanent store on a WORM platter. For each issue, the tagging module adds operator-supplied data that identifies the page images that correspond to individual articles in the issue. An operator can use the browsing module to match articles with requests

containing ambiguous or insufficient information for the system to automatically select the article. The output server copies page image files corresponding to an article from an optical platter and either faxes the images to the requester or prints the article for delivery by mail. All but the output server interact with an operator.

Ultimately all reads and writes to the optical disk server are straightforward, but modules must first determine if the required platter is in a drive. If it is not, human intervention must be invoked through the module labeled "human jukebox" in the figure. In addition, before archiving a journal issue, the archive module must determine the remaining space on a platter to be certain that there is sufficient space for all page images of the issue. Since a file/platter locking feature is not part of the commercial optical disk server software, all modules use the special optical.lok file to prevent one module from requesting the operator to remove a platter that another module is using. Although three of the modules share a few library functions, as shown in Figure 1, in general each module is responsible for how it accesses files on the optical disk server.

Minimum modifications to the software to accommodate a new image store implemented by an optical disk jukebox will have to remove references to operator intervention and to the functions that obtain information about drive and platter status.

Other issues

File format and image organization: The page images in the current system are compressed using the CCITT Group IV algorithm. Each page is stored as a separate file with no header. All of the page images from one journal issue are stored in one subdirectory. The metadata that describes which page images are associated with each article in the issue are stored in one file in the subdirectory with the images. The subdirectory name is a number, assigned consecutively at the time the issue is archived. Thus, any module using images as they are currently stored must obtain information from the system database files to find the path to a given issue, must be able to interpret the metadata file to find pages for a given article and must have *a priori* knowledge of the image file format. To make the image database not only available to a wider audience, but also self-explanatory, changes in file format and organization will be considered.

Access time: Very fast image retrieval is not critical to the system supporting interlibrary loan since the recipients of the articles are not on line waiting for delivery. Earlier studies of jukebox performance [6] found that the time to retrieve one article is about two seconds when the platter on which the images resides is in a drive. When the platter is not in a drive, the retrieval time becomes a function of the number of other requests waiting for service from the jukebox. In general, retrieval times from the jukebox are sufficiently fast to support the interlibrary loan prototype system. If the image database is used for some other project for which inherent retrieval times from the jukebox are too slow, apparent speed can be improved by designing a prestaging algorithm specifically for the application.

Backup: Once each 12" WORM platter of the prototype system is filled to 95% of its total capacity, a duplicate platter is made using in-house software, and a new platter is formatted for succeeding documents. Should a platter fail, which has happened, the backup can be used in its place. To date, the magneto-optical (MO) media in the jukebox have proven to be reliable. Since it is unlikely that an entire MO platter will fail, it may be sufficient to back up the document files to tape in case individual files should become corrupted. The important issue of effective backup procedures has yet to be fully addressed.

Platter spanning: The software controlling the jukebox supports platter spanning [7]. With spanning, up to 16 platter sides can be merged to become one filesystem of about 4.5 Gigabytes. The filesystem can be exported to the Netware server and made available to PC users as a single Netware volume. The current image database would require more than three such volumes.

Proposed Solutions

In addition to the hardware and software requirements discussed earlier, the design of a new image store should include as goals: a) minimum investment of labor and equipment, and b) maximum flexibility to allow future changes to the image store and future use of the image database. Meeting these goals involves tradeoffs. Minimum investment in labor and equipment implies minimum software modifications to the current document delivery prototype system and the use of current storage devices, namely the HP jukebox. Maximum flexibility to position the database for continued and future use may require new hardware procurements for the image store and extensive changes to the current software. These solutions are discussed below.

Solution 1: For Minimum Cost

To minimize software modifications, the new image database would be organized exactly like the current database, with the images of each issue residing in an arbitrarily named subdirectory, accompanied by a cryptic file containing data used to connect individual pages to the respective articles in the issue, and using a headerless file format for the images themselves. To minimize modifications, the current conglomerate of outdated compilers, databases and user interfaces would be preserved. The result may support the document delivery system for several years, but would provide other applications only awkward access to the images. Furthermore, should the image database require another physical move, to be distributed among several servers, for example, the software would likely have to be modified once again.

Figure 2 illustrates how modules of the document delivery system that access the image store would be organized in a system designed to minimize labor and equipment costs. In

this scheme, a selected subset of the images are moved to platters in the HP optical disk jukebox connected to the Sun host. Sixteen platter sides in the jukebox are spanned to create one 4.5 Gigabyte filesystem that is exported to the Netware server. Only the more heavily requested issues would be copied to the new image store, with one Gigabyte reserved for about 2 years worth of additional documents. The remaining images are permanently retired. Files in the new filesystem are organized exactly as in the current document delivery system. The entire filesystem appears to the document delivery workstations as one Netware volume which is mapped to a single drive letter, just as the current optical disk server is accessed though a single drive letter. Functions in the iwmount.lib library, which were previously used to operate the "human jukebox", are replaced by functions bearing the same name whose only purpose is to return a good status. In this way, the tagging and browsing modules need not be rewritten, but only relinked to the new library. Because the archive module is so tightly integrated with the current optical disk server and includes functions, such as determining available space on a platter, that are not in the iwmount.lib library, it must be rewritten to support the same functionality with respect to the jukebox. The required information can be obtained through functions in the Netware Software Development Kit (SDK). Since the SDK supports Microsoft, but not Lattice compilers, the new archive module is written for Microsoft. The output server module may not need to be rewritten or relinked. Because it is intended to operate automatically, even during periods when an operator is not available, it does not directly access the "human jukebox" functions.

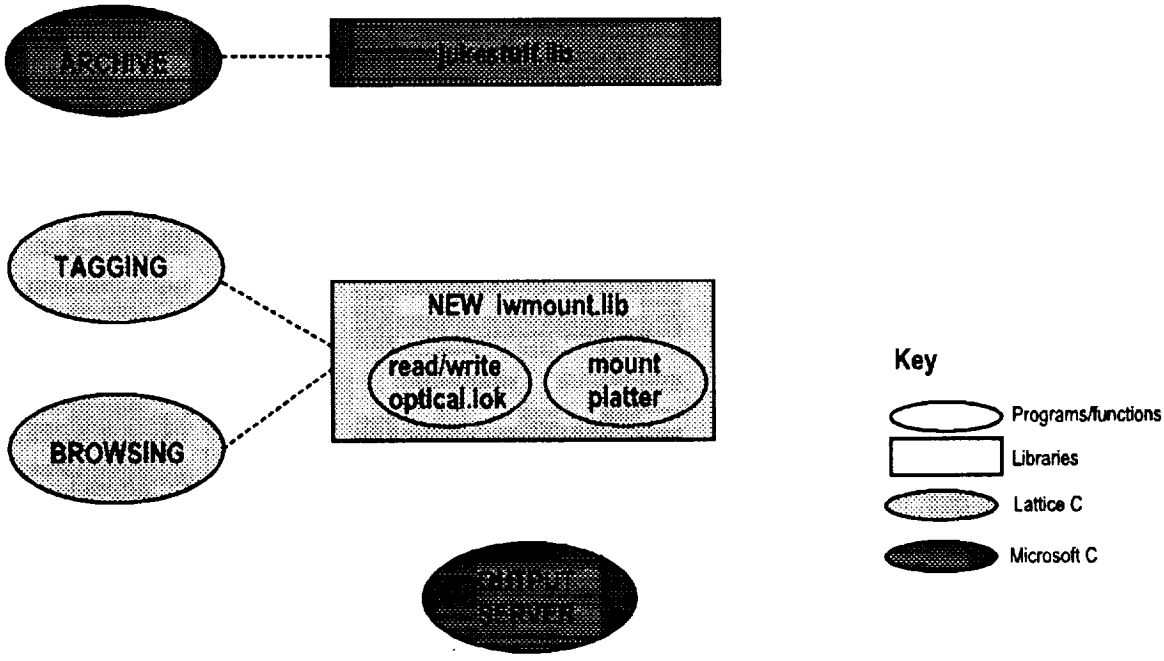


Figure 2. Proposed modules for minimum cost solution.

Solution 2: For Maximum Flexibility

To maximize flexibility and access, the image database would be reorganized for easy management and for intuitive navigation to image files that have a standard file format and header. The path to the subdirectory containing the images for one issue includes a three letter code identifying the journal title and other characters to indicate volume and issue. With each issue there is an easily interpreted text or database file containing data linking page images to articles. All images reside in an optical disk jukebox with sufficient capacity to store the existing database plus at least five years expansion. The jukebox connects directly to the network with software that supports access via both TCP/IP and IPX/SPX. Each platter side appears as one volume to Netware clients and as one filesystem to UNIX clients. All modules of the document delivery system that access the image database would be rewritten to reflect the new organization and location of the image archive. The result would permit easy access to the database by both Netware and UNIX applications. However, the high cost of the new, sophisticated image store and the many person-months of programming effort may need to be justified on programmatic grounds.

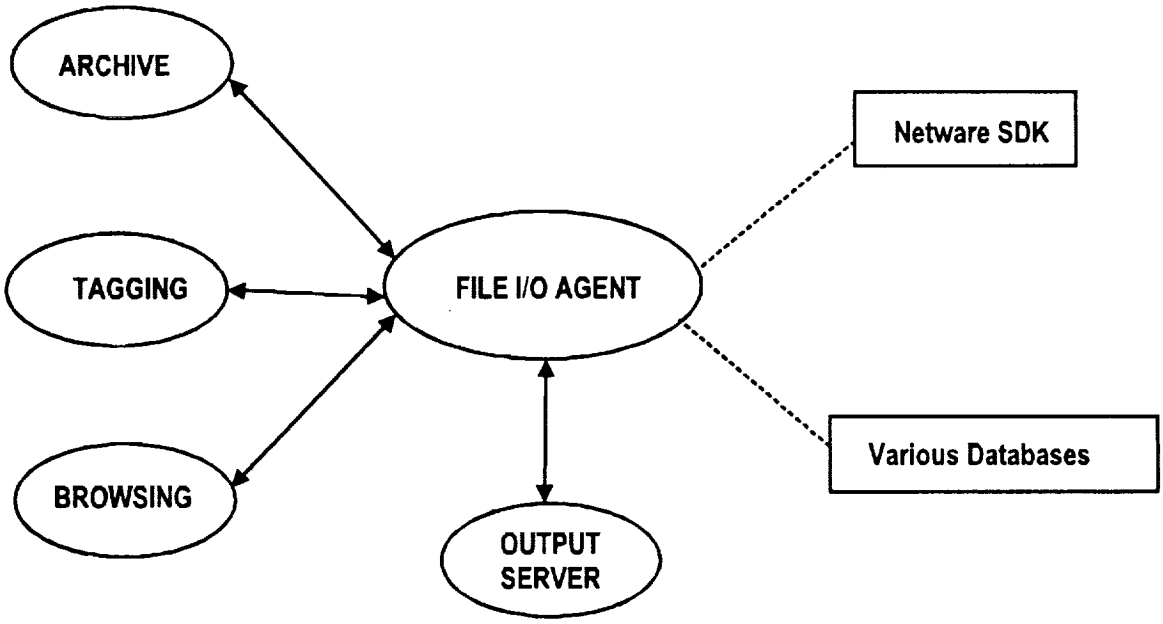


Figure 3. Proposed modules for maximum flexibility solution.

Figure 3 illustrates one concept of how modules of the document delivery system that access the image store would be organized in a system designed to maximize flexibility and access. All modules are rewritten to reflect the new system and file organization. They are no longer individually responsible for understanding image database organization or file location, but invoke a new module for all image file access. The new file I/O "agent" is

responsible for specifying or discovering the location of any file and mediating all reads and writes to the image store. It creates and uses database information to determine the path to a given file and employs the functions in the Netware SDK to obtain information about the volumes containing the files. Should the image database be relocated or distributed among several sites, only the databases used by the file I/O agent would be changed.

Conclusions

Most hardware and software requirements for a new image store are satisfied by the division's HP 100 optical disk jukebox connected to a Sun UNIX platform and accessible from PC Netware clients via the Netware NFS Gateway. Moving the image database to a UNIX platform also immediately increases its exposure to a new set of clients and potential applications. But moving the image database to any new location demands changes in the software of the application for which it was originally created. The difficulty in determining the design of the new image store lies with the conflicting goals of minimizing cost and maximizing flexibility and access. The final decision on the design approach will depend upon the importance to the organization of being able to use the image database in the future.

References

1. "System for Automated Interlibrary Loan (SAIL): System and Operations Description," Internal technical report, Communications Engineering Branch, Lister Hill National Center for Biomedical Communications, National Library of Medicine, Bethesda, MD, November 1992.

2. Hewlett-Packard. Optical Disk Library Systems: Product Family Brief, April 1991.

3. Novell, Inc. NetWare NFS Gateway Supervisor's Guide, 1993.

4. Root R. Data storage in the networks of the '90s. *Inform*, July 1994.

5. Symmetrical Technologies. SPANStor-Opti data sheet, 1994.

6. Hauser SE, Roy G, Thoma GR. Optical disk jukebox performance in multi-user applications. *Proceedings of the 1994 Optical Data Storage Topical Meeting*; Vol. 10: pp.53-55.

7. Alphatronix Inc. Inspire Jukebox: User's Manual for Sun Workstations, June 1993.

High Data Rate Recorder Development at MIT Haystack Observatory

H.F. Hinteregger
MIT Haystack Observatory,
Westford, MA 01886-1299
508-692-4764
hfh@wells.haystack.edu

Abstract

Current operational capabilities of tape recording for Very Long Baseline Interferometry (VLBI) at the Haystack Observatory allow 0.7 terabytes (12 hours at 128 Mb/s) of data to be stored in a 128 in^3 volume. On-going efforts are aimed at full time 1 Gb/s operation with two 36-channel headstacks. Applications for linear digital tape recording, with suitable development of thin-film head arrays, suggest a volume density exceeding 1 TB/in^3 to be achievable in the future.

I. Introduction

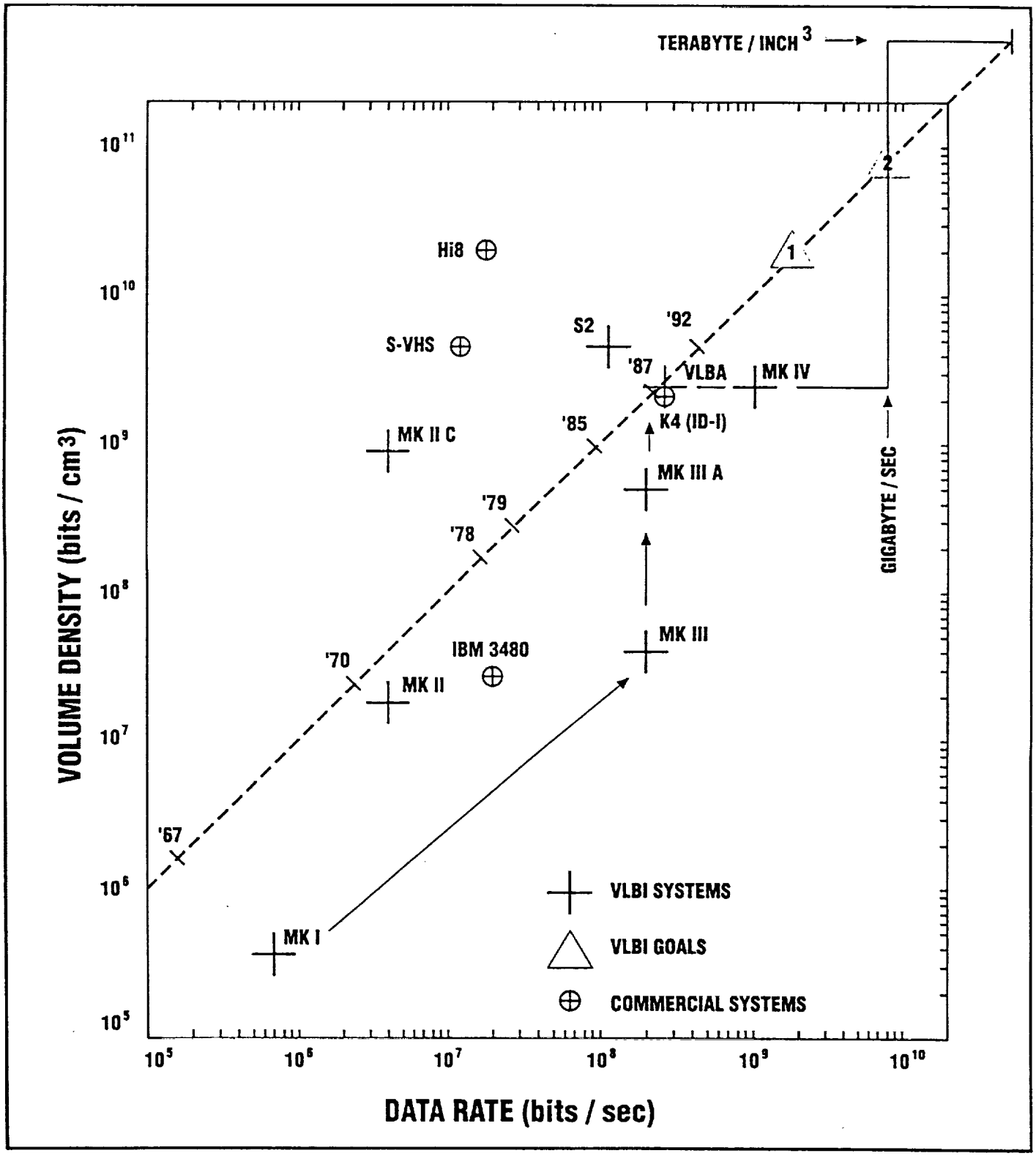
The sensitivity of Very Long Baseline Interferometry [VLBI] observations is proportional to only the square root of the number of bits recorded and processed. This fact, and the lack of affordable alternate means of sustained high-rate data transmission, continues to spur development of ever higher rate and density tape recording systems for VLBI.

The VLBI recorders developed at Haystack for VLBI operation employ the linear [longitudinal] Metrum, formerly Honeywell, Model 96 instrumentation drive, which was selected in 1975 for its excellent tracking repeatability. These drives reliably handle 14" reels of inch-wide tape, now at speeds up to 10 m/s [420 ips], as well as with tape tension as low as 5/8 and as high as 5 N.

VLBI-qualified tapes (3M5345 and SonyD1K) can be shuttled for thousands of passes under these conditions without any noticeable edge-damage [as evidenced by the formation of a bumpy pack]. An understanding of the damage mechanisms and tape-path modifications were developed to make the drive safe for thin tape at high speed. The 15.2 mm thin, 5500 m long, SVHS-similar Co-Fe_2O_3, PET-based tapes, which we have qualified for recent VLBI systems, each store 0.7 terabytes [12 hours at 128 Mb/s] in a 128 in^3 volume.

A total of 576 [38 mm wide] tracks, for example 16 passes with 36 channels at a time, are written at a linear density of 2.25 transitions/mm [2 bits/mm with an 8/9 channel code]. A new version of the recorder (MkIV) operating at 8 m/s is capable of 1 Gb/s operation with two 36-channel headstacks and 2 Gb/s with four headstacks. The growth of the data rate and volume density in VLBI recording is illustrated in Figure 1.

Haystack's narrow-track ferrite read-or-write headstack design, illustrated in Figure 2, has been used since its introduction in 1985. It has been a 'product' manufactured by Metrum since 1989, unfortunately only in the small volumes required by its single VLBI application, which makes it increasingly less affordable. More recent recorder applications all use the same sub-mm resolution positioner [piezoelectric Inchworm actuator and LVDT sensor], one for each headstack. This positioner was developed to support narrow-track serpentine recording. It will need no significant improvement to support much narrower, even sub-mm, trackwidths since actuator resolution is about 4 nm.



In the 25-year history of VLBI, data rate has grown by more than three orders of magnitude to one gigabit/second in the Mark IV system. A further hundred-fold improvement in bandwidth and density is now a technically-realistic goal.

Figure 1

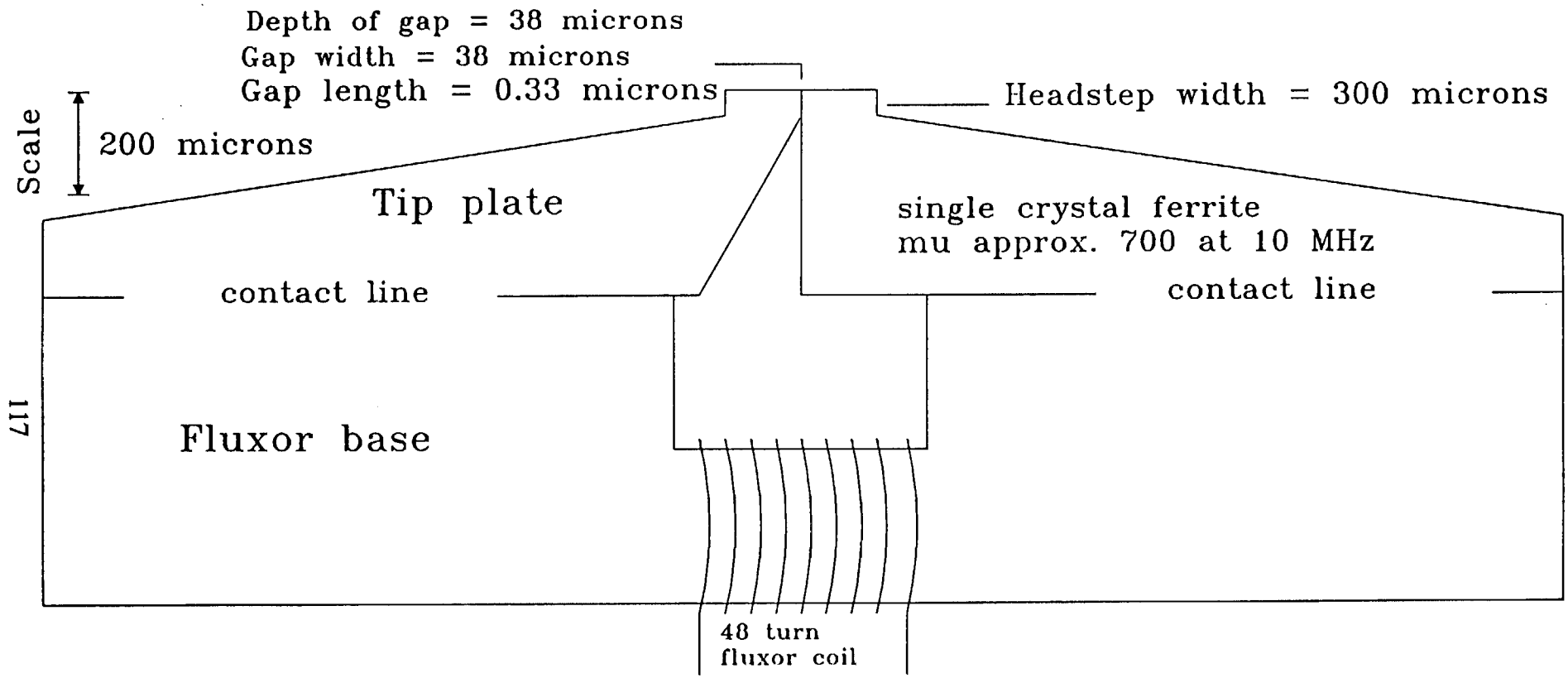

Depth of gap = 38 microns
Gap width = 38 microns
Gap length = 0.33 microns
Headstep width = 300 microns
Scale
200 microns
Tip plate
single crystal ferrite
mu approx. 700 at 10 MHz
contact line
contact line
Fluxor base
117
48 turn
fluxor coil

Figure 2

II. Goals for Future Developments

The more recent version of our VLBI tape recorder developed for the NRAO VLBA system is designed to operate full-time at 128 Mb/s at 2 m/s. With a 10-station array, the VLBA collects 14 TB/day and could process 56 TB/day with its 20-station 4 m/s correlator. A new correlator (MkIV) under development at Haystack will be configurable to keep up with 4 Gb/s per station; another iteration could easily raise that to 4 GB/s. Processing hardware will therefore soon cease to be the chief limit on VLBI bandwidth, hence sensitivity.

Thus there is a need to provide affordable full-time VLBI recording soon at Gb/s, and not much later at GB/s, rates. The first goal, achieving Gb/s rates, requires only new thin-film head-arrays. The second goal, achieving GB/s rates, requires new tape as well. Both these goals are discussed below.

The highly repeatable tracking of the Model 96 drive has been modeled and is now well understood. It is good enough to permit trackwidth reduction from 38 to 7-10 mm without, and probably less than 2 mm with, possibly only minor modifications. Short-term nonrepeatability is about 1 mm p/p and is due to wobble of the capstan, which is in close proximity to the head. The 'perfect' edge-guide [bearing point] in the slanted-wall vacuum pocket is much further away. The desired modifications under development should have the effect of bringing head and edge-guide very much closer together. They must also increase the along-tape distance from head to capstan, and/or isolate it with another sliding edge-guide between head and capstan.

The principle of tilted-wall vacuum-pocket edge-guiding can be applied to vacuum-less guides such as tilted posts or flexible conical-foil air-bearings. Long-term nonrepeatability and sensitivity to pack imperfections which produce a reel once-around 'bump' signature is also eliminated by bringing head and edge guide together. Much more robust, pack-imperfection-tolerant passive tracking should result.

a. Thin-Film Head-Arrays

Haystack is seeking support to collaborate with Seagate Tape Technology Inc. to further develop thin-film head-arrays for linear multichannel drives. Thin-film head-arrays, with inductive write (IW) and magnetoresistive (MR) read elements, are key to achieving both high data rate and density at the lowest cost. Sensitive MR heads will allow a 4-fold increase in track density with the present tapes. A read-width of 9 mm and write width of 11 mm can therefore now be targeted. More conservative specifications for VLBI thin-film head-arrays proposed in 1993 (not funded) are given in Table 1.

Current processes for thin-film IW-MR disk heads allow increasing channel density from 36 to at least 144 per inch. Channel densities of 1000 per inch or so are not far off for MR and 1-2 turn IW heads. With a mature high-yield process, if chips are of comparable size, the cost of a dense head-array should eventually become comparable to that of a single-channel disk slider. This will happen only when and if a comparably large linear tape head market develops. Commonality at the wafer level with a high volume PC/consumer product is therefore highly desirable. The manufacturability of dense arrays is the central issue for this kind of product.

b. *The Promise of Advanced Tapes*

Advanced high-SNR, high-resolution, super-smooth, ultrathin (3-7 mm) metal-evaporated, metal-particle, and barium ferrite tapes promise further 4-fold density increases in each dimension and continue to be evaluated. TB/in^3 volume density is technically within reach with products like WVHS and Hi8Master tape.

Efforts to guarantee more nearly ideally robust edges for ultrathin tape must be pursued with the

Specifications for VLBI Head Array Prototypes

Read-Only Array	
Technology Thin Film	MR
Track Width	19 Microns
Head Pitch	349.25 microns
Number of Heads in Array	72
Equivalent Spacing	<0.05 microns
Write/Read-Verify Array	
Technology Thin Film	Inductive
Track Width	22 microns
Head Pitch	349.25 microns
Number of Heads in Array	72
Equivalent Spacing	<0.1 microns
For Both Arrays	
Minimum Record Density	56,000 kfci
Tape Speeds Up To	8 m/s (320 IPS)
Media	3M 5345 (16-micron thick) Similar to S-VHS)

Table 1

collaboration of manufacturers. A degree of rounding of the corners of order 10% of thickness and comparably small edge asperities are needed to guarantee that plastic deformation of these asperities does not lead to edge-thickening which results in a bumpy pack.

Since the stiffness of tape goes as thickness cubed there may be a more fundamental problem with winding much thinner than 15 mm tapes on large self-packing reels. These were invented for VLBI in 1978 and the design was improved in 1994 explicitly to suppress scatter-wind so as to make the tape directly shippable. Self-packing reels intentionally use close spacing and curvature of the flanges to guide the tape into a smooth-faced pack. This does not damage the edge even under extreme high-speed [8 m/s], high-tension [4.4 N], long-term [1000 pass] running conditions. Both 3M5345 and SonyD1K typically pass such 'torture' tests with no evidence of damage [a bumpy pack when wound at 2 m/s]. With 9-mm tape samples similar to 3M5354 the tape edge tends occasionally to hang up slightly on the flange and voids are formed because subsequent turns yield too easily. Even without this problem, unreasonable thickness uniformity requirements may be placed on ultrathin tape wound into a 14" diameter pack. Ultrathin tapes are likely to be much better suited for formats where the pack diameter is kept under about 3" [VHS, 3480, DLT, QIC, etc.].

III. Longer Term Potential of Tape Recording Applications

Tape recording is and for the foreseeable future will remain by far the densest and most economical form of data storage. Applications that must sustain high data rates in the 0.1 to 10 Gb/s neighborhood or that require convenient access to enormous [of order 10**15 bytes (petabyte)] data libraries will continue to rely on tape.

The overwhelming volume density and capacity advantage of tape is due entirely to its high relative thickness-density. The areal density of tape recording can keep up with the advances in disk recording. Though track densities on tape [especially in linear systems] have traditionally been lower than on disk, simple passive edge-guides can provide essentially perfectly repeatable tracking; active track-following should not be needed to at least 10,000 tpi.

The two fundamental advantages of linear [compared to helical-scan] tape drives are: (1) its mechanical simplicity, and (2) its ability to support many parallel channels, hence very high aggregate data rates, without added complexity.

A low-cost high-rate linear digital VCR is possible in the future. Given the 4 in^3 capacity of a VHS cassette or 3480 cartridge for example, a 250 Mb/s machine could operate for two hours at a volume density of only 1/16 terabyte/in^3, a density that can be achieved with some of today's tape and head components. The 'effective density' of the Hi8 analog consumer video tape recording format in long play mode is about 1/20 TB/in^3 for example. This density is already so high that the use of so-called data-compression [redundancy reduction, especially the lossy varieties used for video] for tape-storage seems counter-productive, an unnecessary added special-purpose complexity. Rather, with more than an order of magnitude higher TB/in^3 density on the horizon, the highest-volume market for such high capacity storage should be cultivated. There should be no doubt that the all-purpose high-rate capability of the linear recorder can be made attractive to the consumer/PCuser.

The key prerequisite for this development is the availability of mass-production processes for (1) dense thin-film narrow-track head-arrays as discussed above, and (2) simple, low-power, multi-channel ICs.

The thin-film head and integrated channel technologies are ones in which the U.S. maintains a

strong leadership. These provide a major opportunity for the U.S. storage industry to take the initiative to reenter the consumer market with native drive, head, channel, and tape component technologies, each of which has substantial technical and/or manufacturability advantages.

Author's VLBI-Related Work Background

Hans F. Hinteregger received B.A. and Ph.D. degrees in physics from MIT in 1964 and 1972 respectively and has been at Haystack Observatory since 1967 at the time of the first very long baseline interferometry (VLBI) experiments. He pioneered the geodetic/astrometric applications of VLBI by introducing means to accurately measure group delay (by coherently sampling a wide spanned bandwidth). Since 1975 his work within the Haystack VLBI Group has focused on the development of the extreme wideband digital tape systems required by VLBI. The latest version of this system (Mark IV) has demonstrated 1 Gb/s operation with sixty-four 16 Mb/s channels [using two headstacks of the author's 1985 design].

Petabyte Mass Memory System Using the Newell Opticel*

Chester W. Newell
Primelink Technologies Inc.
#208, Advanced Technology Centre
9650 - 20 Avenue
Edmonton, Alberta, Canada T6N 1G1
Phone: +1-403-440-0111; FAX: +1-403-440-0110
Email: evergreen@maugham.atc.edmonton.ab.ca

Abstract

A random access system is proposed for digital storage and retrieval of up to a Petabyte of user data. The system is comprised of stacked memory modules using laser heads writing to an optical medium, in a new shirt-pocket-sized optical storage device called the Opticel*. The Opticel described is a completely sealed "black box" in which an optical medium is accelerated and driven at very high rates to accommodate the desired transfer rates, yet in such a manner that wear is virtually eliminated. It essentially emulates a disk, but with storage area up to several orders of magnitude higher.

Access time to the first bit can range from a few milliseconds to a fraction of a second, with time to the last bit within a fraction of a second to a few seconds. The actual times are dependent on the capacity of each Opticel, which ranges from 72 Gigabytes to 1.35 Terabytes. Data transfer rate is limited strictly by the head and electronics, and is 15 Megabits per second in the first version.

Independent parallel write/read access to each Opticel is provided using dedicated drives and heads. A Petabyte based on the present Opticel and drive design would occupy 120 cubic feet on a footprint of 45 square feet; with further development, it could occupy as little as 9 cubic feet.

Introduction

The Compact Disc digital audio player dramatically illustrated the classic criteria for a successful new market entry: it produced a significantly higher-quality product in a smaller size, with important new features yet at a competitive price, in a marketplace which had stagnated.

The Newell Opticel promises the equivalent for video bandwidths: a digital recorder-player module which, by meeting the same criteria, would bring about an even more far-reaching revolution in the computer and home-entertainment markets.

---

*Trademark

The Opticel drive combines novel cartridge concepts with three other discrete components: optical heads, optical media, and signal and control electronics. It is packaged in a standard 3-1/2-inch form factor, one-half the volume of the highest density 5-1/4-inch CD ROM drive, with over ten times its capacity, yet with access times comparable to single-disk systems, and transfer rates limited only by the digital electronics.

There are many development efforts under way seeking such an objective. Those known all suffer critical shortcomings: disk-based systems are bulky and have limited transfer rates; tape-based systems have poor access times. The Opticel retains the advantages of both with none of the fundamental disadvantages.

The Optical Head

Primelink is developing two heads:

(a) An IR-laser head, on a conventional 2-axis fine actuator movement (tracking and focus) and a stepper-motor/ball-screw carriage assembly for coarse tracking, is being designed.

(b) A red-laser-based, integrated, solid-state, multi-channel head system is being researched, with no tracking movements (i.e. no moving parts).

The first head will go to production in mid-1995; the second is scheduled for release in a second-generation product (probably around the year 2000).

The Optical Recording Media

Primelink has working agreements with two U.S. companies for archival or WORM optical tape, and for non-archival, erasable tape. Both are compatible with the optical head lasers.

The Data Encoding/Decoding Method

Background - The encode/decode technique normally used in today's WORM disks is to ablate a pit using an IR laser. The data are encoded by phase-modulating the position of the pit-edge along the track.

If the wavelength of the laser is λ, the numerical aperture of the lens N, and ablation is set to occur at the 50% level on the Gaussian energy curve, then the pit diameter becomes

$$d_{pit} = \frac{\lambda}{2N} \qquad (1)$$

A commercially available head[1] employs, for example, λ = 785 nm, NA = .53. This gives a spot size of 0.74 micron. To allow for inter-track grooves and tracking servo error, a track pitch of 1.6 micron is employed. This yields a raw areal density of

$$D_{raw} = \frac{10^6}{0.74 \times 1.6}$$

$$= 0.84 \times 10^6 \text{ cells/mm}^2 \quad (2)$$

A 5-1/4-inch CD-ROM provides 8×10^3 square millimeters of usable recording area. The capacity of the ROM is thus

$$C_{raw} = 0.84 \times 8 \times 10^9$$

$$= 6.72 \times 10^9 \text{ cells} \quad (3)$$

For digital data, run-length-limited (RLL) encoding can be used, giving 1.5 bits/cell.

Formatting and error correction to a bit-error rate (BER) of a 1×10^{-15} requires an overhead of about 50%. The number of bits/cell is thus

$$N = 1.5 \times 0.5$$

$$= 0.75 \text{ bits/cell} \quad (4)$$

The net user capacity is thus

$$C_{user} = \frac{6.72 \times 10^9 \times 0.75}{8 \times 10^6}$$

$$= 630 \text{ MB} \quad (5)$$

High Density Encoding

Two state-of-the-art techniques have been evaluated for the encoding/decoding circuitry:

(a) Light Intensity Modulation (LIM) - Asahi[2] has demonstrated that it is feasible to reduce the spot diameter by as much as one-sixth that previously realizable for an IR laser, and the track pitch to 0.87 micron, thus theoretically increasing the raw storage capacity of a 5-1/4-inch CD-ROM to as much as 10 Gigabytes. This was accomplished by modulating the light intensity of the laser, using feedback to allow operation higher up on the Gaussian curve. To achieve this density in

practice, however, would require a shorter-wavelength reading laser and extremely high mechanical tolerances.

Solid-state blue lasers are under development by several companies;[3] however, they are not expected to be commercially available for several years. However, green lasers are becoming available and should be able to resolve cell lengths in the order of 0.47 micron. This is also more practical mechanically.

Assuming a green laser, and assuming the bits/cell of equation (4), the user bit length using LIM would be:

$$d_{LIM} = \frac{0.47}{0.75}$$

$$= 0.63\,\mu \qquad (6)$$

Using Asahi's track spacing of 0.87 microns, this would give an areal density of

$$D = \frac{1}{.63 \times 0.87}$$

$$= 1.8\ \text{Mb/mm}^2 \qquad (7)$$

(b) Mark Edge Recording - SOCS Research[4], Los Gatos, California, and Sony[5] have demonstrated the ability to write three bits on both the leading and trailing edges of the cell (see Fig. 1). Primelink has taken a license under this patent. Using an IR laser, Primelink has demonstrated:

- a track width of 0.87 microns
- a cell with 6 bits on 1.67-micron centers gives an average bit length of

$$D_{min} = \frac{1.67}{6}$$

$$= 0.28\mu \qquad (8)$$

Using the Sample Servo method for tracking in place of grooves, a track pitch of 1.2 microns was achieved. This gives a raw areal density of

$$D_{raw} = \frac{10^6}{0.28 \times 1.2}$$

$$= 3.0\ \text{Mb/mm}^2 \qquad (9)$$

The Cartridge/Drive System

As described above, the raw areal densities achievable using production heads and state-of-the-art encoding techniques on 5-1/4-inch disks, have been demonstrated to be between 1.5 to 3.0 Megabits per square millimeter.

Primelink has developed a digital video optical recorder using the Mark Edge Encoding method on phase change (reversible) and ablative (archival) media. To provide "high fidelity" video graphics to the PC market, it was assumed that at least VGA resolutions of 640 x 512 pixels/frame are needed. To upgrade NTSC and VGA to SVGA resolution and provide for HDTV quality for ultimate home entertainment systems, two proprietary DSP algorithms were used to quadruple the pixels to 1280 x 1024. 8 bits/pel were used for luminance, 4 bits/pel for each of the chrominance pixels. From subjective tests using a simple, proprietary, lossy compression algorithm, it was determined that compression ratios of 10:1 would yield decompressed images subjectively virtually indistinguishable from the original.

Using the above parameters, the required user capacity/frame during transmission and storage is

$$C_{user} / \text{frame} = \frac{640 \times 512 \, (8 + 4 + 4)}{10}$$

$$= 0.52 \text{ Mb/frame} \qquad (10)$$

For full motion, 30 frames/sec. are required. The average bit transfer rate is thus:

$$\text{BTR} = 0.52 \times 10^6 \times 30$$

$$= 15.7 \text{ Mb/s} \qquad (11)$$

$$C_{user}/\text{hr.} = \frac{15.7 \times 10^6 \times 3600}{8 \times 10^9}$$

$$= 7.1 \text{ GB/hr.} \qquad (12)$$

Combining equations (7), (8), and (12), and allowing 50% overhead for error correction and formatting, the recording surface area required, depending on the laser/media required, is:

$$\text{A/hr.} = \frac{7.1 \times 10^9 \times 8}{(3.0 \text{ to } 1.8) \times 10^6 \times 0.5}$$

$$= (3.8 \text{ to } 6.3) \times 10^4 \text{ mm}^2/\text{hr.} \qquad (13)$$

Background

Using 5-1/4-inch CD-ROM disks with a useful area of 1×10^4 square millimeters, from equation (13), the number of disks required would be:

$$N = 3.8 \text{ to } 6.3 \text{ disks/hr.} \quad (14)$$

This is not an attractive solution for digital video, for the following reasons:

1. To be a next-generation (full bandwidth digital) version of SVHS (analog) recorders, the capacity should provide for at least 5 hours. 20 and 40 CD-ROM disks or two disks, requiring a stackloader, 15 to 20 inches in diameter would be needed.

2. Even the 5-1/4-inch drive form factor is too large for lap-top computers and hand-held camcorders. A single standard for universal application would not be possible. This is a requirement for the Primelink system.

For 15.7 megabits per second transfer rate, 50% overhead, from equations (6) and (8), the required beam writing velocity is:

$$V = 15.7 \ (0.28 \text{ to } 0.63) \times 2$$

$$= 10 \text{ to } 20 \text{ m/s} \quad (15)$$

and the total capacity required is:

$$C_{min} = 7.1 \times 5 = 35.5 \text{ GB (use 36 GB)} \quad (16)$$

While this writing velocity is not difficult to achieve with rotating disks and stepped laser-optical heads, disks have been shown above to be impractical for our purposes because of their limited capacity. To achieve the required storage area, especially in a 3-1/2-inch form factor, a tape system became mandatory.

Tape

Known attempts to achieve such optical beam writing velocities using conventional tape drives have all employed scanning devices of one type or another:

(1) Rotating scanners:

Many historical attempts have been made using rotating prisms, lasers on head-wheels, etc. They have invariably failed to reach the market in small, modest-cost systems because of the problem of tracking to the tolerances of laser writing.

(2) Solid-state scanners

LaserTape and others have attempted use of solid-state scanners without commercial success, because of difficulty in achieving the required scan angle in compact-geometry systems, as well as in tracking lateral tape skew.

(3) Linear-motor scanners

CREO is the only company which has achieved substantial sales with an optical tape recorder. Using a linear motor-driven platform on which 16 writing and 32 reading lasers are arrayed, a stationary field of 35-mm. tape is scanned, after which the tape is incremented to the next field. The system is mechanically massive, and sells for about $250,000 CDN. This approach offered no solution for the Primelink project.

The Newell II (NII*) Tape Cartridge [6] can easily achieve the required writing velocities without use of an intermediate scanner, and could have been used for this application. The author demonstrated such a cartridge at Newell Research Corp., using 100 meters of 8-millimeter ICI optical tape on 25-micron basefilm, thus providing 8×10^5 square millimeters of surface area at tape speeds to 1000 inches per second. This would potentially provide raw capacities of

$$C_{NII} = \frac{(1.8 \text{ to } 3) \times 10^6 \times 8 \times 10^5}{8}$$

$$= 180 \text{ to } 300 \text{ GB} \qquad (17)$$

for video recording times of

$$T = \frac{180 \text{ to } 300}{7.1}$$

$$= 25 \text{ to } 42 \text{ hours} \qquad (18)$$

Such a system was an "overkill" for the proposed Primelink project, however, and would be more expensive than desired because the cartridge must be reversed at beginning and end of tape. The pulse-power of the motor-drive system required to reverse the system within a time interval that could be economically buffered for continuous video, would be high.

* Trademark

The Newell Opticel[7]

The low tape consumption required for the Primelink project suggested the use of an endless tape loop in a "scramble bin." Such a loop in concept is much simpler than the NII cartridge, in that no tape drive belts or tape reels are required within the cartridge (see Figs. 2,3).

While endless loop systems are well known in magnetic tape systems, the unique problems of an optical tape system using a loop were not trivial, due to the following factors:

- The loop must revolve at high speeds, thousands of times during each record/play cycle.
- The optical recording surface must not be fogged during the life of the cartridge (hundreds of thousands of passes).
- Due to the sub-micron dimensions of the bit cell, debris must not be allowed to accumulate on recorded surfaces from wear or environmental contamination.
- The film must be coupled to the drive and tensioned precisely in the longitudinal direction (X-axis control).
- The film must be edge-guided with very low edge pressure to avoid edge fatigue (Y-axis control).
- The film must be stable in the axis of the laser optics due to very shallow focal depth of the optics (Z-axis control).

Each of these problems, and the method for dealing with it, is disclosed in several patents pending, available on request under a suitable Confidential Disclosure Agreement.

To achieve five hours of high-definition digital video, 16-millimeter tape was used, and two loop lengths were provided for the two encoding methods:

$$L = \frac{5(3.8 \text{ to } 6.3) \times 10^{-2}}{16 \times 10^{-3}}$$

$$= 12 \text{ to } 20 \text{ m} \qquad (19)$$

To achieve long life, bending stresses in the tape were kept within 35% of elastic limits, and no optical surfaces or tape edges touched any other surface. The Opticel case was hermetically sealed and back-filled with one atmosphere of a suitable dry gas. 12.5-micron tape was used.

The above parameters required Opticel case sizes of:

Height:	18.5 mm
Width:	90 mm
Length:	95 mm and 140 mm

The standard Opticel is on the footprint of a 3-1/3-inch floppy disk.

The drive used exactly the same elements as a WORM disk drive:

- Laser head (with Y-Z axis control)
- Drive motor (brushless)
- Control and DSP electronics (ASICs)

3-1/2-inch standard drive dimensions were employed:

Height:	41.3 mm
Width:	101.6 mm
Length:	152.4 mm

An Interchangeable Opticel Using the NII Principle

To achieve much higher capacities, the same cartridge/drive interface geometry of the Opticel scramble-bin cartridge can be employed in an NII-type cartridge (see Fig. 3). This allows a high-performance drive system to be developed that would be downward compatible in writing and reading the Opticel.

The NII development is not within the scope of the present Primelink project. However, this development would be a logical follow-on to the Newell Opticel.

Summary

- The Opticel drive uses the 3-1/2-inch standard form factor for both standard and "stretch Opticels.
- The time to the first bit is less than 5 milliseconds; to the last bit is 1 second for the standard Opticel, 2 seconds for the "stretch" Opticel
- The parts count in the drive is lower than that in a typical 5-1/4-inch WORM optical disk drive.
- Capacity of each Opticel (formatted data with BER of 1×10^{-15}) is 36 Gigabytes for the standard Opticel, and 72 Gigabytes for the "stretch" Opticel.

Two Proposed Petabyte Systems

Based on the loop Opticel

A "stretch" Opticel with MER encoding using 24 meters of tape gives 72 Gigabytes per Opticel. This will fit in a 6-inch drive, with 1-1/2-inch overhang.

For a Petabyte, number of drive modules needed is:

$$\frac{10^{15}}{72 \times 10^{9}} = 13{,}889$$

In a stack loader (see Fig. 4) use:

64 modules/column
8 columns/drawer
3 drawers/cabinet
9 cabinets

This would give dimensions of:

Height: 2.75 m
Depth: 0.813 m
Width: 0.572 m x 9 = 5.15 m

With additional R & D the size and access time can be reduced by using:

- Smaller cells:

  Light intensity modulation, MER encoding
  Short wavelength laser (green or blue)
  Tighter track servoing

  From Asahi predictions:

  Minimum cell size: 0.2 μ

  Cell spacing: $1.67 \times \frac{0.20}{0.74} = 0.45\mu$

  Track pitch: 0.87μ

Equation (9) becomes:

$$D_{raw}' = \frac{6 \times 10^6}{0.45 \times 0.87}$$

$$= 15 \text{ Mb/mm}^2 \qquad (9')$$

- Thinner tape:

  The limiting factor is the shear strength between coating and basefilm; ie., the capacity is inversely proportional to the basefilm thickness. By using 4-micron basefilm, maximum length in the "stretch" version can be increased, and equation (19) becomes

$$L = 24 \times \frac{12.5}{4}$$

$$= 75 \text{ m.} \qquad (19')$$

- Higher tape velocity:

  Primelink has attained tape velocities of 25 meters per second with no difficulty, giving an access time of 3 seconds to 75 meters.

User capacity/Opticel with loop would thus increase to:

$$C_{user} = \frac{0.5 \times 75 \times 15 \times 16 \times 10^9}{8}$$

$$= 1.125 \text{ TB} \qquad (20)$$

For the Petabyte, number of drive modules would be:

$$\frac{10^3}{1.25} = 889$$

The stack loader would require only two drawers in one cabinet.

<u>Based on the NII Opticel</u>

The NII cartridge with 4 micron tape can store 450 meters of 16-millimeter tape in the standard Opticel case size.

- Again, using proven MER encoding with IR lasers, the user capacity becomes

$$C_{user} = \frac{0.5 \times 450 \times 3 \times 16 \times 10^9}{8}$$

$$= 1.35 \text{ TB} \qquad (21)$$

Maximum access time would be 18 seconds. Number of drive modules for a Petabyte would be

$$\frac{10^3}{1.35} = 741$$

Again, two drawers in one cabinet would be required.

- Using LIM encoding with green lasers, the user capacity could be theoretically increased fivefold to 6.75 Terabytes, reducing the number of modules to 148, thus reducing cabinet dimensions to, say

Height:	1.60 m
Depth:	0.813 m
Width:	0.470 m

Its volume would thus be 9 cubic feet.

REFERENCES

1. Mitsubishi ME3H1.

2. I. Morimoto et al., "Ultrahigh Density Recording Using Overwritable Phase Change Optical Disk," SPIE Proceedings, Feb., 1992.

3. W.J. Kozlovsky et al., "Optical Recording in the Blue Using a Frequency-Doubled Diode Laser," SPIE Proceedings, Feb. 1992.

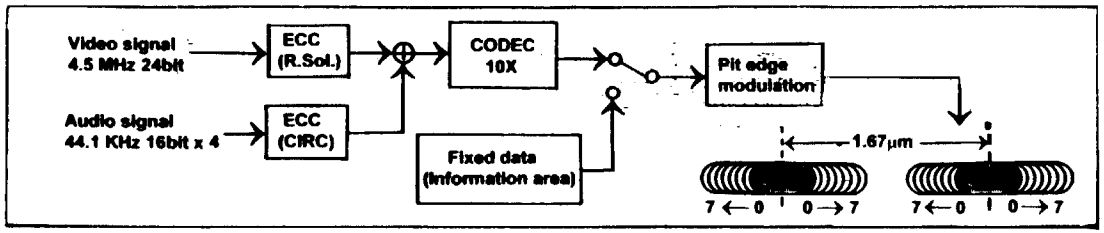
4. U.S. Patent No. 4,736,258, "High Density Storage of Information on a Compact Disc."

5. S. Kobayashi, J. P. de Kock, T. Harigome, H. Yamatsu, H. Ooki, "High Density Optical Disk Recording by Pit Edge Modulation," Corporate Research Laboratories, Sony Corporation.

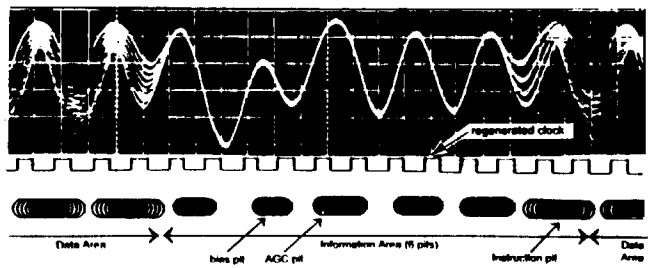
6. U.S. Patent No. 4,172,569, "Tape Transport System With Peripheral Belt Drive."

7. U.S. Patent Pending, Primelink Technologies Inc.

FIG. 1: PRIMELINK MARK EDGE RECORDING FORMAT



Source: Primelink Technologies Inc.



Source: Sony Corp.

FIG. 2: SCRAMBLE BIN OPTICEL

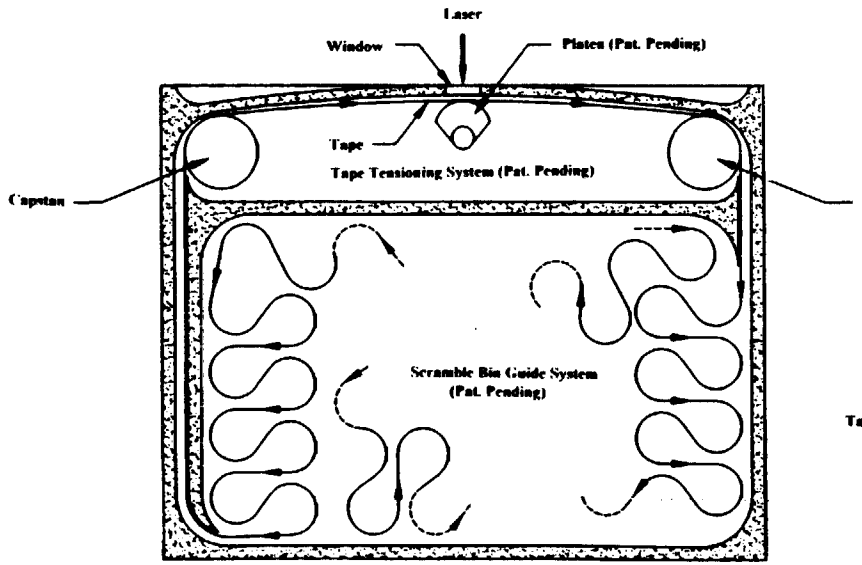


FIG. 3: NII PRINCIPLE OPTICEL (4-HEAD SYSTEM)

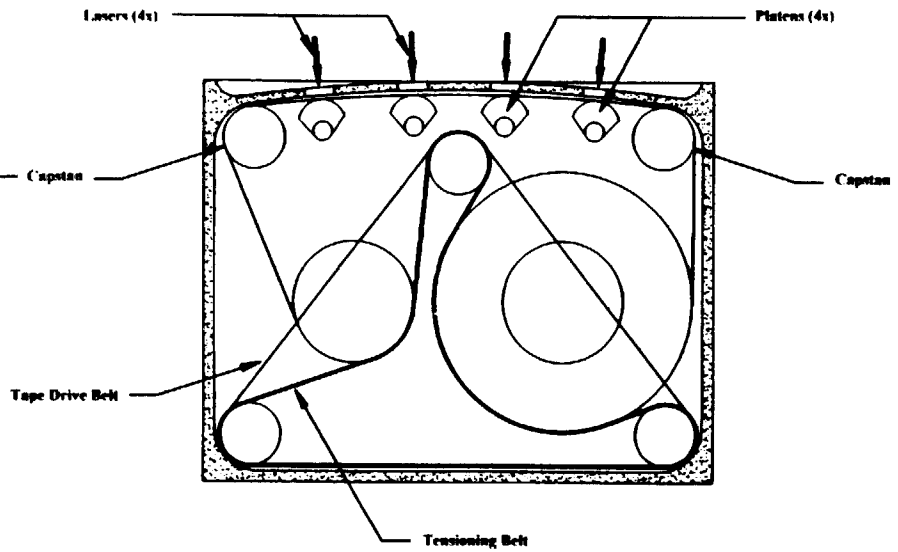
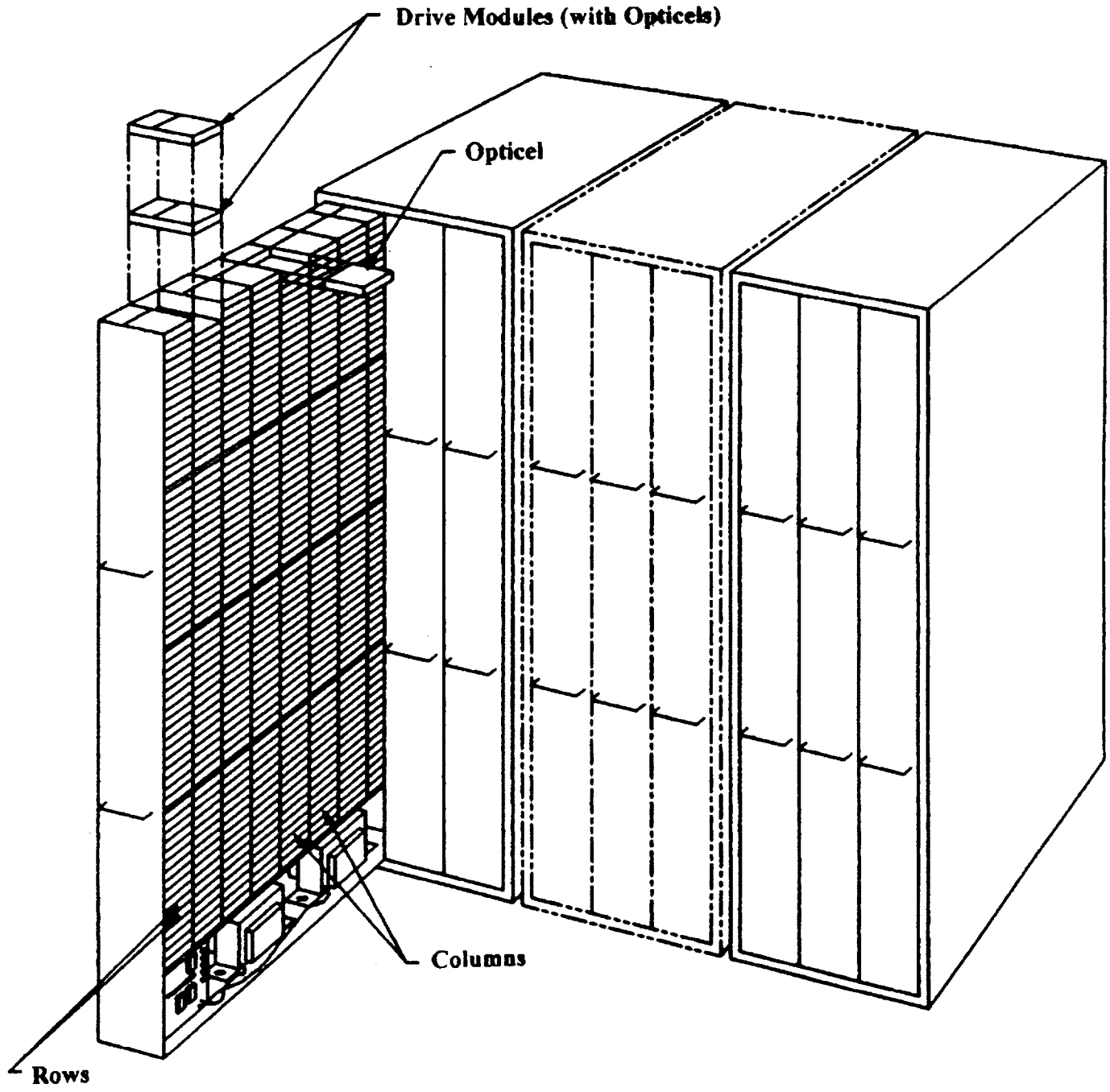


FIG. 4: PETABYTE MASS MEMORY SYSTEM

Proposed by Primelink Technologies Inc.



Integrating UniTree with the Data Migration API

David G. Schrodel
Convex Computer Corporation
3000 Waterview Parkway
PO Box 833851
Richardson Texas 75083-3851
Tel: +1-214-497-4565
Fax: +1-214-497-4500
schrodel@convex.com

1.0 Abstract

The Data Migration Application Programming Interface (DMAPI) has the potential to allow developers of open systems Hierarchical Storage Management (HSM) products to virtualize native file systems without the requirement to make changes to the underlying operating system. This paper describes advantages of virtualizing native file systems in heirarchical storage management systems, the DMAPI at a high level, what the goals are for the interface, and the integration of the Convex UniTree+ HSM with the DMAPI along with some of the benefits derived in the resulting product.

2.0 Introduction

For years developers of Hierarchical Storage Management (HSM) systems have had to choose between integrating their system with the underlying Operating System (OS), or take the more "open systems" route of not requiring any changes to the OS. Integration with the OS allows an HSM to virtualize the native file system. The open systems approach allows easier porting from platform to platform.

The advantage to virtualizing the native file system is that most all applications that reside on top of the file system will continue to operate without requiring changes. Even low level functions like the NFS[1] network protocols continue to function on HSM controlled file systems. In addition, applications that access data in these file systems can expect performance equivalent to file systems not controlled by an HSM for resident files. Examples of products in the market place today that require changes to the OS include the

[1] NFS is a registered trademark of Sun Microsystems, Inc.

EpochServ[2] product from Epoch[3] Systems Inc., and the AMASS[4] product from Advanced Archival Products Inc.

The Epoch Serve product uses "hooks" in the OS via a special device driver and changes to the native file system to gain access to events like file creates, removes, accesses, etc. Although the OS changes have been minimized, subtle changes in the operating system from release to release can introduce problems in the HSM support functions. For example, suppose the OS vendor changes the interface to the kernel memory allocator that the HSM hooks use. Now the HSM must be modified to work with the new allocator, qualified, and released with all of the costs associated with that process.

The AMASS product installs a new file system into the host operating system at the Virtual File System (VFS) layer. Although the VFS interface is fairly well defined industry wide, subtle changes from release to release can cause problems interfacing with the kernel support functions not as well defined. Although changes to the OS can be minimized, kernel integration makes porting of the products more difficult.

UniTree[5] is an example of a product that requires no operating system changes. It sits above the OS and accesses the required OS functions via the POSIX p1003.1 interfaces. It provides access to the data in the archive via the FTP, RCP, and NFS protocols. It does not use native file systems or utilities, but instead provides separate implementations of each. As a result, the product is more easily ported from platform to platform and usually does not require changes as a result of a new OS release. This portability comes at a cost however. UniTree products do not benefit from enhancements provide by the base operating system. New operating system releases tend to have both functionality and stability enhancements included. Because UniTree implements its own file system, the new functionality may not be available. Applications that access a file system by read and write system calls cannot access data stored in the UniTree file system without going through the NFS protocol stack. The access methods provided by UniTree tend to be slower than the same services provided by the underlying operating system. To quantify the performance penalty for this, a single stream access to the local file system in the ConvexOS can reach rates above 45MB/s. UniTree file system access through local host NFS protocols are under 1MB/s.

An alternate approach to the two type of HSM's described above is one that requires no operating system changes yet has access to the native file systems through a set of kernel

---

[2] EpochServ is a registered trademark of Epoch Systems Inc.

[3] Epoch is a registered trademark of Epoch Systems Inc.

[4] AMASS is a trademark of Advanced Archival Products Inc.

[5] UniTree is a trademark of UniTree Software Inc.

supplied "file system hooks". Three examples of HSM's with this attribute is the FileServ Software[6] product from EMASS[7], the Convex Storage Manager (CSM) product from Convex Computer Corp., and NAStore developed by NASA Ames Research Center. All of these products sit on top of the ConvexOS operating system which runs on the C-Series architecture's. ConvexOS exports a set of interfaces that provides functionality similar to what the kernel intrusive HSM's described above export to their user level applications. An application can receive events like read(), write(), trunc(), create(), etc. as well as suspend access to data in a file for non HSM applications. The HSM applications can read data from a file without updating time stamps, punch holes (i.e. free space in a file without changing the apparent size of the file), fill files with data previously migrated out, and re-enable access to the file for non HSM applications. All of these operations can be accomplished completely transparent to normal applications. This allows development of HSM applications like the kernel intrusive ones described above without the drawbacks of having to integrate changes into the base operating system.

The major drawback to the ConvexOS file system hooks are that they are not available on any platform besides the C-Series line of products. What HSM vendors require is access to a standard set of file system hooks across a diverse set of operating systems and platforms. Without this, an HSM vendor can only make a business case to support platforms where they can deliver sufficient volume of product to offset the inherent cost of supporting kernel modifications. This realization was what prompted a set of competing vendors of computer platforms and HSM products, to form an industry consortium known as the Data Migration Interface Group (DMIG).

3.0 DMIG

The DMIG consists of a large group of vendors and a smaller group of active participants in the specification process including: 3M, ACSC, Amdahl, Auspex, Avail Systems, Bull, Convex, E-Mass, Epoch, Hitachi Computer, HP, IBM, Lachman / Legent, Legato, NASA / Ames, Netsor, Novell / USL, OpenVision, SCO, SGI, Sunsoft, and Veritas[8]. This group got together through the 1993 - 1994 time period to produce a specification for a file system interfaces that all parties could agree to support (if not in a product, at least in spirit). The goal of the interface is to enable development of HSM and backup products on computer systems that virtulize the native filesystem without requiring OS modifications. Needless to say, there were many heated debates during the course of the meetings and on the DMIG reflector, but the group did finally agree on a set of interfaces known as the DMAPI.

---

[6] FileServ Software is a registered trademark of E-Systems.

[7] EMASS is a registered trademark of E-Systems.

[8] The companies listed are those that attended at least one meeting in the 1994 time frame and if I missed someone, I apologize in advance.

4.0 DMAPI [1]

The DMAPI is a set of interfaces to be provided by the base operating system that enables HSM and backup applications to gain access to native file system data and metadata transparent to normal applications. It includes the following basic concepts:

1: *Events* - Data Management (DM) applications can request to be informed of specific events like read(), write(), etc. The event notification is via messages which DM applications gain access to via the dm_get_events() call. There are two distinct message types; synchronous, and asynchronous. Synchronous messages have tokens associated with them. Asynchronous do not have tokens associated with them.

2: *Tokens*- a token is a reference to the state associated with a synchronous event message. The contents of a token are opaque to the DM application. DM applications can modify the contents of a token only through the dm_request_right() and dm_release_right() call. Rights can be granted to a DM application including DM_RIGHT_EXCL, DM_RIGHT_SHARED, and DM_RIGHT_NULL. DM_RIGHT_EXCL prohibits any access to a file except through DMAPI calls that accept tokens, and only then if the token with the DM_RIGHT_EXCL right is passed to the interface. DM_RIGHT_SHARED protects against any modification of the data or metadata associated with a file, by normal or DM applications, but will allow multiple accesses to the data or metadata. DM_RIGHT_NULL grants no rights.

3: *Managed Regions* - A managed region designates the portions of a file that the DM application is managing. There are 3 possible events that may be enabled on a managed region; DM_REGION_READ, DM_REGION_WRITE, and DM_REGION_TRUNCATE. If the corresponding action, read, write, or truncate, is attempted by a non DM application to the associated region, an event will be generated for the DM application that has expressed interest in the corresponding portions of the file. The number of managed regions supported by the DMAPI is implementation defined.

4: *Handles* - Pathname independent references to file system objects. A file handle uniquely identifies a file system object. Handles exist for file systems, directories, files, symlinks, and one special global handle that does not refer to any object but allows a DM applications to register for mount events.

5: *Sessions* - The interface between the DM application and the DMAPI is session based. The session is the identifier used to determine who receives events for a particular file system object. Sessions can be thought of as two things; a queue that event messages are queued ut to, and an identifier for use in tracking, auditing, and controlling access to the DMAPI facilities.

6: *Data Management Attributes* - Space for a DM application to store pertinent information about a file. The data in the attribute space is opaque to the DMAPI implementation. Examples of information that might be stored include the location of data from the file in the archive for migrated files. DM attributes are an optional portion of the DMAPI.

7: *Holes* - Holes can be created two ways. First is via the lseek() function. If an application seeks past the existing end of a file, and writes some data, implementation may create virtual space in a file without any corresponding storage allocated. The second way is via the DMAPI call dm_punch_hole(). This call frees the storage associated with the portion of the file where the hole is punched (i.e. frees file system space once a file is migrated to tertiary storage).

In general DM applications create a session, register for mount events, catch mount events and establishs interest in files by registering for events within a file system. They then catch events for the files of interest, and perform actions to migrate files in or out as needed.

DM applications are viewed as a part of the file system implementation and as such have no security restrictions placed on them. All calls to the DMAPI are expected to be run as superuser or some other file system permission that would give an application un-restricted access to the data and metadata in a file system. There is no association of DMAPI objects to any one process. Any application, running at the appropriate security level, with the correct session identifier and token identifier, can take actions on file system objects. The rational for this is to allow a DM application to use as many processes as required to deliver acceptable performance. There is also an underling assumption in the specification that only one DM application will attempt to control a file system. There is nothing to prevent multiple DM applications from controlling a file system, but coordination of the DM applications is outside the scope of the DMAPI. An example of a case where multiple DM applications may run on a single file system is where an HSM application and a Backup application cooperate to backup the HSM controlled file system. The reason for this is, for more than one application to control objects in a file system would require considerable machinery in the DMAPI. One of the primary goals of the DMIG group was to produce a specification that could be implemented in a six man month development project. To produce the machinery required would extend the level of effort far beyond the six man month goal.

The following are examples of DM applications pulled from the DMAPI specification. These are included to give a better idea of how the interfaces are intended to be used. [1]

4.1 Stageout

This example will stage out a 512 megabyte file names `/test/foo`. The first 64K will remain as a fence post. The remainder of the file will be staged out in 32 megabyte clusters.

The following concepts are illustrated:

- Use of the file change indicator

- Invisible read
- Setting managed region
- Punching holes

```
char            *buf;
void            *hanp;
size_t                  hlen;
size_t                  retrgns;
size_t                  nchunks;
int                     change_end, change_start,
off_t           off;
dm_token_t      filetoken;
dm_stat_t       statbuf;
dm_region_t             rgnbuf[2];
dm_size_t       roff;
dm_off_t                rlen;
dm_sessid_t     sid;

if (dm_init_service() == -1) {
        err_msg("Can't initialize DMI\n");
        return(0);
}

dm_create_session(DM_NO_SESSION, &sid, "generic_app");
dm_path_to_handle("/test/foo", &hanp, &hlen);

/*
 * In order to stat the file, we need a shared lock.
 */
dm_create_userevent(sid, 0, (void *)0, &filetoken);
dm_request_right(sid,hanp,hlen,filetoken,DM_WAIT,DM_SHARED);

/*
 * While writing the file out to tertiary storage, we drop locks. We first
 * get the file change indicator, and query it after we're done.
 * If it changed, then we give up
 */
dm_get_fileattr(sid, hanp, hlen, filetoken, DM_AT_CFLAG, statbuf);
change_start = statbuf.dt_change;

/*
 * We don't bother with any DM attributes, just the data.
 * We write the file out in 32 meg chunks.
 */
nchunks = (512 * 1MEG) / CHUNKSIZE;
for (i=0; i<nchunks; i++) {
        dm_read_invis(sid, hanp, hlen, token, off, CHUNKSIZE, buf);
        dump_data_to_archive(hanp, hlen, off, buf);
        off += CHUNKSIZE;
}
dm_release_right(sid, hanp, hlen, filetoken);

/*
 * Store an DM application specific information, such as the file size,
 * file handle, etc., with the data
 */
dump_myinfo_to_archive(hanp, hlen);
```

```
/*
 * Check the file change indicator to see if the file changed
 * while we were doing other things. If not, then set a managed
 * region on the file
 */
dm_request_right(sid, hanp, hlen, filetoken, DM_WAIT, DM_EXCL);
dm_get_fileattr(sid, hanp, hlen, filetoken, DM_AT_CFLAG, statbuf);
change_end = statbuf.st_change;
if (change_start != change_end) {
        err_msg("File changed, bailing...\n");
        do_cleanup();
        return(1);
}

/*
 * Set up the managed regions so that the first 64K won't cause events
 * to be generated, but a foray into the rest of the file will generate
 * events
 */
rgnbuf[0].rg_off = 0;
rgnbuf[0].rg_size = FENCESIZE;
rgnbuf[0].rg_flags = DM_REGION_NOEVENT;
rgnbuf[1].rg_off = FENCESIZE;
rgnbuf[1].rg_size = (512 * 1MEG) - FENCESIZE;
rgnbuf[1].rg_flags = DM_REGION_READ | DM_REGION_WRITE |
DM_REGION_TRUNCATE;
dm_set_region(sid, hanp, hlen, filetoken, 2, rgnbuf, &retrgns);

/*
 * Punch a hole in the file. We assume that we know that what the
 * rounding constraints are so that we don't have to do a dm_probe_hole()
 */
dm_punch_hole(sid, hanp, hlen, token, rgnbuf[1].rg_off, rgnbuf[1].rg_size,
                &roff, &rlen);
/*
 * We're done. Release the token
 */

dm_respond_event(sid, filetoken, 0,(void *)0, DM_CONTINUE, 0);
```

4.2 Stagein

This example will stage in the file that was staged out in the above example.

There is a master daemon that receives messages from the kernel, and sends them on to worker bees for processing. The master daemon is only monitoring the ***read***, ***write***, and ***truncate*** managed region events in this example for the filesystem **/test**. There is some magic here that is not shown. The master daemon knows about, and has access to, two other sessions that are used to perform event-specific handling. Information is shared between the master daemon and these other processes through some application-specific mechanism that is not shown. This could be shared memory, a socket, a well-known file, or any other mechanism. The details are left to the dazed reader.

The following concepts are illustrated:

- Sharing of tokens and sessions
- Setting event disposition
- A simple get_event loop
- Having a master daemon *move* an event to another session
- Complex lock upgrade
- Setting managed regions

```
extern dm_sessid_t  rw_sid, trunc_sid;

void                *fs_hanp;
void                *msgbuf;
size_t                      fs_hlen;
size_t                      msgsize, msgbuflen;
dm_sessid_t         sid;
dm_token_t          fs_token, newtoken;
dm_event_set_t      eventset;
dm_eventmsg_t       *msg;

if (dm_init_service() == -1) {
        err_msg("Can't initialize DMI\n");
        return(0);
}

dm_create_session(DM_NO_SESSION, &sid, "generic_app");

/*
 * Since we'll be communicating with other processes, we do some
 * magic setup to get their sids, establish communications channels,
 * etc. We could use dm_send_event(), stuff the data in a shared memory
 * region, open a socket, or whatever
 */
setup_communications(sid, &rw_sid, &trunc_sid);

dm_path_to_fshandle("/test", &fs_hanp, &fs_hlen);

/*
 * Now get a token and rights so that we can set the disposition
 * of events
 */
dm_create_userevent(sid, 0, (void *)0, &fs_token);
dm_request_right(sid, fs_hanp, fs_hlen, fs_token, DM_WAIT, DM_EXCL);

/*
 * Set the disposition of the events we want to monitor
 */
DMEV_ZERO(eventset);
DMEV_SET(DM_READ, eventset);
DMEV_SET(DM_WRITE, eventset);
DMEV_SET(DM_TRUNCATE, eventset);
dm_set_disp(sid, fs_hanp, fs_hlen, fs_token, &eventset, DM_MAX_MSG);

dm_release_right(sid, fs_hanp, fs_hlen, fs_token);
```

The master daemon now enters a simple loop where it will spend all its time. It simply asks the DMAPI for more messages and dispatches them to its worker processes.

```
/*
 * Find out the size of the largest message that can be delivered
 * on this filesystem. We use this to size an event buffer to get
 * an arbitrary (16 in this example) number of messages at the same
 * time.
 */
dm_get_config(fs_hanp, fs_hlen, DM_MAXMSG_SIZE, (long)&msgsize);
msgbuflen = msgsize * 16;
msgbuf = (void *)malloc(msgbuflen);

/*
 * Enter a simple loop, looking for messages. We don't worry about
 * resizing the buffer
 */
for (;;) {
        dm_get_events(sid, msgbuflen, msgbuf, &ret_msglen, 0, DM_WAIT);
        msg = (dm_eventmsg_t *)msgbuf
        while (msg != NULL) {

                /*
                 * For read and write events, we send them to other processes
                 * with 'well known' sids that are handling these things.
                 */
                if (msg->ev_type == DM_READ || msg->ev_type == DM_WRITE) {
                        dm_move_event(sid, msg->ev_token, rw_sid, &newtoken);

                } else if (msg->ev_type == DM_TRUNCATE) {
                        dm_move_event(sid, msg->ev_token, trunc_sid, &newtoken);

                } else {
                        err_msg("Unknown event type\n");
                        dm_respond_event(sid, msg->ev_token, 0,(void *)0,
                                DM_ABORT,EINVAL);
                        continue;
                }
                msg = DM_STEP_TO_NEXT(msg, dm_eventmsg_t *);
        }
}
```

The worker bee processes also do some initial setup, which won't be shown. For a simple stagein, we'll assume we've receive a ***DM_READ*** event on a managed region. We join our fearless process at the point in which it has received the event that was directed to it from the master daemon, and is starting to reload the data.

```
int             change_start;
void            *hanp;
size_t          hlen;
size_t          nchunks;
size_t          retrgns;
dm_off_t        off;
dm_size_t       len;
dm_right_t      right;
dm_sessid_t     sid;
dm_stat_t       statbuf;
dm_eventmsg_t   *msg;
dm_data_event_t *read_event;
```

```
dm_region_t                rgnbuf[1];

msg = eventbuf;
read_event = DM_GET_VALUE(msg, data, dm_data_event_t *);

hanp = DM_GET_VALUE(read_event, handle, void *);
hlen = DM_GET_LEN(read_event, handle, size_t);

/*
 * Check to see what the rights are that came with the message. If
 * they aren't exclusive, we must go get them
 */
dm_query_right(sid, hanp, hlen, msg->ev_token, right);

if (right == DM_SHARED) {
        /*
         * We really need exclusive. We'll try to upgrade the lock,
         * but if that fails, we'll have to drop it and go to
         * sleep.
         */
        if (dm_request_right(sid, hanp, hlen, msg->ev_token, DM_EXCL,
                        DM_NOWAIT) == -1
        {
                if (errno != EAGAIN) {
                        err_msg("Can't upgrade lock\n");
                        do_cleanup();
                        return(1);
                }
                /*
                 * Before we drop the lock, get the file change indicator
                 */
                dm_get_fileattr(sid, hanp, hlen, msg->ev_token, DM_AT_CFLAG,
                        statbuf);
                change_start = statbuf.dt_change;

                dm_release_right(sid, hanp, hlen, msg->ev_token);
                dm_request_right(sid,fshanp,fshlen,msg-
>ev_token,DM_WAIT,DM_EXCL);

                /*
                 * Now that we've come back from sleeping, see if the file changed.
                 * If so, we just bail.
                 */
                dm_get_fileattr(sid, hanp,hlen,msg->ev_token,DM_AT_CFLAG,statbuf);
                if (statbuf.dt_change != change_start) {
                        err_msg("File changed. Bailing...\n");
                        do_cleanup();
                        return(1);
                }
        }
} else if (right == DM_NONE) {
        dm_request_right(sid,hanp,hlen,msg->ev_token,DM_WAIT,DM_EXCL);}
}
```

The worker bee is now at the point where it has exclusive access to the file. This is needed for dm_write_invis().

```
/*
 * Now that we have exclusive access to the file, we need to
```

```
 * go off and find where we stored the file's data in our
 * repository
 */
offset = DM_GET_VALUE(read_event, offset, dm_off_t);
len    = DM_GET_VALUE(read_event, len, dm_size_t);
find_our_file(hanp, hlen, offset, len);


/*
 * Restore the data for the file.
 * We'll assume that the file length is some nice integral multiple
 * of our chunksize;
 */
nchunks = len / CHUNKSIZE;
for (i=0; i<nchunks; i++) {
        get_data_from_archive(hanp, hlen, off, buf);
        dm_write_invis(sid, hanp, hlen, token, off, CHUNKSIZE, buf);
        off += CHUNKSIZE:
}


/*
 * Clear the managed region
 */
rgnbuf[0].rg_off = 0;
rgnbuf[0].rg_size = 0;
rgnbuf[0].rg_flags = DM_REGION_NOEVENT;
dm_set_region(sid, hanp, hlen, msg->ev_token, 1, rgnbuf, &retrgns);

/*
 * We're done. Release the token
 */

dm_respond_event(sid, msg->ev_token, 0,(void *)0, DM_CONTINUE, 0);
```

4.3 DMAPI Status

The current status of the DMIG is that the V2.0 version of the interface specification is out for industry review. Several companies are actively prototyping the interface, and some are close to having product available in the marketplace. Below is the status of several companies polled for their stance regarding the DMAPI:

1: Convex - Development under way for C-Series platforms for NAStore product. Development underway for HP platforms supported as data management platforms. Plans to port DMAPI to SPP product line in Q1-Q2 1995.

2: Hitachi Computer - Development underway to provide DMAPI on Veritas file system to integrate with Epoch product.

3: IBM - Prototyping underway. No firm plans to release support

4: SGI - Will release support for DMAPI in first half of 1995 in their XFS file system.

5: Sunsoft - Plan is to watch marketplace. No plans currently exist to support DMAPI.

6: Legent - "Legent, through its acquisition of Lachman Technology, has been behind the DMAPI interface ever since the group began meeting. We believe in the goals of the group and anxiously await adoption of the interface by UNIX operating system vendors. DMAPI will strengthen these platforms and will help vendors like Legent offer its storage management products across the broadest range of available systems."

Although not all vendors have committed resources to the DMAPI, it is clear from the above list, that several vendors believe that it is a viable interface and worth investing in. Given that the early adopters of the interface are successful in the market place, other companies are likely to put plans in place to formally support the DMAPI.

5.0 UniTree+ and the DMAPI

In section 2.0, several of the deficiencies in UniTree+ were described. It fundamentally is a result of the fact that UniTree+ does not virtualize the native file system. What was not discussed in section 2.0 was that UniTree+ has the capability to handle 1,000,000+ files and many terabytes of data in the archive. The combination of UniTree+ and an access method that virtualizes the native file system yields a product that provides the functionality and performance of native file system accesses, and the ability to support very large archives with high data ingestion and retrieval rates.

5.1 Standard UniTree+

UniTree+ as shipped today has an architecture that consists of a the following major blocks:

1: Nameserver - A server that maps pathname to capability ID. It supports a namespace that looks very similar to traditional UNIX[9] file system namespace.

---

[9]UNIX is a registered trademark of UNIX Systems Laboratories, Inc., a wholly owned subsidiary of Novell, Inc.

2: Disk Server - The server responsible for providing magnetic disk cache for data in the repository. All access methods read and write through the disk cache. No direct reading or writing of tape is supported.

3: Tape Server - The Server responsible for moving data to and from the tape system(s) and the magnetic disk cache. Provides mapping for capability ID to tape location mapping.

4: Access Servers - Daemons that provide the FTP and NFS protocols.

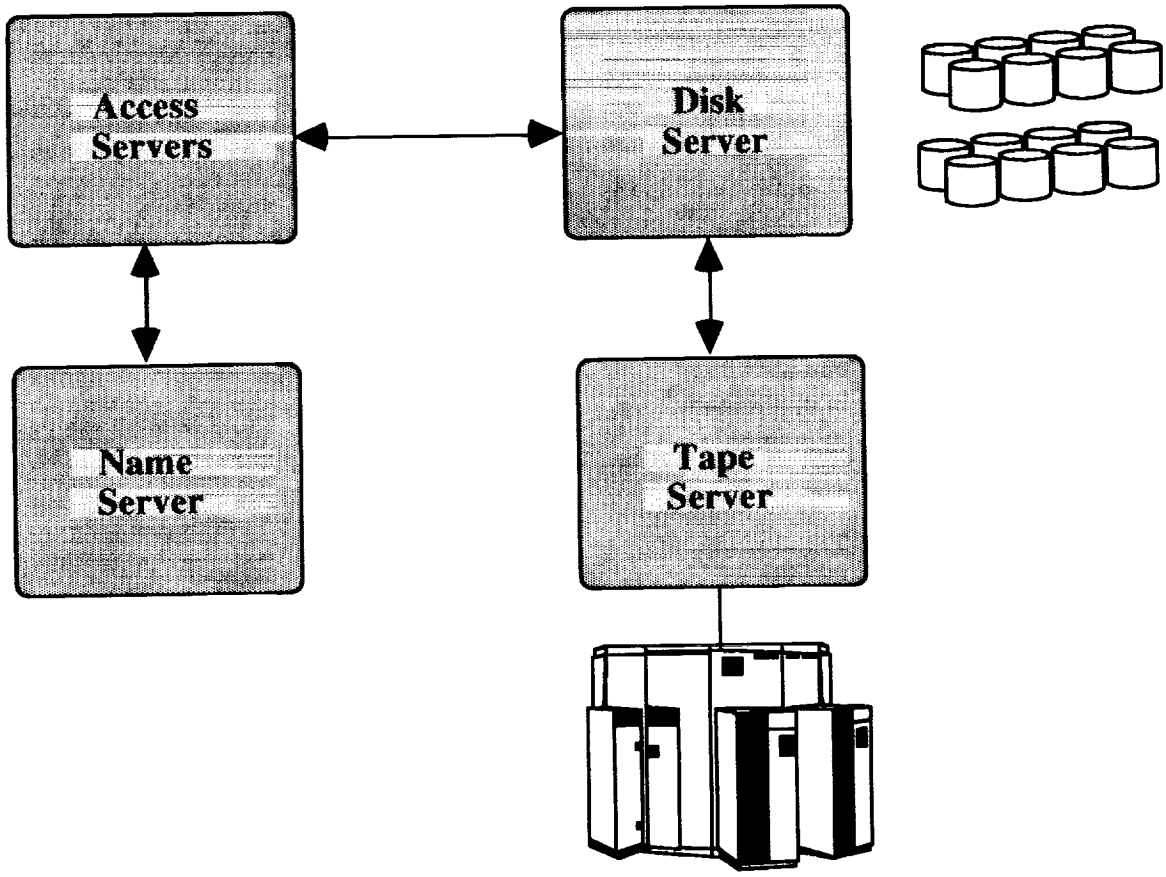


Figure 1: UniTree+ Today

The problem areas in the above architecture reside in the access servers and the name servers. All accesses traverse a network protocol. Even same machine accesses go through the localhost interface. The bulk of the slowdowns in the UniTree interface is because of latency associated with small transfers. This is especially true for metadata operations. If

you view the graph below comparing native HP/UX[10] file system operations versus UniTree+ running on the same HP[11] system, you will notice a substantial difference in wall clock execution time. Times shown as .1 seconds returned zero seconds when measured with the HP/UX *time* command because the granularity of wall clock time is 1 second. There was no effort made to optimize the times shown below, but they are instead just an indication of relative execution times between native file system operations and UniTree+.

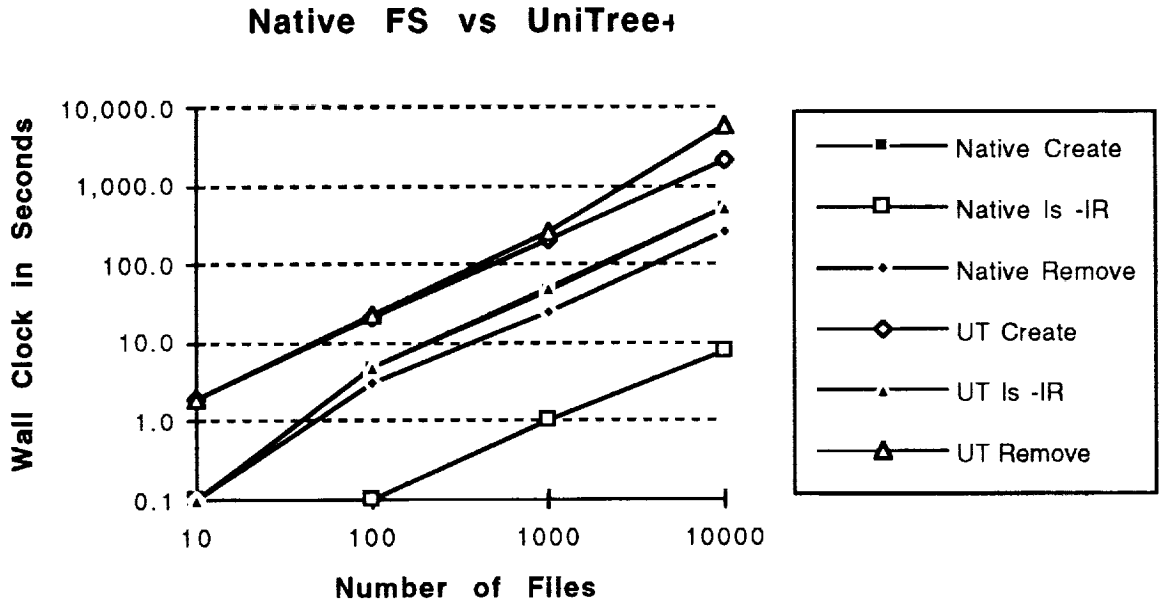


Figure 2: Performance Graph of Native File System versus UniTree+

All of the files created in the above test were of zero length. UT 10,100,1000,10000 refers to UniTree performing the operation listed on 10,100,1000,10000 files respectively, where Native 10,100,1000,10000 refers to the same operations executed on the Native file system and 10,100,1000,10000 files respectively. Creates of zero length files in UniTree+ involve primarily the NFS access daemon and UniTree+ name server. Replacement of those two servers in UniTree+ with native file system operations will dramatically improve the metadata operations in the resulting HSM.

---

[10]HP/UX is a registered trademark of Hewlett Packard Corp.

[11]HP is a registered trademark of Hewlett Packard Corp.

5.2 Convex Virtual Disk Manager (CVDM)

The Convex Virtual Disk Manager (CVDM) does exactly what the name implies. It's function is to control space allocation in native file systems using the DMAPI. It currently runs on ConvexOS and HP/UX systems controlling the UFS native file systems. The major functional blocks are as follows:

1: Migd - Responsible for startup and shutdown of system.

2: SSD - Responsible for creation of daemons for each file system put under CVDM control.

3: Migdmon - Daemon responsible for interfacing with the DMAPI to catch events and start actions required by the events.

4: Migout - Responsible for scheduling migrate out processing.

5: Migin - Responsible for migrating in data as a result of an access to non-resident data within a file.

6: Interface Manger - Responsible for communication with UniTree+ back end.

The structure of the daemons looks as follows:

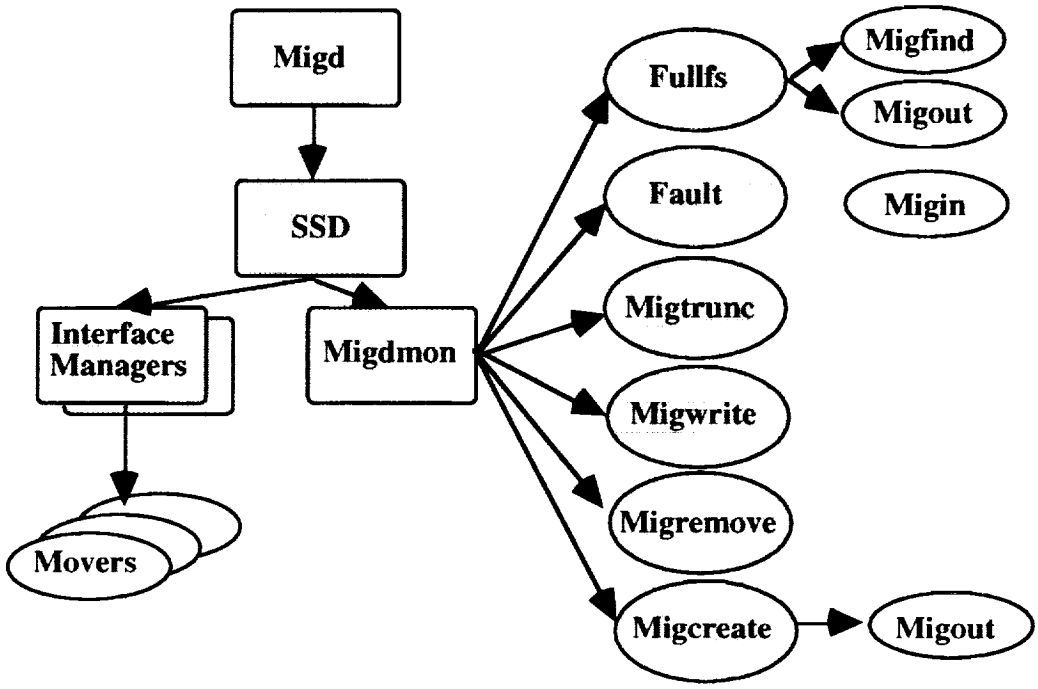


Figure 3: CVDM Server Architecture

The interface manager and movers are responsible for communicating with the UniTree+ back end. As a result of the separation of CVDM from the UniTree+ archive portion, multiple instances of CVDM can communicate with multiple instances of UniTree+. The following diagram illustrates a possible configuration:

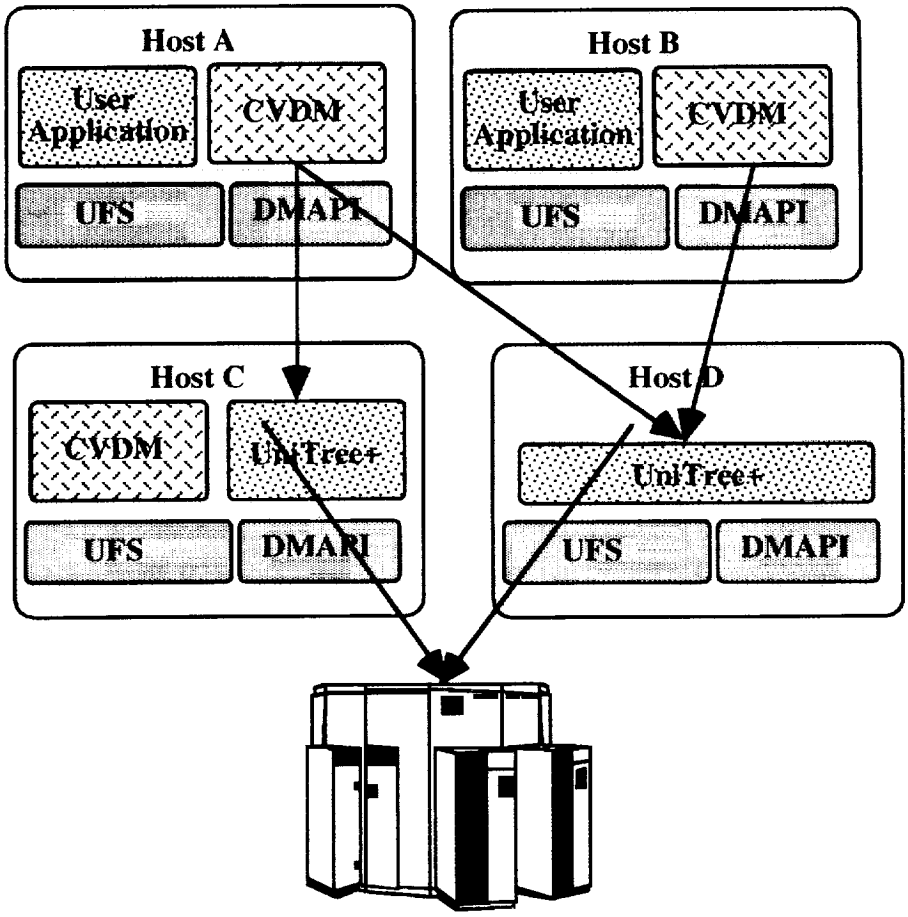


Figure 4: CVDM/UniTree+ Configuration Example

The above example configuration shows a configuration where the access servers are distributed over three distinct server platforms. The archive servers are also distributed over two distinct server platforms. This allows the site to tailor their system to meet the load that their environment places on the servers. If the system tends to be a read mostly system, increasing the number of access servers improves performance of the data retrieval. If a site uses their system in a write mostly environment, increasing the number of UniTree+ archive server platforms allows a system to handle very large data ingestion rates.

The architecture of a system running CVDM and UniTree+ on the same system appears as follows:

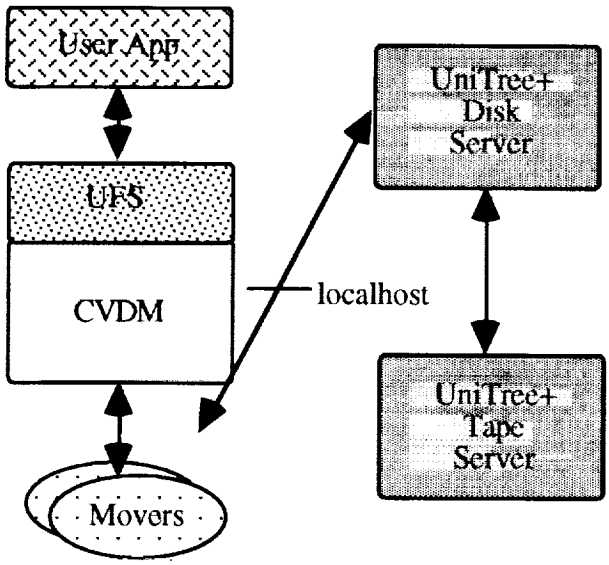


Figure 5: CVDM/UniTree+ Architecture

As shown above, the user application accesses the file data exactly as it would in a non HSM environment. Performance on resident files has shown to have < 1% deviation from native file systems not under HSM control. Except for delays for non resident files, the HSM is completely transparent to the application.

Migration in the current system transfers data from the native file system, via CVDM movers, to the UniTree+ disk cache, via UniTree+ disk movers. The data in the disk cache is marked to purge immediately and is therefore scheduled to move to tape from the disk cache in the next UniTree+ migration round. Requiring files to move from native disk to the UniTree+ disk cache has benefits, but also introduces some problems. The benefit of caching data is it allows remote CVDM servers to transfer data at network speeds. Data then is written to tape at tape speeds. The disk cache acts like a rate matching buffer between the network and tape devices. Tape devices are never tied up waiting on network transfers. The drawback is that the UniTree+ disk cache must be large enough to hold the largest file being migrated. Moreover, for local CVDM servers, there is no need for the rate matching buffer because the data can come from local disk. Non of these issues are architectural in nature, but instead are a result of merging CVDM and UniTree+ with extremely minimal changes to the UniTree+ base. In fact, UniTree+ can support both CVDM access servers and the existing UniTree+ access servers simultaneously.

Migration rounds in CVDM can be initiated in a variety of ways. Administrators can start a migration round via cron jobs or manually. Candidate selection is also configurable. A utility, migfind, scans the UFS name space looking for possible candidates. Weights are assigned to file attributes like size, modification time, creation time, owner, etc. Migfind then generates a sorted list of candidates to hand to migout, which in turn initiates migration. Migration rounds can also be initiated when space in a file system crosses preset thresholds. In this case, the system will first look for files that are migrated, but whose data is still cached in the file system. For these files, space can be freed by punching a hole in the file with the dm_punch_hole() call. If enough space is not freed by pruning cached data, a full migration round is initiated.

5.3 CVDM Futures

In future releases of CVDM, the UniTree+ tape mover and the CVDM disk mover will be collapsed into a unified mover for local CVDM servers. This unified mover will only be used for the local CVDM case so the rate matching nature of the UniTree+ disk cache will be retained for the remote servers. The use of a unified mover will dramatically decrease the overhead associated with migration from local CVDM servers.

Remote CVDM servers benefit greatly from buffering data in the UniTree+ disk cache. For large files, it may desirable to start the migration of data from the UniTree+ disk cache to the tape archive prior to completely transferring all of the file data. This is especially true for high bandwidth networks like HIPPI which can transfer data at rates above existing tape transports, or FDDI combined with low transfer rate tape drives. The UniTree+ system will be extended to support this.

6.0 Summary

The DMAPI has the potential to increase both the availability and quality of HSM products while providing functionality only available in the kernel intrusive implementations of today. As is usually the case, availability of the DMAPI will be driven by the market place. If customers ask for it, or as competitors begin winning sales because they have it, more OS and HSM vendors will deliver products based on it.

[1] Information in this section was obtained from the Data Migration Interface Group - Interface Specification. Version 2.0. This document is available via anonymous FTP at internet address 143.127.0.2

Constraint Based Scheduling for the Goddard Space Flight Center Distributed Active Archive Center's Data Archive and Distribution System

Nick Short Jr. - Information Science and Technology Branch
NASA - GSFC
Greenbelt Road
Greenbelt, MD 20771
301-286-6604
short@dunloggin.gsfc.nasa.gov

Jean-Jacques Bedet and Lee Bodden - Hughes STX
7701 Greenbelt Road, suite 400
Greenbelt, MD 20770
301-441-4285 Fax (301) 441-2392
{bedet,bodden}@daac.gsfc.nasa.gov

Mark Boddy, Jim White, and John Beane - Honeywell Technology Center
Honeywell Technology Center
3660 Technology Dr.
Minneapolis, MN 55418
612-951-7355 Fax 612-951-7438
{boddy,jwhite,beane}@src.honeywell.com

Abstract

The Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC) has been operational since October 1, 1993. Its mission is to support the Earth Observing System (EOS) by providing rapid access to EOS data and analysis products, and to test Earth Observing System Data and Information System (EOSDIS) design concepts. One of the challenges is to ensure quick and easy retrieval of any data archived within the DAAC's Data Archive and Distributed System (DADS). Over the 15-year life of EOS project, an estimated several Petabytes (10^15) of data will be permanently stored. Accessing that amount of information is a formidable task that will require innovative approaches. As a precursor of the full EOS system, the GSFC DAAC with a few Terabits of storage, has implemented a prototype of a constraint-based task and resource scheduler to improve the performance of the DADS.

This Honeywell Task and Resource Scheduler (HTRS), developed by Honeywell Technology Center in cooperation with the Information Science and Technology Branch/935, the Code X Operations Technology Program, and the GSFC DAAC, makes better use of limited resources, prevents backlog of data, and provides information about resource bottlenecks and performance characteristics. The prototype which is developed concurrently with the GSFC Version 0 (V0) DADS, models DADS activities such as ingestion and distribution with priority, precedence, resource requirements (disks and network bandwidth) and temporal constraints. HTRS supports schedule updates, insertions, and retrieval of task information via an Application Program Interface (API). The prototype has demonstrated with a few examples, the substantial advantages of using HTRS over scheduling algorithms such a First In First Out (FIFO) queue. The kernel

scheduling engine for HTRS, called Kronos, has been successfully applied to several other domains such as space shuttle mission scheduling, demand flow manufacturing, and avionics communications scheduling.

Introduction

The main objective of the Code X Operations Technology Program (X-OTP) is to provide advanced techniques in order to reduce NASA's operational costs by focusing on reusable software technology. In addition to numerous technologies such as electronic documentation, database management systems, system diagnosis, and data analysis tools to name a few, one of the successful areas of X-OTP has been the application of planning and scheduling technologies to missions operations throughout NASA. In cooperation with the GSFC DAAC and Honeywell Technology Center, X-OTP has initiated a program to apply scheduling technology to various areas within the EOSDIS. In addition to providing local management for this project, the Information Science and Technology Branch, which is part of the GSFC supercomputer facility or the Space Data and Computing Division, has been providing its Intelligent Information Fusion System (IIFS) as a modular, end-to-end, advanced prototype system for testing these new technologies. Free from many requirements of operational systems, this prototype system is being used to guide several of the technological extensions for this scheduling project.

This paper presents the first phase of this project by discussing the capabilities of the Honeywell Task and Resource Scheduler (HTRS) as they apply to the scheduling of operations in large mass storage systems. The GSFC DAAC architecture is briefly introduced and the main DADS functions are described as they relate to mass storage issues. The approach used to solve scheduling issues and the specific DADS scheduler requirements is then explained. The architecture of the scheduler, its domain model, and an application Program Interface (API) to communicate with the scheduler is also presented. In particular, the paper describes the application of a constraint-based scheduling to a mass storage system for the management of data ingestion, dissemination over a network environment, and distribution of datasets copied to tapes. Due to the large number of daily tasks and their dependencies, the slow seek time on tapes, and deadlines which must be met, a First In First Out (FIFO) scheduling algorithm, as well as other queuing approaches, is not adequate. HTRS increases the throughput of the various DAAC activities by making efficient use of the DAAC's computer resources. The HTRS is an adaptive, dynamic scheduler capable of modeling numerous system resources such as disk storage, robotic devices, processors, memory, and network bandwidth. HTRS handles resource contention, prevents deadlocks, and makes decisions based on a set of defined policies. By modeling database operations as tasks with priority, precedence, duration, resource requirements and temporal constraints, HTRS efficiently supports schedule updates, insertions, and retrieval of task information.

General Scheduling Issues

Given many of the misconceptions about scheduling, this section will cover a brief summary of the common definitions and topics surrounding data processing scheduling for those readers not familiar with the terminology in the following sections. In general, most non-real-time Operating Systems (OS) handle task management by assuming that tasks operate independently of each other and that execution characteristics cannot be accurately determined a priority. Hence, simple queuing methods dominate this category, often providing sub optimal solutions (e.g., FIFO scheduling). The UNIX OS, in fact, was designed for general purpose workstations where little is ever known about task characteristics.

Improvements to these approaches require an analysis of the operating characteristics of typical tasks, such as determining if tasks have priority or deadlines, arrive periodically or arbitrarily, operate in a uniprocessor/multiprocessor or heterogeneous/homogeneous environments (i.e., different or same processors), exhibit predictable resource properties, and organize into a data flow graph (i.e., tasks whose execution precedes and passes data to others). These improvements are constrained by the operational requirements such as trying to minimize task completion time, demanding that most or all tasks meet their deadlines (i.e., soft real-time or hard real-time), and allowing tasks to be preemptable or nonpreemptable to name a few.

Based on these characteristics, the *scheduling problem* can be defined as given a set of tasks *T* associated with a subset *C* of the aforementioned constraints, determine the execution sequence, if possible, that best satisfies *C*. Two basic types of scheduling approaches exist: *static* (or deterministic) and *dynamic* (or non-deterministic). Static schedulers create schedules off-line after all task information has been collected while dynamic schedulers determine schedules on-line during continuous data collection. Any static scheduler is optimal only if it produces schedules that satisfy *C* whenever any other scheduler satisfies *C*. A dynamic scheduler, however, produces an optimal schedule if it always produces a feasible schedule when a static scheduler with complete information can create one. Obviously, static schedulers are always sub-optimal when the collected information changes before the schedule is produced, regardless of which scheduling algorithm is used. While dynamic schedulers suffer less from this problem, they incur a lot of overhead due to the cost of constantly collecting information. For this reason, many schedulers utilize a *hybrid* approach where scheduling is done off-line while adjustments are made on-line.

Related to this issue, schedulers are also classified as *adaptive* or *non adaptive* depending on whether the environment provides feedback to the scheduler. That is, the scheduler's control mechanism changes in response to system histories or trends. Dynamic schedulers are almost always adaptive. Of course, the type of information collected determines how well the scheduler performs. Estimates of task duration can be based upon best, average, or worst-case estimates depending on optimism or pessimism. Other statistics can include modeling the average number of tasks arriving for particular times, hot spots for resource usage, inter-task communications costs, etc. Determining how refined the statistics model always depends on the performance requirements, which often change to meet evolving bureaucratic policies.

Institutional requirements usually determine the control architecture of the scheduling environment. For example, *Centralized* systems such as shared memory models are those where processors essentially operate in a group where inter processor communication costs are minimal with respect to processor execution costs. By contrast, decentralized systems such as wide area networks (e.g., the DAAC's) imply high inter processor communication costs. Often times, centralized or distributed scheduling means that the computing environment is centralized or decentralized. This should not be confused with the much harder problem of using multiple schedulers to control a distributed environment versus using a centralized scheduler to control a distributed environment.

Adding to the confusion, the power of scheduling algorithms is often overestimated. For example, scheduling tasks with arbitrary precedence between tasks for multiprocessors is proven to be NP-hard (i.e., essentially known to take an exponential number of steps as a function of the number of tasks) with only unit execution time, regardless of whether tasks are preemptive or non preemptive. Hence, because most of the non NP-hard algorithms (i.e., polynomial) are too restrictive, schedulers realistically must utilize heuristic

approaches (i.e., smart guessing) while searching for feasible schedules. This involves the construction of a function, often called an *objective function*, that encapsulates a notion of "goodness" for evaluating one proposed schedule versus another during the search through the space of possible schedules. Objective functions can be explicitly represented by numeric formulae for simple comparison or they can be implicitly captured in the scheduling policy algorithm. Regardless, the objective function or scheduling policy algorithm should be flexible enough to change as the institution governing the processing environment modifies its notion of a good schedule. For instance, an institution may want to guarantee that all or most task deadlines are met one day while on another day, it may wish to minimize completion time of tasks.

Scheduling issues with mass storage systems

Today's mass storage systems are critical resources that usually must operate in a complex and changing institutional environment. These institutions must process large volumes of data while providing efficient and reliable service to a large number of users, who typically request resources at unpredictable times. Satisfaction of these users is critical in order to justify the enormous investment required to run these large institutions. Also, proper decision making about which resources is absolutely necessary for controlling the high cost of these computing environment. Scheduling technologies allow institutions to provide services according to reasonable user deadlines while providing information about which resources are bottlenecks that must be alleviated with the purchase of appropriate hardware.

These characteristics are certainly true of the EOS architecture and, in particular, are being evaluated in the context of the GSFC DAAC -- a system that is intended as an operational testbed for EOS. Although nowhere near the size of the final EOS system, the GSFC DAAC is estimated to process 250 SeaWiFs orders per day, corresponding to 40 GB of data. In addition, 20 GB of non-SeaWiFS products are expected to be ordered each day while 26 GB of new data will be ingested. Due to the large number requests and the large volume of data to process, manually generating feasible schedules will not be possible. Moreover, using the FIFO queue approach is not an acceptable solution because it does not make the best use of the resources available (e.g., tape drive, disk space), it doesn't have the ability to guarantee that most deadlines are met, and it provides little information about resource bottlenecks.

Of particular note, each request (e.g., distribution) has several tasks that must be scheduled individually. For example, to process an order for data requested on an 8-mm tape, the tasks may consist of retrieving the data from near-line devices, transfer the files to a staging area, and then copy the files to an 8-mm tape. Overall thousands of tasks with predecessor and successor tasks, each with specific needs for resources, must be scheduled and tasks cannot be treated equally. Requests for data to be sent over the network may be given a higher priority than data requested on tapes. Hence, the schedule should reflect these DAAC policies that determine, for example, deadlines and priorities.

Given that many of the operations involve transfer from one medium to another, proper migration from slower devices such as mass storage to faster devices is necessary to minimize average access times. That is, anticipation of requests should cause data to be moved into faster devices "just in time" for the request to be satisfied. This, of course, is similar to the notion of "locality of reference" in any memory hierarchy where the storage management system pre-fetches blocks of data in anticipation of future access to those blocks. Only here, the pre-fetching is also based on models of the external task environment in addition to load characteristics of the tasks, implying that a powerful scheduler can reduce access times by performing tasks just in time for delivery.

Because of the need to quickly anticipate trends in the external and internal processing environment, another challenge is to have a schedule that can be dynamically and quickly updated when new orders are received, when some of the resources become unavailable during a period of time, or when a given resource must be restricted to improve the overall performance of the system. For an example of this last category, tests conducted at the DAAC have shown that the number of concurrent NFS actions between the Unitree cache and the distribution staging area had to be limited to six or seven in order to achieve an acceptable throughput. Thus, the scheduler should model resources such as NFS resources to anticipate the proper localities of reference.

While the anticipation of many actions can be automated, many external events to the system requires that a human operator be present to make adjustments. That is, in any symbiotic production environment involving both computers and humans, tools must exist to help operators identify the status of the orders and their respective tasks as they relate to policies provided by management. For example, an estimated completion time for each task could be presented to the operator in order for the operator to communicate information back to high priority users.

These estimates should be based not only on the approximate duration of each task but on the availability of the resources. In fact the estimates for each task could be complex, however, the actual and the estimated times can be continuously compared so that better statistical approaches can be introduced. After conducting several tests simulating next year's workload, it became clear that scheduling was very important.

GSFC DAAC architecture

The GSFC DAAC has been developed to support existing and pre-EOS Earth science datasets, facilitate scientific research, and test EOSDIS operational concepts. Its design is based on the EOSDIS functional requirements and the requirements generated by specific Science projects such as Sea-viewing Wide Field-of-view Sensor (SeaWiFS).

GSFC DAAC has three main components illustrated in Fig 1. The Product Generation System (PGS) receives low-level data products and generates higher level data products. The Data Archive and Distribution System (DADS) role is to archive all new data products and to distribute over the network or on a variety of physical media, data ordered by researchers. The Information Management System (IMS) is a data base of the data holdings which can be searched, browsed by researchers to help them identify and order data of interests.

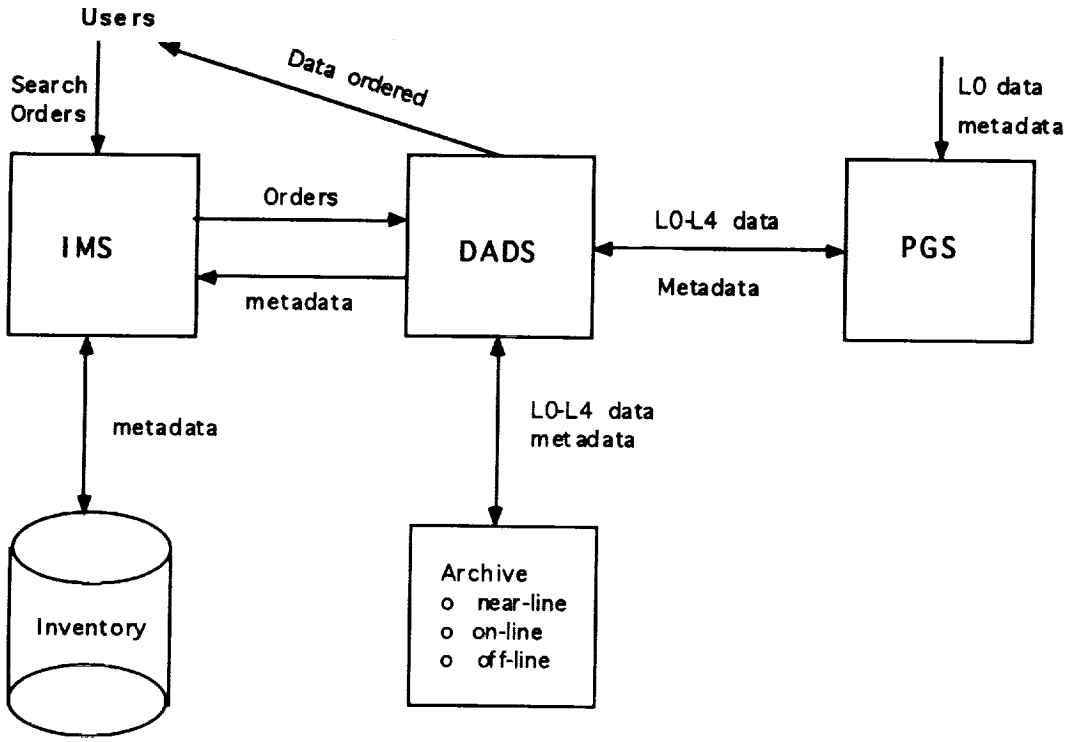


Figure 1 GSFC DAAC components

Although smaller than the overall facilities in the Space Data and Computing Division, the GSFC DAAC has currently 731 GB of data archive but this number is expected to increase to about 18 TeraBytes by FY97 [1]. To satisfy these requirements the GSFC DAAC has the following hardware architecture.

• The IMS system with its Oracle data base runs on a dedicated SGI 4D/440 VGX.

• The DADS software and the Hierarchical Storage Management (HSM) system Unitree to automate the migration and the stage operations, run on a SGI 4D/440 S. Data are archived either on a Cygnet 1803 jukebox (1179 MB) with 2 ATG WORM drives or an RSS-600 Metrum Automated Tape Library (ATL) (8700 MB) with 4 RSP 2150 VHS drives. The SGI 4D/440 S was too limited in terms of I/O bandwidth and ports. A SGI challenger L (DADS2) has been acquired to handle all the distribution copies on tapes. There are currently nine 8 mm drives, four 4 mm drives, and two 9 track drives attached to the EOSDADS2 machine.

• There is a future plan to build a Backup system that will run on an SGI Challenge S. Its function will be to keep a second copy of all data ingested at the DAAC.

• The PGS is composed of 3 SGI workstations. Two additional workstations are used to do Q/A on the data.

• The DAAC's distributed environment includes two Ethernet Local Area Networks, and an FDDI network.

GSFC V0 DADS functions

The three main functions of the DADS are archive, distribution, and data management. The archive function consists of accepting data products from outside the system, extracting metadata, validating files, and updating the database. The distribution function retrieves files from archives, stages them to a distribution staging area, reformats the data if necessary (e.g., tar is the normal format for orders), and writes the data to tapes or to the FTP staging disk. The DADS management handles the schedules, tracks DADS activities, and allocates/deallocates resources.

DADS V0 Scheduler

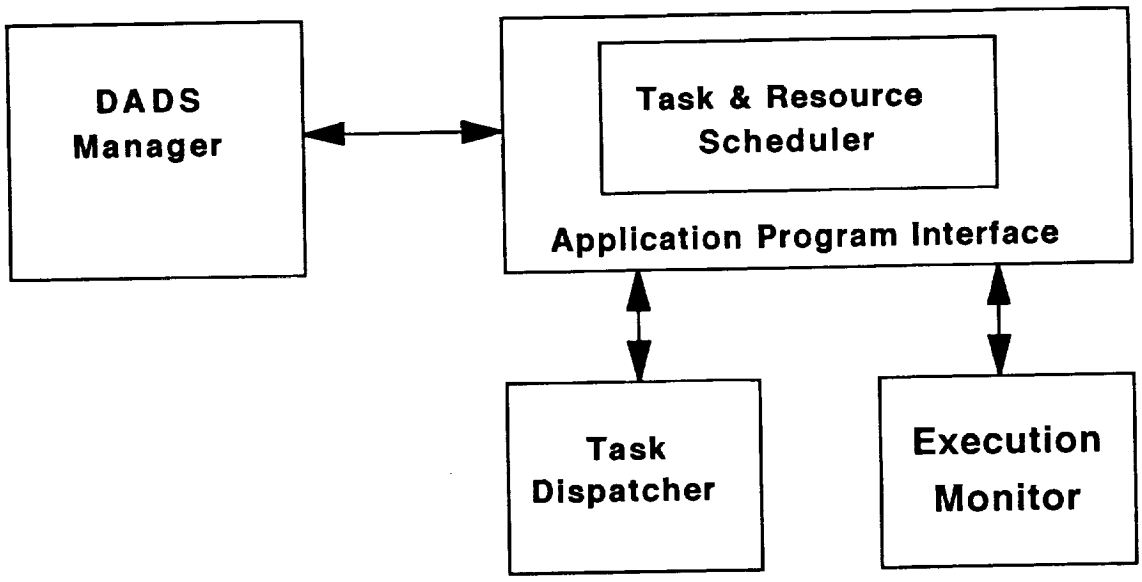


Figure 2. The HTRS Scheduler's Architectural Environment

The DADS V0 Scheduler is responsible for scheduling actions and resources to ingest data from a network to buffer disks, transfer buffered or cached data to a mass storage archive, and to retrieve archived data upon request. The scheduler was developed concurrently with the design and implementation of the GSFC V0 DADS. Consequently, the architecture and interfaces must tolerate changes as the system design evolved. The current version of the DADS software uses a multi-level priority queue algorithm, as a baseline system, to schedule its activities, however, there are plans to integrate the Honeywell Task and Resource Scheduler to the DADS for performance improvements. The baseline architectural environment of the HTRS scheduler is depicted in Figure 2. This environment continues to evolve, but its conceptual and functional characteristics remain stable, so many system changes can be accommodated in the Application Program Interface (API).

The DADS Manager submits scheduling requests, handles errors, and retrieves schedule information. The Task Dispatcher periodically queries the scheduler for a list of upcoming scheduled activities to be executed. The execution monitor notifies the scheduler of events that affect the schedule.

Approach

Constraint envelope scheduling technology offers an attractive, proven method of meeting the scheduling needs of data archiving and distribution. This technology, embodied in Honeywell's enhanced implementation of the Time Map Manager (TMM), supports the concept of a Temporal Constraint Graph (TCG) which can be used to represent multiple projections of future system behavior, thereby providing rapid rescheduling with minimal disruption in the presence of schedule uncertainty or changing policy situations.

The DADS V0 Scheduler is an application of the Kronos scheduling engine that is built on top of TMM and designed to be adaptive and dynamic. Kronos has been successfully applied to domains such as space shuttle mission scheduling, demand flow manufacturing, and avionics communications scheduling. It has handled scheduling problems involving 20,000 tasks and 140,000 constraints, with interactive response times for schedule modification on the order of a few seconds on a SPARC10.

Scheduler Requirements

Detailed scheduler requirements were initially established for the DADS application, then extended and adapted to encompass the scheduling needs of other NASA programs based upon feedback from the IIFS. The following paragraphs summarize requirements at a high level. They confirm the need to be appropriate to the application domain, to be compatible with the target system, and to provide responsive performance reliably.

Domain Appropriate - Commercial scheduling tools sacrifice domain relevance to extend their range of applicability, and hence their marketability. They often lack the capacity to efficiently handle the precise scheduling needs of large, complex applications such as those presented by EOS. In order to select or define a scheduling tool that is domain appropriate, application-driven requirements must be established. Whenever possible, these requirements should be based on multiple examples of domain operations and scheduling functions using realistic data sets. They must include a quantitative demonstration so that capacity and performance goals can be met simultaneously.

Since the GSFC V0 DADS is being developed concurrently with the prototype scheduler, we were careful to maintain a high degree of generality in the scheduler implementation. By first building a core scheduling capability derived from our Kronos scheduling engine, and then extending that capability through specialization, we were able to meet the specific needs of DADS while providing a scheduling tool that can easily be applied to similar problem domains in EOS.

Stated as a system requirement, the scheduling core domain model must be compatible with objects and functions required by the target application. Further, its customization capabilities must support accurate modeling of every schedule and relevant aspect of the domain. Care should be taken to ensure that this model reflects the intended scheduling policies and procedures of the application, and not the characteristics of analytical models used to project system performance.

Details of the scheduling core domain model are described in the Domain Model section. For the prototype scheduler, subclasses were created to capture application specific attributes and relationships. These attributes may be used to carry system data through the schedule or to support performance monitoring and analysis.

By creating persistent requirement and persistent resource profile classes as subclasses of the requirement class and resource profile class using an object-oriented model, respectively, we were able to provide the necessary scheduler functionality with a minimum of disruption. Persistent requirements have the option of specifying that they begin, use, or end with their associated activity. This allows the resource allocation to be open ended if desired.

To be effective, any tool must be functionally complete and be able to solve the problems for which it is applied. A scheduler must enforce structural constraints (i.e., predecessor-successor and parent-child relationships), temporal constraints (e.g., earliest start or deadline), and resource availability constraints while carrying out the desired scheduling and resource allocation policies in an automated fashion. In the prototype scheduler, policies are currently encoded as functions and a domain-specific algorithm (as described in the Scheduling Policy section.

We plan to eventually excise policy details from the scheduler by defining syntax for policy specification. One possible solution would be to utilize a rule- or knowledge-based approach to represent the numerous institutional policies. The major advantage of this approach is that rules (e.g., if-then statements) can naturally represent situations when a particular schedule is "good". Likewise, a dependence on rules allows for the incorporation of several knowledge acquisition tools. In the IIFS, for example, the Advice Taker/Inquirer (AT/I) allows users to enter and modify expertise in lucid forms such as natural language. Should a policy change, a tool like the AT/I could be used to quickly modify the appropriate rule governing that policy.

Compatible - The scheduling tool described here is designed be integrated as a functional component into the target application system. It cannot dictate requirements to that system, rather, it must adapt to the physical and logical demands of the encompassing system. The scheduler must execute on available hardware running the specified operating system. It must be able to communicate with asynchronous functional modules of application system via standard interprocess communication system facilities.

The scheduler must also be linguistically compatible with the surrounding system. It must be able to interpret and respond appropriately to requests for service and information. The prototype scheduler meets this requirement in several ways. The scheduler includes an API customized to the syntactic and semantic needs of the DADS modules with which it interacts. An underlying set of basic API functions facilitates this customization.

The scheduler supports the notion of activity state. The exact states and legal state transitions are defined for the application. In DADS, activities can be scheduled, committed, dispatched, executing, complete, or failed. Additional states and even additional state dimensions can be added as the need arises.

Responsive - Performance is often a critical requirement, but it is frequently overlooked in scheduling. There are often assumptions that scheduling will be performed once in an initial scheduling effort and that the resulting schedule will satisfactorily describe the actual execution of activities. This view is seldom correct and certainly incorrect in data processing scheduling.

We have segregated the total problem into two phases, planning (what to do) and scheduling (when to do it). In other words, planners are allowed to substitute similar tasks in order to find a set of tasks that have feasible schedules. Schedulers per se are given a fixed set of tasks and only leeway in the selection of resources and start/end times. Unlike the DADS and for that matter, the rest of EOSDIS, the IIFS utilizes the

planning/scheduling approach to generate browse products in lieu of standard products when computational constraints are too great for standard product generation. For example, computationally cheaper, yet less accurate tasks can be intelligently substituted for expensive tasks in order to better meet deadlines or minimize resources. This situation occurs often in image processing where resampling routines can reduce the image size. The browse products can be used by users to decide whether to initiate a standing order request. In this way, just as it was one of the first systems to suggest object-oriented programming and databases for the EOSDIS domain, the IIFS has allowed for the testing of risky, new ideas that may not yet have been considered within the operational DAACs.

Nevertheless, by making this distinction, we have not only, made each aspect more manageable, but we can tailor the functionality and performance of each component's implementation to the needs of the application. Planning typically occurs before scheduling, though replanning may become necessary. In the GSFC V0 DADS application, there is a small set of functions to be performed (e.g., ingestion, distribution). These can possibly be pre-planned in advance and described to the scheduler as tasks (with subtasks).

The scheduler must, on demand and in near real time, fit each new instance of a task into the current schedule in accordance with task priorities and deadlines while ensuring that necessary resources will be available. As actual events occur in the execution of the scheduler, it must rapidly reschedule to reflect the impact of the event. It must provide data to support graphic presentation of the current schedule, and even allow operator manipulation of tasks.

Reliable - The fault tolerance approach employed by the target application must be supported by the scheduler. In the GSFC V0 DADS this translates to requirements for redundant archiving of schedule information and rapid recovery of the schedule after a failure. The prototype scheduler does not fully include these features at present. However, basic mechanisms needed for reload are present in the script processor described in the Prototype Environment section. Also, previous schedulers based on the Kronos engine have included schedule storage and reload capabilities.

Prototype Environment

The DADS V0 Scheduler is being developed concurrently with the GSFC V0 DADS. Consequently, a stand-alone environment was needed in which to test and demonstrate scheduler functionality. The operation of components external to the scheduler was simulated via a script processor as shown in Figure 3. The script processor is controlled from a demonstration Graphical User Interface (GUI) that displays schedule activities and resource utilization profiles. Snapshots of the demonstration GUI screen may be seen in Figures 6 and 7. The GUI supports selection and execution of an event script which the script processor translates into API commands that it sends to the scheduler.

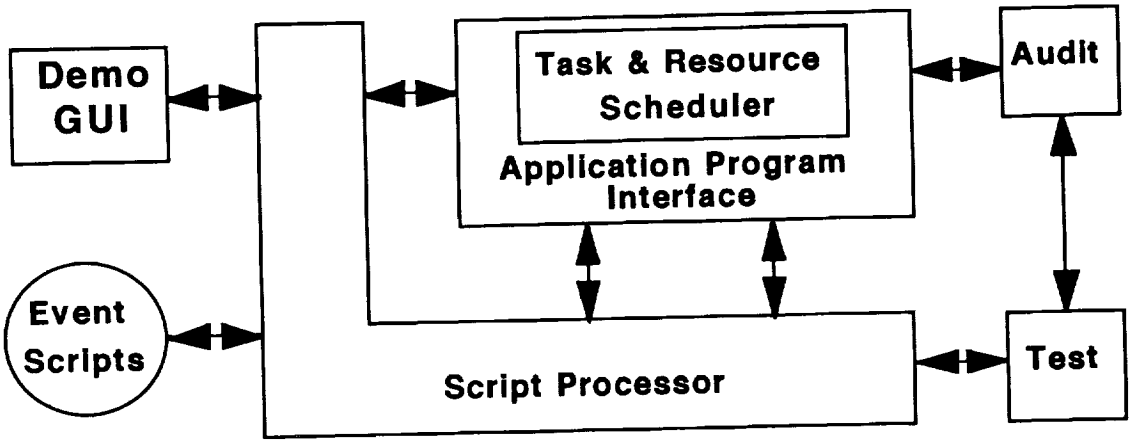


Figure 3. The Prototype System Architecture

A typical script initializes the scheduler by describing the resources available for scheduling, commands the creation of activities to be scheduled, and simulates execution events such as completion of execution. The script also notifies the GUI as objects to be displayed are created.

Graphical presentation of scheduler operation is visually convincing, but it is inconvenient for testing and benchmarking purposes. Recently, auditing and test functions were added to facilitate execution and validation of complex event scripts. The test function automates the execution of scripts and the invocation of the audit function, which checks the schedule for consistency and correctness.

Architecture of the Scheduler

The internal architecture of the scheduler is depicted in Figure 4. The base layer supplies basic temporal reasoning capability. This includes objects such as uncertain time-points and constraints, and functions for updating and querying the temporal knowledge base.

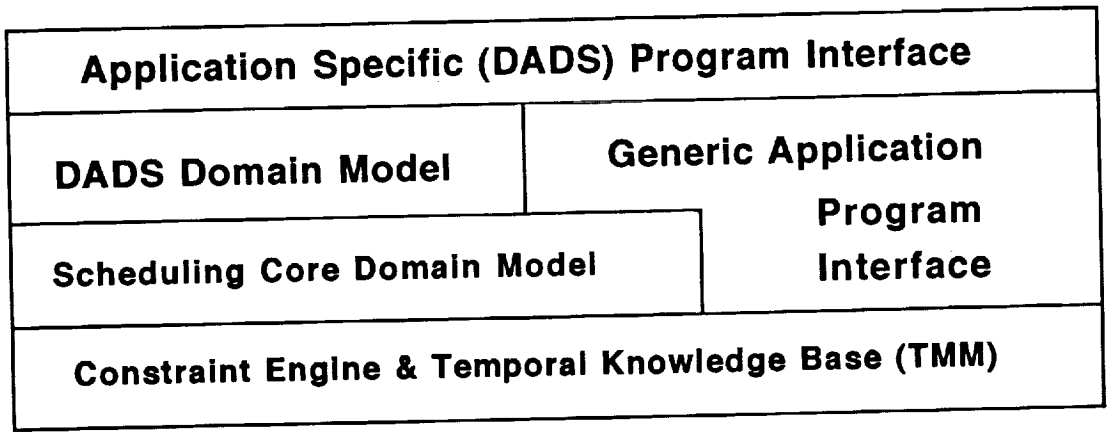


Figure 4. The Architecture of the Scheduler

The Scheduling Core Domain Model supplies the basic objects and functions needed for scheduling and resource management. Combined with the Generic API, these layers form a core scheduling capability that can be applied to various scheduling domains. In the DADS V0 Scheduler implementation, the base domain model was extended through specialization and extension to provide appropriate domain-specific capabilities, shown in the figure as the DADS Domain Model and the DADS API.

Domain Model

Key object classes of the scheduling core domain model include resources, requirements, activities and hierarchical activities. These are shown in Figure 5. along with related objects classes of the DADS scheduling domain model.

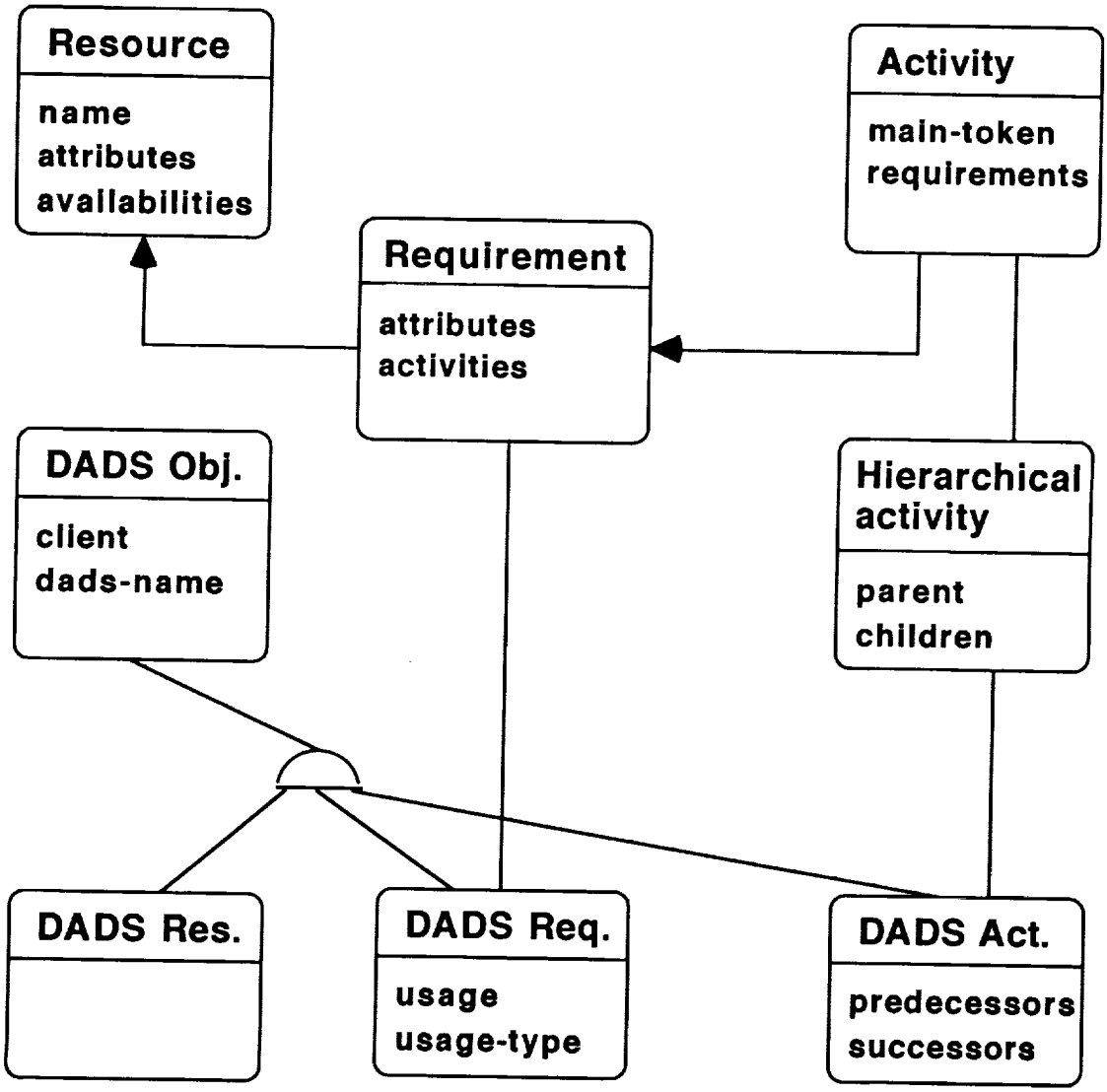


Figure 5. Key DADS Scheduling Object Classes

An activity represents an action to be scheduled. Each activity has an associated main-token which defines its end points in time and its possible duration range. An activity may be linked to multiple resource requirements. These abstractly define attributes that must be satisfied by the resources allocated to the activity. A subclass of the activity allows hierarchical activity structures to be defined. These were used in the DADS scheduler to implement tasks with component subtasks.

As an example, in the DADS application, a data ingestion task will have several subtasks. The data buffering subtask requires access to the FDDI network and a specific amount of space on one of the data ingestion magnetic disks. A subsequent archiving subtask requires access to the data on buffer disk and space on the UNITREE archive magnetic disk.

The core resource classes allow resources to be conceptually organized into pools using a hierarchical name structure (which permits wildcards) and using a list of resource attributes. Each resource has an associated availability that defines the maximum quantity of that resource and its temporal range.

Specialization of the core object classes extend the hierarchy to include characteristics of the target domain. In the DADS scheduler these specializations share a common parent class, the DADS object, which defines attributes every DADS activity, resource requirement, or resource must have. Only the client and dads-name attributes are shown in the figure.

Application Program Interface (API)

The Application Program Interface was specified formally by documenting data content (i.e. fields and forms) of the primary information components (i.e. tasks, subtasks, resources, etc.) exchanged between the scheduler and DADS subsystems. For each command, the documentation details the participants in the exchange utilizing the command, the conditions under which the command occurs, the intent (semantics) of the command, and the scheduler's response to the command under both normal and error conditions.

The following command categories describe the functions of the scheduler visible via the API. The categories have been intentionally kept rather abstract and high level here. Not all command categories have been fully implemented in the prototype scheduler.

Definition/Instantiation - Inform the scheduler of the existence of scheduling entities such as activities (i.e. tasks and subtasks), resources, and abstract resource utilization requirements. These commands do not cause scheduling to occur.

Modification - Change the specifics of information known to the scheduler. This category encompasses only changes to the scheduling problem (e.g. relaxation of a deadline). It does not include notification of real-world execution events.

Interrogation/Retrieval - Retrieve schedule and resource allocation information from the scheduler. This information is based on the scheduler's model of the problem space, its record of past events, and its projection of future events including resource utilization.

Scheduling/Rescheduling - Compute a new schedule with resource allocations. Commands in this category may be invoked indirectly by commands in the Update/Synchronization

category. Update/Synchronization - Inform the scheduler of the occurrence of real-world events (e.g. activity execution completion) which may affect the schedule. This category also includes commands for the transfer of responsibility for an activity from the scheduler to another subsystem (e.g., an execution monitor or dispatcher).

Notification - Inform another subsystem that a problem (or potential problem) has been detected by the scheduler.

Communication Handshaking - Provide positive acknowledgment of information transfer.

Fault-Tolerance/Recovery - Support for information backup and recovery from failures.

Scheduling Policy

The operation of the scheduler is controlled by scheduling policies. These are currently captured in domain-specific, hard-wired algorithms for resource assignment and activity scheduling.

The baseline resource assignment and scheduling algorithm is:

For each activity to be scheduled:

If the activity has component activities,
Schedule each of its component activities (i.e., apply this algorithm recursively).

If the activity is scheduleable,
For each resource requirement of this activity:

- If a satisfactory resource is available for use without causing it to be oversubscribed,
assign that resource to meet the requirement.
Availability implies that the resource is part of the resource pool specified in the resource requirement and has the attributes specified in the resource requirement.

-If no satisfactory resource is available,
apply the following stratagems in sequential order,
using the possible resources until one of them successfully eliminates the oversubscription:

* Constrain the order of activities involved in the oversubscription:
Individually before the activity, or
Individually after the activity, or
Collectively before the activity, or
Collectively after the activity.

* Relax the deadline of activities involved in the oversubscription and constrain the order of activities (as above)

* Constrain the order of parent activities of the activities involved in the oversubscription (as above)

* Report failure [and Exit]

If the activity is still scheduleable
and all component activities of this activity have been scheduled,
Mark the activity scheduled.

Then update:

The schedule's temporal knowledge base,

The time bounds of all changed resource utilization profiles.

One thing to notice in the algorithm is the emergence of situations to control the scheduling. For example, take the situation where the scheduler should schedule activities if their resources won't possibly be oversubscribed. This was a DADS requirement that other domains need not be constrained to have. But, in its current incarnation, it is hard-wired into the algorithm. Should this change, then the algorithm must be modified, increasing scheduler maintenance costs. As new policies are incorporated, these costs will be untenable. Hence, changing over to other approaches such as rule-bases, will constrain costs and allow for evolvability.

Scheduling Example

The operation of the prototype scheduler is revealed in Figures 6 and 7. In this simple example, seven data ingestion tasks have been scheduled. Each task contains four subtasks (not visible) and is represented in the display as a horizontal timeline. The solid portion of the timeline indicates the earliest possible execution of the task. The dashed portion of the timeline indicates scheduling flexibility between earliest execution and the task's completion deadline.

At the bottom of the display, the resource utilization of a selected resource is shown. The black profile line indicates expected resource utilization if all tasks execute as early as possible. The gray profile line indicates possible resource utilization.

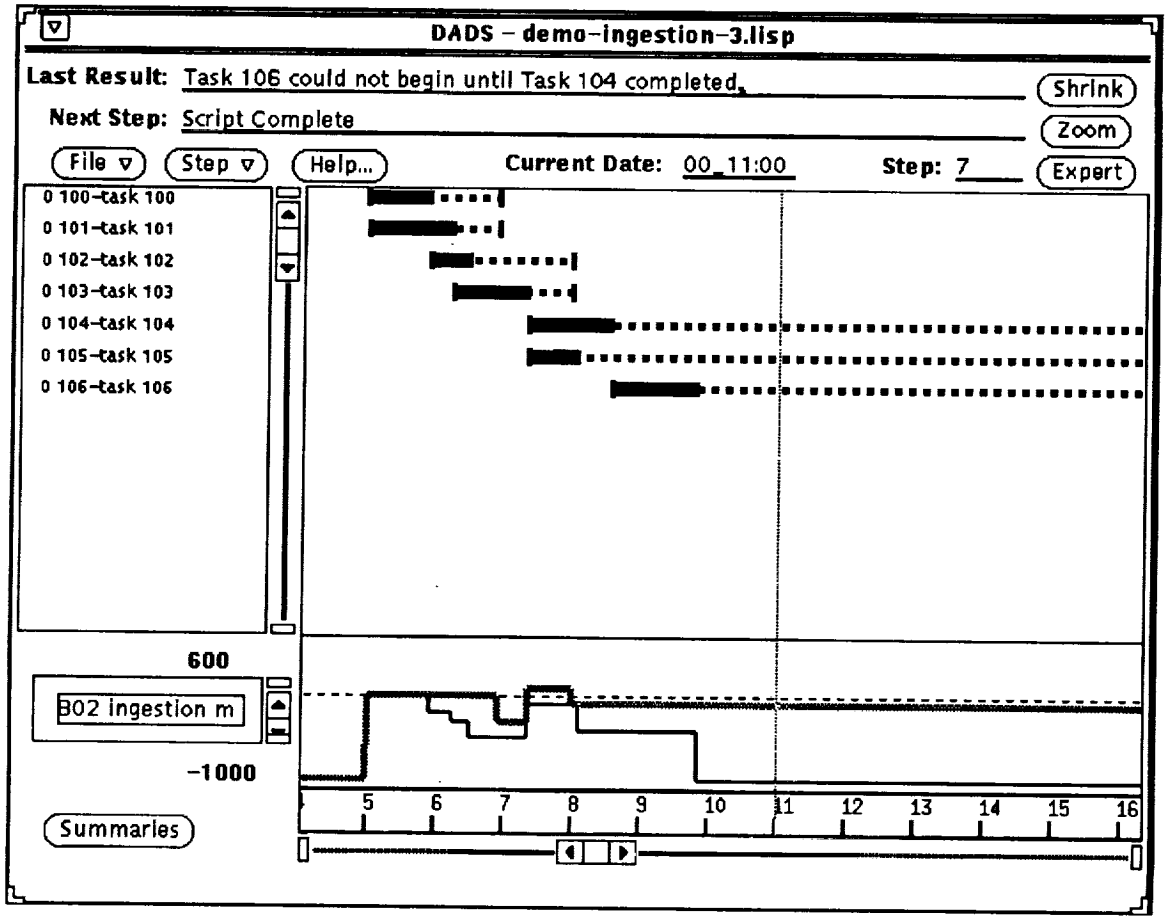


Figure 6. Simulation of the Baseline DADS V0 Scheduling Approach (FIFO Queue)

In this figure, the tasks have been configured to simulate the baseline DADS V0 scheduling approach. In the baseline approach, all resources needed by the component subtasks are allocated to the task. Then tasks are then scheduled using a First-In First-Out (FIFO) Queue. Additional constraints were added to enforce this queuing.

Parallel task execution occurs until resource utilization reaches 100%. The subsequent tasks must wait for ongoing tasks to complete.

The deadlines of tasks 104 through 106 could not be met. These deadlines were removed, causing the dashed portion of the timeline of these tasks to extend to infinity. Task 106 actually started AFTER it's completion deadline.

The resulting resource utilization is very inefficient. It has large regions of rather low utilization an is spread over almost five hours.

Figure 7. Shows the same tasks scheduled using the full power of the HTRS scheduler.

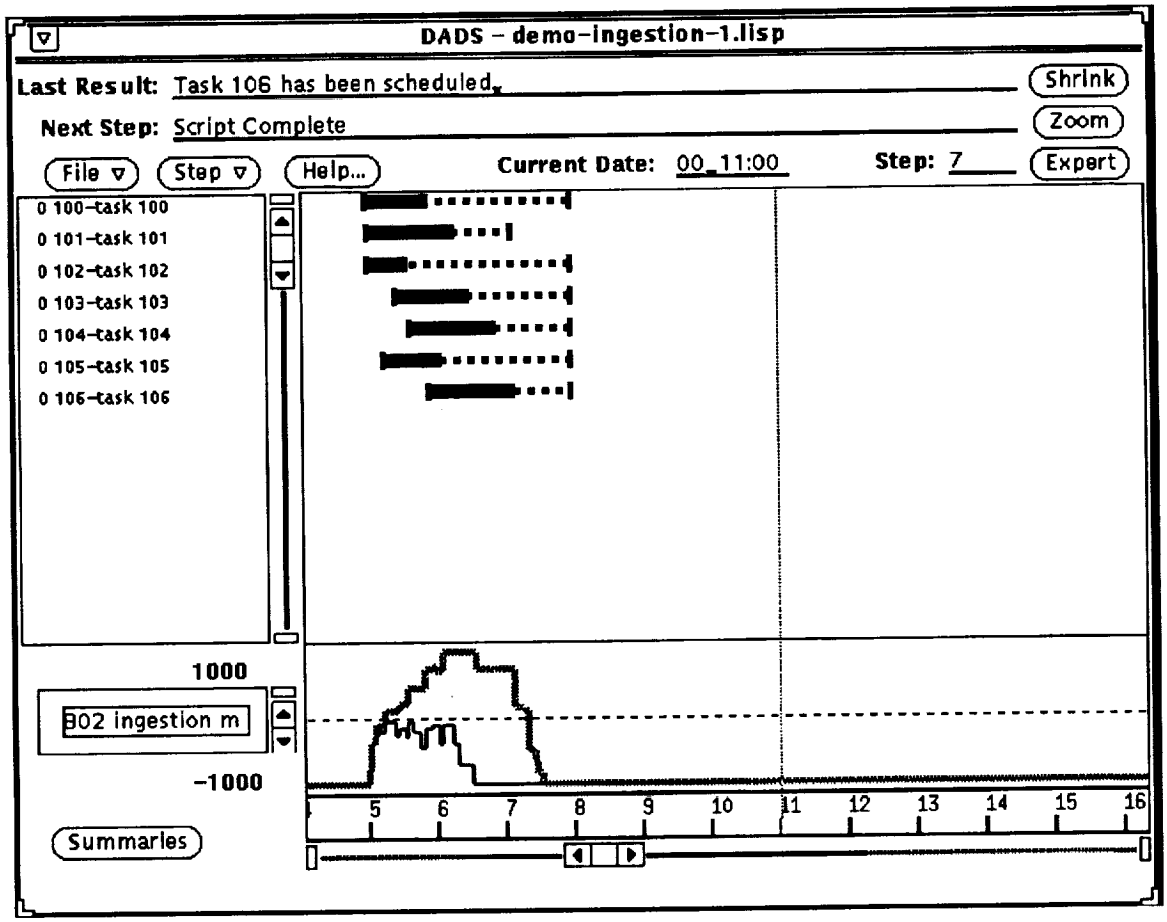


Figure 7. Efficient Automated Scheduling and Resource Allocation

Resource requirements were specified with respect to individual subtasks, and FIFO queuing constraints were not imposed. The resulting schedule is clearly superior.

All tasks were scheduled within their deadlines. The scheduler has optimized resource utilization (as evidenced by the compact profile). And the entire group of tasks requires only slightly more than one hour. This leaves time for an opportunistic system to initiate several tasks such as trend analysis routines, maintenance tasks for adding new system components, or quicker service. Moreover, using less powerful hardware could lengthen the overall duration, but still beat the performance of the FIFO approach.

This simple example shows that a constraint-based task and resource schedule can provide substantial improvements in system performance over simple queuing schemes. What may not be evident are the benefits it provides through rapid rescheduling in response to unexpected events (e.g., resource failures), and through the automation of complex scheduling policies. These performance improvements could translate into cost savings through the use of less expensive hardware.

Conclusion

The HTRS scheduler prototype has been successfully developed for the GSFC DAAC. Special attention should be paid in the identification of the scheduler requirements and the DADS domain model. Even though the prototype is still in its initial phase and it has not yet been integrated into the DADS, the effort has been very informative. In particular, it was demonstrated using realistic examples of DADS activities that a FIFO queue algorithm can be extremely inefficient under certain conditions, and can drastically reduce the overall performance of the DADS. The scheduler cannot only make better use of limited resources and prevent a backlog of data, but it can also provide valuable information about resource bottlenecks and performance characteristics. The next challenge will be to integrate HTRS in the DADS , monitor its performance, and evaluate its benefits when running in a real operational system such as the GSFC DAAC.

In the context of mass storage systems, scheduling can help ensure that timely service is provided to users who expect a lot from these expensive computing facilities. Likewise, scheduling can be used as a simulation tool to predict the performance from adding particular hardware. By utilizing the same scheduling environment, these simulations can be based on real information from the operating environment and can provide quality information for decision makers. In some cases, decision makers may avoid costly hardware purchases by tweaking the scheduling policy algorithm. Hence, the scheduling policy algorithm must be flexible enough to be modified quickly in order to contain software maintenance costs. Certainly, the use of scheduling will provide better service for users, faster processing throughput, and cheaper costs.

In fact, many of the scheduling issues presented here have arisen throughout numerous NASA applications. Over the years, the X-OTP has provided scheduling expertise to various projects by focusing on rapid prototyping of new technologies for mitigation of risk, technology transfer through continued software development from prototypes, and reduction in cost through software reuse of generic tools. By working with Honeywell Technology Center, X-OTP is further reducing software development costs by providing difficult requirements to companies, who can then apply developed techniques to other commercial domains such as aviation communications scheduling. By helping companies expand into new markets, NASA, without incurring high maintenance costs, increases the likelihood that dual-use commercial software will survive over the lifetime of lengthy projects such as EOS.

X-OTP, on the other hand, requires feedback from projects whose requirements push the state-of-the-art. As intended, the GSFC DAAC, through Hughes-STX, has provided this feedback before the larger EOSDIS has gone into operational use. The GSFC DAAC, however, is an operational system that cannot be interrupted with technology that is too risky. Hence, prototypes such as the IIFS can quickly test very risky technology in an end-to-end framework without adversely affecting operations. For one thing, the IIFS was the first system at GSFC to suggest the use of object-oriented databases for the EOS domain. Likewise, the IIFS was the first system to suggest the use of neural networks for classifying remote sensing data -- a technique that is now widely accepted in remote sensing circles. And, finally, the use of this particular scheduling software was based upon a NASA internal R&D project (i.e., Directors Discretionary Fund) entitled "Near real-time generation of Browse Products" and incorporated into the IIFS. Because of the development of the IIFS and the close proximity to NASA projects, the Information Science and Technology Branch has provided in-house expertise regarding emerging technologies such as these. Moreover, in addition to applied research, the branch has developed one of the DAACs operational quality assurance routines for the TOVS

pathfinder data sets. Likewise, the Space Data and Computing Division, for which the Information Science and Technology Branch is a part of, is currently GSFC's only supercomputing facility with an extremely large mass storage system (over 20 Terabits); this enables feedback regarding technology integration of large, expensive systems. All in all, elaborate collaborations such as these will obviously be required to evolve one of the most ambitious engineering and information system projects, or namely, the Earth Observing System.

Acknowledgments

Primary funding for this scheduling project has been provided by the Code X - Operations Technology Program under Dr. Mel Montemerlo, NASA-HQ. Funding has also been provided by the GSFC DAAC as well as Honeywell Technology Center. Special thanks are extended to members of the Information Science and Technology Branch, who have provided invaluable feedback through the use of the Intelligent Information Fusion System.

1. L. Bodden, P. Pease, JJ. Bedet, W. Rosen: Goddard Space Flight Center Version 0 Distributed Active Archive Center. In Third Conference on Mass Storage Systems and Technologies. NASA CP-3262, 1993, pp. 447-453.

2. H. El-Rewini, T. Lewis, H. Ali, Task Scheduling in Parallel and Distributed Systems. PTR Prentice Hall, Englewood Cliffs, New Jersey, co. 1994

A Comparison of Rotary- and Stationary-Head Tape Recorders

John R. Watkinson
'Resurgam', 2 Hillside
Run Length Limited, Burghfield Common RG7 3BQ, U.K.
Phone: +44-734-834285

Abstract
Digital recording may take advantage of many types of media, but usually a preferred type of drive or transport emerges for each. In magnetic tape recording, two approaches have emerged in which essentially the same medium is tracked in two radically different ways. This paper compares the characteristics of Rotary- and Stationary-Head transports in an attempt to establish which approach might be considered for a given application. The conclusion is that in many cases there is no obvious choice based on recording physics and that often the choice will be made on the experiential knowledge of the designer.

The Limits of Tape Recording

This paper restricts itself to digital recording, but in practice a tape transport does not know the meaning of waveforms passing through its heads and media. These waveforms experience an analog channel which has suboptimal frequency response as well as non-linearities and various noise mechanisms. It is the discrete decision-making process of the replay data separator which renders the entire machine digital. The channel coder in the record section merely produces waveforms which are advantageous to a discrete data separator. The impairments of the real channel result in a certain error rate distribution and a suitable error correction strategy will be employed in order to meet the residual BER demanded by the application.

Assuming an acceptable BER, tape-based data recorders are measured by the following primary parameters: Unit cost, maintenance cost, cost per bit stored, transport and medium size and weight, access time and transfer rate. Secondary parameters are figures which are only critical in certain applications. These include environmental tolerance, power consumption, speed range, startup time and so on. The storage density emerges as a critical factor as an improvement allows the same job to be performed with a smaller, lighter machine at a lower cost per bit. Storage density improves by paying attention to three dimensions. Thinner tape allows a greater surface area in a given volume and allows better conformity with the head. It does, however, require a substrate with higher tensile strength and more precision in the tension control system of the transport. A reduced track pitch increases density but requires a more accurate track-following mechanism, improved means to reject crosstalk and a higher output medium to restore the noise performance. Reduced bit-length along the track is the third dimension and demands heads with smaller gaps, better head/tape contact, higher output media and channel codes with improved figures of merit. In general an increase in density will raise the raw BER and require a more powerful error correction strategy.

An improvement in superficial (or areal) density reduces access time as a shorter tape holds the same data. Smaller reels have lower inertia and withstand harsh acceleration environments better. The linear (along-track) density is primarily determined by magnetics and coding, whereas the cross-track density is primarily limited by tracking accuracy which is mechanically determined.

Mechanical Considerations

The above criteria can now be examined from the alternative viewpoints of stationary and rotary head implementations. [Reference 1].

As tracks of the order of 10 micrometers wide are in use today, clearly a single track tape is a mechanical impossibility. A stationary head recorder will as a practical matter need to use a significant number of parallel tracks across the tape and the bit rate will be divided between them. These tracks will be recorded and played by multi-track head stacks. High density machines will need active track following mechanisms physically to move the headstack in compensation for tape weave, typically using piezo-electric or magnetostrictive "muscles".

The track width and pitch are fixed and are determined by the head design. Crosstalk in the form of mutual inductance may take place between the various magnetic circuits in the headstack and this must be controlled by the introduction of spaces and/or shields between the magnetic circuits which result in guard bands between the tracks. Photolithograpically produced heads are better from the standpoint of mutual inductance because of their flat construction, but spaces between the tracks are still inevitable because of the need to arrange windings around the poles.

As an alternative interleaved headstacks may be made in which only one in N tape tracks is furnished with a magnetic circuit. Depending on the bit rate required, the transport may have N headstacks or may transport the tape N times through the machine in a serpentine fashion, indexing the headstack to one of N places on each pass. If N headstacks are used, each must have its own track following actuator. A serpentine recorder needs only one headstack actuator, but its travel will be much longer.

To record the same bit rate, the rotary head recorder produces a large number of slant tracks by the rapid rotation of a small number of heads. These tracks can be nearly perpendicular to the tape motion in transverse scan recorders or nearly parallel to the tape motion in helical scan recorders. In both cases the tracking mechanism relies upon the cross-track component of tape linear motion which can thus be controlled by capstan phase. The track pitch is controlled by the linear tape speed and is independent of the dimensions of the head. It is possible to have a machine which supports more than one track pitch so that, for example, tapes of various coercivities can be used. The heads on a rotary scanner are not in a stack, but are distributed around the periphery so that mutual inductance effects are negligible. Rotary transformers are required to couple the rotating signals with the stationary signal processing circuitry and these will be prone to crosstalk, although modern multi-head machines have rotating pre-amplifier circuitry so that the transformers do not handle signals direct from the replay heads.

We can say that a rotary head recorder is no more than a mechanical multiplexer which lays down tracks rapidly with few heads whereas a stationary head recorder lays the tracks down slowly with many heads. We could well use the analogy of serial and parallel transmission. Both approaches require active track following at high density. The stationary head transport does this by moving the head whereas the rotary head machine moves the tape. Thus the common criticism that rotary head machines are complex is not so strong at high densities where a stationary head transport needs a track following servo, or with interleaved heads, several servos.

Capstan phase control in rotary head recorders cannot accommodate track straightness errors due to, for example, interchange inaccuracies. In this case, the rotating heads may have an additional track following mechanism which allows the heads to be deflected along

the scanner axis (i.e transversely with respect to the tape track) as the scanner rotates. In this case geometric errors within the track can be compensated. In a rotary head recorder the head/tape interface is complex. The revolving scanner builds up an air film and the film thickness stabilizes when the tape tension balances the air pressure. This is one reason why tape tension is critical in rotary head recorders. As a result of the air film the scanner itself does not touch the tape and friction around it is very low. The head pole must project out of the scanner by a distance equal to the film thickness plus an amount needed to deform the tape to give the required contact pressure. The traveling deformity in the tape results in acoustic noise which may need attenuation in some applications. The linear speed of the head with respect to the tape must be kept below the propagation speed in the tape to avoid the creation of mechanical shock waves which result in rapid wear. At the relative speeds involved, there is appreciable aerodynamic lift attempting to separate the head and the tape. The conditions are in a region midway between the firm contact of a slow speed stationary head tape and the non-contact system of a hard disk. The wear reducing properties of the lift can be balanced against separation loss. In practice head wear is greater on new tapes where asperities are linished by the heads. Older tapes show reduced head wear and error rates.

Head Design

Naturally rotary heads experience high frequencies and the magnetic circuit must be constructed is such a way that eddy current losses are minimized. Ferrite is non conductive but saturates before today's high coercivity tapes can be fully modulated. Metal pole tips can be fitted to ferrite bodies, or lamination or sintering can be used to raise head resistivity. A single head may reach 200 megabits/second, but in practice lower figures are used for other reasons such as the need to distribute data over several heads to give resistance to clogging or to reduce the frequency at which the associated circuitry must operate. Whilst the reading speed has no effect on most types of noise, raising the speed does increase the replay signal in proportion and so the effect of head and preamplifier noise is reduced.

Conversely inductive heads are at a disadvantage at low relative speeds. For a given bit rate, stationary head, or parallel recording, implies low frequencies where magneto-resistive heads with their non-derivative action give a noise advantage. There is a changeover at approximately one megabit per second where the two types are roughly equal in performance. Thus in general rotary head recorders use inductive heads where eddy current losses are a concern whereas stationary head recorders will use magneto-resistive heads where eddy current losses are insignificant unless prodigious bit rates are envisaged.

Apart from the requirement for the appropriate magnetic orientation for transverse scan, there is little difference in the magnetics between tape intended for rotary and for stationary head machines. Therefore any development in tape technology is available to both. Similarly developments in channel coding and signal processing are also available to both. Rotary head recorders require smaller DC components in channel codes due to the presence of the rotary transformers, although in practice both types of machine have been seen with the same channel code. Similarly techniques such as partial response are equally applicable.

In error correction, stationary head recorders see defects in several tracks simultaneously and need to interleave by distributing codewords over several heads to restrict their impact. Rotary head recorders to an extent interleave mechanically as a circular dropout appears as a spaced out series of defects in successive head scans.

One significant difference between serial and parallel recording is that the rotary head recorder is naturally complemented by the adoption of azimuth recording. Rotary head transports have a small number of heads and these are spaced apart physically. It is easy to make such heads (minimum two) with alternating azimuth angles. If a suitable channel code is employed to restrict the ratio of maximum and minimum wavelengths, erasure by overwrite can be employed. Not only does this eliminate the erase head, it allows the track width to be determined by the tape speed and not the head design. If the poles of the record head are made wider than the track pitch, part of the side of a given track will be erased by the next track to be written. Azimuth effect allows replay heads to read these adjacent tracks despite the lack of a guard band. As a result azimuth recording has come to be synonymous with the term guard-band-less recording.

Tapes of different coercivity can be handled by choosing an appropriate track width and driving the tape at a suitable linear speed. This approach is used in RDAT which can operate with 13 micrometer tracks on metal particle tape but which uses 20 micrometer tracks on barium ferrite tape which is required for contact duplication.

In principle certain aspects of azimuth recording can be used with stationary heads, and such devices are known if uncommon. In one implementation, two interleaved headstacks are used, one of each azimuth type. The first head writes tracks which are oversized, and the second writes tracks of the correct width between the others, side trimming them to size by overwrite. In practice it is difficult to fabricate multitrack heads with azimuth angles other than 90 degrees. This applies to conventional heads as well as those which are fabricated using photolithography.

Variable Speed

The different transport designs react differently to the requirement to operate at variable speed. It is necessary to be quite precise about the kind of variable speed operation being considered. In instrumentation, variable speed operation implies that the timebase of the recording and reproduce processes is different, but all of the data are fully recovered. This allows, for example, the high data rate of a practical experiment to be recorded as it occurs, but reproduced at a rate appropriate for the analysis process, which may well have a restriction such as the I/O speed of a computer. A linescan recording from a high speed airplane may be studied at leisure on a display.

On the other hand the requirement in a digital video recorder is only that a recognizable picture shall be available at non-standard speeds, and so a great deal of data can be lost.

In computation, the transfer rate of a given drive is usually fixed, but a variety of drives may be available, at different costs, which can offer different transfer rates on a common interchange medium.

In a stationary head recorder, the data rate from the heads is directly proportional to the tape speed. If a variable bit rate is required, then changing the tape speed will require a corresponding change in any record or reproduce equalization in every active track. In machines with a large number of tracks this becomes very complex. At high speeds the frequencies seen by the heads become very large. This precludes the use of stationary heads for production (as opposed to consumer) video recording. Although normal speed operation is perfectly feasible, high shuttle speeds (100x - 200x) cannot produce pictures. Rotary head recorders are not capable of operating over a wide range of transfer rates where no data are lost. This is because the transport aerodynamics must be optimized for

one speed. Changing the transfer rate requires the scanner and capstan to change speed by the same amount and this results in a significant change to the pumping effect of the scanner, with consequent changes to the air film thickness and tip penetration.
The helical scan recorder is advantageous for digital video recording because the tracks are nearly parallel to the tape edge. When the tape is driven linearly at the wrong speed, the scanner speed is not changed in proportion and so the tracking breaks down. Despite that it is possible to recover around 40% of data as heads cross tracks at a grazing angle. Provided the track crossing angle is sufficiently shallow, sync blocks on the track can be recovered and if they are uniquely addressed the data can be used to update a frame store. A further advantage of helical scan is that the head to tape speed is dominated by the scanner speed. As a result it is possible to maintain a reasonably constant head to tape speed over a wide linear tape speed range simply by modulating the scanner speed. The result is that the replay electronics will see constant frequencies and their data separators will operate normally.

The short tracks of the transverse scan recorder are almost at right angles to the tape motion and as a result the length of track recovered at shuttle speed is too small to allow sync blocks to be recovered. Thus the transverse scan machine is at a disadvantage for the production video recorder market where pictures in shuttle are mandatory.

Video recorders fitted with deflecting heads are capable of following entire tracks over a range of speeds typically from -1 to +3. When a helical scan transport records, the tape is wrapped around the scanner at the helix angle, which is determined by the construction of the scanner. However, the tape is moving as the scanner rotates, and the result is that the track angle differs from the helix angle. Thus variations in tape linear speed affect the effective track angle, and the head must be deflected by a ramp waveform to follow. The steepness of the ramp is proportional to the deviation from normal speed. It is possible to use the head deflection mechanism of a rotary head recorder to increase the proportion of data recovered during shuttle.

In instrumentation recording, incremental operation is becoming popular. In an incremental recorder, the transport and data channel are optimized for a single data rate, and all lower rates are implemented via a buffer memory which absorbs input data until the transport can run at speed and record a whole block. Similarly on replay data are output at any rate from the memory and the transport runs in bursts in order to keep the memory topped up. Incremental recording has been seen on stationary head, transverse and helical scan machines, but all are not equally suitable.

An bit rates near to maximum, the size of the memory is a function of the data rate and the time taken for the transport to change mode. At high density, it is not acceptable to leave IRGs (inter record gaps) in between increments. In practice, at the end of an increment the transport will cease writing, stop, reverse a short way and wait. On writing the next increment the transport will accelerate to speed (pre-roll) and read the end of its last increment so that the new data can be appended contiguously after the old in an assemble edit. This avoids the creation of a gap on the tape. However, the memory must be able to absorb virtually the full data rate for the time taken to reposition and pre-roll.

At bit rates well below maximum, the transport spends only a small proportion of time transferring data. The rest of the time it is idle or repositioning. An idle stationary head recorder does not suffer head wear as there is no relative motion. However, rotary head recorders must keep the scanner running in order to eliminate the lengthy acceleration period. There is thus a potential headwear problem which can only be avoided by unwrapping the tape from the scanner. The transport then enters a standby mode. The time

taken to go between standby and functional modes must be added to the reposition time and so determines the memory capacity required at low speeds.

In helical scan, unwrapping is a complex process which requires the operation of several moving guides and which takes an appreciable time. On the other hand a transverse scan rotary head transport can be unwrapped simply by retracting the single cupped guide which conforms the tape to the scanner. This can be done in milliseconds with a solenoid. As a result the transverse scan rotary head transport finds itself at an advantage in the incremental recording application as smaller buffer memory is needed.

Size

Size means different things in different applications. In some cases it is the size of the tape reel or cassette which is important, especially if it needs to be shipped. On the other hand it may be that the overall size of the recorder is restricted, for example in airborne installations.

Tape cassettes are advantageous in that they protect the tape well from handling damage and require little skill to insert in the transport. However, cassettes are at a disadvantage for shipping because they are volumetrically inefficient. A cassette contains two reels, but when one is full, the other must be empty. As a result, instrumentation users will sometimes choose to retain open reels when really large quantities of data must be shipped on tape.

Where overall size matters, the choice of cassette or open reel becomes irrelevant as two reels are needed by both. The stationary head transport can be made very compactly as the head block takes up very little space. In contrast the scanner and threading mechanism in a helical scan transport will take up appreciable space, often making the deck area double that of the cassette. The transverse scan design is appreciably more compact as the headwheel has a much smaller diameter and the axis of rotation is parallel to the tape. The threading mechanism is trivial and takes up little space.

Ruggedness

In harsh environments differences will be found between transport designs. The helical scan transport is most sensitive as the large scanner has appreciable inertia and can generate large precessive forces and timing errors in a mobile or airborne environment. It is also the most critical on tape tension as variations cause changes in air film thickness and also result in changes in track angle which may cause interchange problems. Humidity changes can also affect the track angle in helical scan whereas transverse and stationary head recordings are unaffected.

Abstractions

When a helical scan recording is played at the wrong speed, parts of the track are recovered, allowing typically 40% of the data to be recovered. If, however, the tape speed is normal, but the scanner is driven at around twice the correct speed, then full data recovery is possible. Sync blocks will be recovered non-sequentially, and block addressing will be used to return the data to its correct sequence in memory. Error correction restores

any sync blocks that are not recovered. This is the principle of the non-tracking (NT) rotary head recorder which clearly needs no adjustment for interchange.

In principle, an NT transport could play tapes having a variety of footprints provided the heads were of the appropriate azimuth angle and approximately the right width.

Non-Tracking is an attractive technology as instead of requiring increasing precision to allow narrower tracks, NT dispenses with the need for tracking altogether and, indeed depends upon severe mistracking to allow the sync blocks to be recovered in a statistical manner over several head sweeps. Thus an NT player can play tapes having a variety of track angles. If azimuth recording is used, tracks of various width can also be played. Following this argument further, it should be possible to play a stationary head multi-track recording using a NT helical scan transport. Provided the azimuth of the heads is appropriate, sync blocks can be recovered as the heads cross the tape tracks. Deflecting heads could be employed to increase the proportion of data recovered on each head sweep.

The converse argument is that, subject to details such as azimuth, a stationary head transport fitted with track following heads should in principle be able to play a helical scan tape by deflecting the heads at an appropriate speed to replicate the helical track angle. If several such heads are fitted, one can be resetting whilst another crosses the tape. This is a messy arrangement and is advanced only as an introduction to a better approach.

Magneto-optical readout has primarily been addressed to disk recording where the magneto-optic element is in the disk itself. It is, however possible to use magneto-optics to read conventional magnetic tape. This requires that the magneto-optic element is in the head. Briefly, a head is made having two poles and a narrow gap, but which is wide enough to span the entire width of the tape. A given track on the tape will cause the area of one of the poles above said track to follow the track magnetization. If polarized light is incident on the head pole, the reflected light will have that polarization rotated. A suitable analyzer can turn the rotation into an intensity variation which a sensor can detect. However, the sensor is a linear sensor which develops a one dimensional image of the cross-track magnetism. Any number or layout of tracks can be handled simply by sampling the cross track image at the appropriate points. If the sensor is, for example, a linear CCD element, the cross track image can be shifted out and analyzed in a software driven process. Thus track weave of a stationary head recording can be eliminated by shifting the sampling points in sympathy.

If, however, a helical scan recording is played, the slant tracks appear to continuously drift across the image. As one is lost at one edge of the head, another begins at the other edge. Again, subject to azimuth, it is possible to play a helical scan recording on a stationary head magneto-optic transport. Although the head is physical stationary, it has virtual movement by way of following the images of continuous slant tracks in the analysis process.

Thus in the limit, rotary head recorders can be made to play stationary head tapes and vice versa, suggesting that the techniques are not all that different. The rotary and stationary approaches are both complex as density rises, but the non-tracking principle may give the edge to the rotary transport in the future, with competition from magneto-optic replay in stationary head design.

To illustrate that nothing is new, in W.W.II a German fixed head audio tape recorder intended for dictation was fitted with a rotary playback head so that it could reproduce speech without pitch change over a wide range of linear tape speeds.

1. Watkinson, J.R. *The Art of Data Recording*, Chapters 7 and 8. Oxford: Focal Press (1994) ISBN 0 240 51309 6.

Client/Server Data Serving for High Performance Computing

Chris Wood
Maximum Strategy Incorporated
801 Buckeye Court, Milpitas CA. 95035
chrisw@maxstrat.com
408-383-1600
408-383-1616 (fax)

Abstract

This paper will attempt to examine the industry requirements for shared network data storage and sustained high speed (10's to 100's to thousands of megabytes per second) network data serving via the NFS and FTP protocol suite. It will discuss the current structural and architectural impediments to achieving these sorts of data rates cost effectively today on many general purpose servers and will describe an architecture and resulting product family that addresses these problems.

The sustained performance levels that were achieved in the lab will be shown as well as a discussion of early customer experiences utilizing both the HIPPI-IP and ATM OC3-IP network interfaces.

Introduction:

Back in the dark ages, about the time that touch-tone telephones were coming into vogue, computers were simple things that read in some data and a program, processed the data and spit out the answer. Data storage typically consisted of small amounts of core memory, card reader/punch machines and maybe a tape drive. Disk storage added the dimension of random access to data, but was typically directly attached to the central processor by any number of proprietary I/O schemes[1].

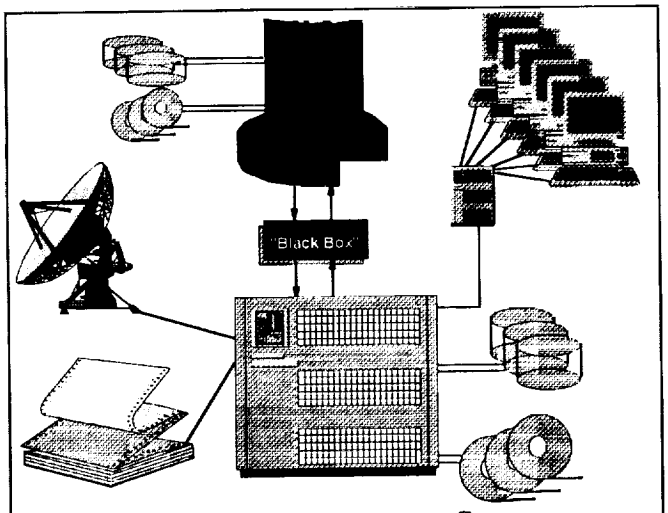


Figure 1: Traditional Supercomputer Center

Data sharing

Early customers in the high performance arena often owned several processors: one or two very large number-crunches and several smaller machines used to prepare the data for the large machine and/or print the output stream generated by the number crunchers[2]. A typical installation might have looked like that shown in Figure 1.

Sharing of data across computing platforms was typically done by copying data located on one processor to tape and reading it on another. Just attaching disk storage to multiple

processors did not address the problem since most processors utilized different physical and logical attachments. Even if two machines, by chance, could physically share a disk storage device, different machines wrote and read data in different ways and could not access each others data.

Some early solutions involved the use of black boxes that attempted to mate different interfaces and address data format incompatibilities. The large number of interfaces and file structures in use today (and growing!) tended to work against this type of an interconnect solution[3]. In an early, and very innovative, attempt to address the problem of incompatible file systems, the Los Alamos Lab's created the Common File System (CFS)[4] on an IBM mainframe base. Architecturally, CFS could be considered to be the first implementation of networked data serving. It addressed the issues of data sharing amongst heterogeneous hardware platforms, incompatible file systems (e.g. its name...Common File System) and the problem of incompatible physical I/O attachment schemes - although this was often accomplished via "black box" I/O mating hardware.

The ever growing complexity of the "black box" solution for heterogeneous platform interconnect ruled this method out as a long term answer to seamless data sharing. Alternatively, all vendors could adopt a common file system and I/O structure. This was thought unlikely.

Client/Server to the rescue?

A little over ten years ago, a group of UNIX architects at Sun Microsystems realized that the only way to address the data sharing problem (as well as data currency, consistency and access) was to remove the "ownership" of the data from the compute processor (the entity who processes the data) in a manner similar to that utilized by CFS and store it on an independent "server" who's only job is to store and retrieve that data when requested to do so by the compute processor (e.g. the "client"). Most importantly, they also defined a standard way to access the data that would be independent of any particular physical file system and physical interconnect. Thus was born the Network File System (NFS); the original foundation of client/server computing.

Speed limit: 5 MPH.

DEC talking to IBM, SGI communing with HP, The USSR making peace with the USA! All these things became true; unfortunately the Cold War lasted 50 years and that seems to be how long (in a relative sense) any self respecting high performance computer seems to have to wait for data from its "server" today. The convince of sharing data across many platforms comes at a price - speed. NFS (and essentially any exportable file system available today) is primarily hamstrung by three major bottlenecks:

1. The speed that the server's disk subsystem can deliver data to the NFS server, and
2. The speed that the NFS server can process this data through its own file system and encapsulate (packetize) this data with the UDP and IP network protocols and deliver it to the network fabric, and
3. The finite usable speed of the fabric[5].

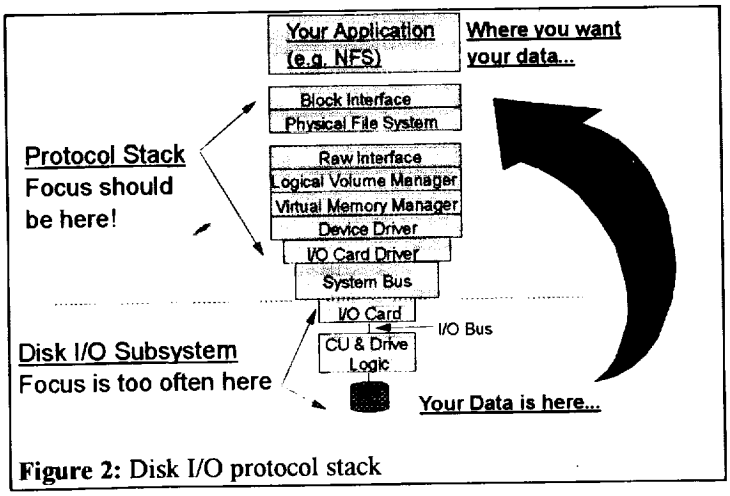


Figure 2: Disk I/O protocol stack

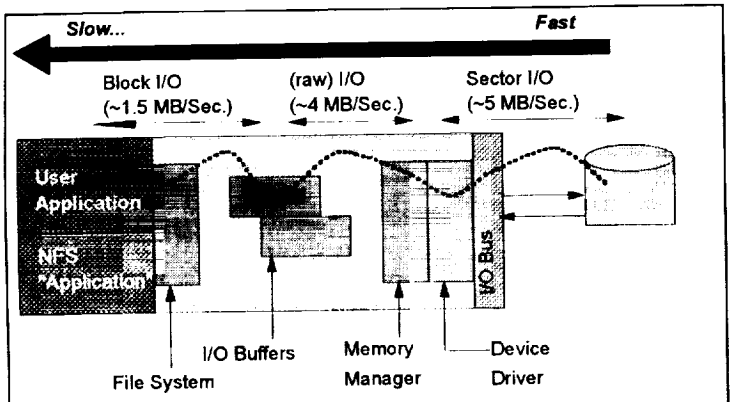


Figure 3: Effective data rate through a UNIX platform

Ethernet, at 10 Mbit/second focused everybody on item three because of its low usable bandwidth and, after some frustration, begat FDDI (and later HIPPI, Fibre Channel and ATM) which was supposed to be 10 times faster. To everybody's amazement, they did not get 10X the data rate, they got 2-3X the data on a good day and often less. Adding more FDDI rings, routers, bridges and other network paraphernalia did not seem to help. Just speeding up the fabric did not seem to be the answer. Items one and two were now the gate, but seemed to have received less attention by industry over the past several years than may have been deserved[6].

The real culprits exposed!

For an NFS server to deliver data to a client it first has to read the data off the disk subsystem in the server. In a typical UNIX environment the software protocol stack would look something like that shown in Figure two. If you measure actual disk data rates starting at the disk interface and working your way towards an application (e.g. NFS server or a user application) the effective delivered rate decreases with each step up the protocol stack. Figure three illustrates this point graphically[7]. Fast disk I/O becomes slow disk data by the time it reaches the requesting application. A typical SCSI disk of recent vintage can deliver about 5 megabytes a second of user data off the media. By the time the data has traversed the protocol stack labyrinth, the sustained delivery rate has often decreased to about 1.5 megabytes a second. Most storage subsystems[1] are not capable of delivering high sustained bandwidth to the requesting application; be it the NFS server or a users application.

Incrementalism: Disk striping, data caches and other software improvements.

Various incremental methods have been proposed and/or implemented in a limited sense to attempt to address this problem. The most common of these is disk striping wherein the disk device drivers and (usually) the layer of code that performs the function of the logical volume manager are modified to break up a users data record into smaller chunks and write (stripe) these chunks onto multiple disks simultaneously. Various RAID types may

[1] A "storage subsystem", as used here, represents the complete collection of components necessary to deliver data to the requesting application: Disk drives, I/O cards, software layers, file systems, etc.

also be imbedded in the software to increase data availability. Some high data rates have been achieved under laboratory conditions by using this method but they typically required extremely large data request sizes on the order of multiple 100's of megabytes or more that are primarily sequential single stream/single user in nature.[8] By definition, a shared network server must deliver multiple streams of data to multiple users. Each network request, when using the NFS protocol, is limited to 8 kilobyte datagrams under NFS-2 and 60 kilobyte datagrams under NFS-3. Software disk striping, at least as currently implemented, does not seem to be the answer.

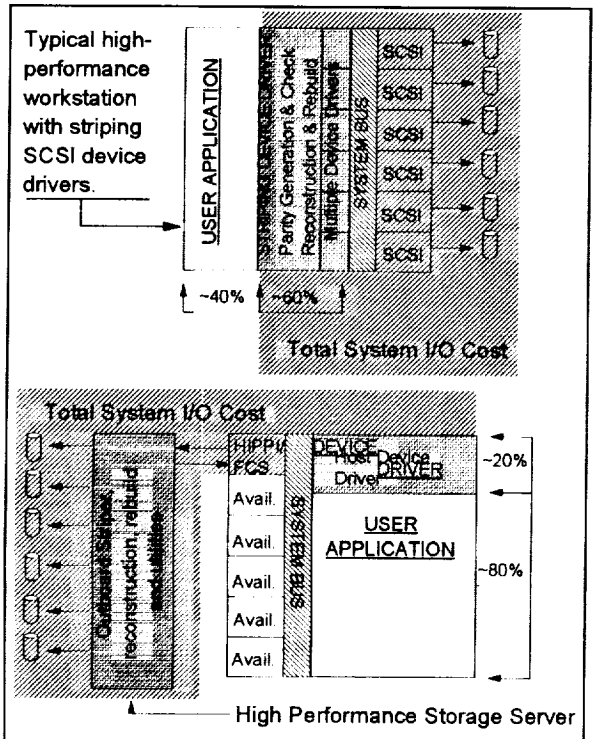


Figure 4: Software vs. Hardware striping

Massive data caching can address some of these concerns by preemptively reading ahead (i.e. turning small user requests into large I/O requests) multiple megabytes of data in order to achieve high bandwidth from the disk subsystem. Depending on the locality of reference, sequential (or non sequential) nature of the clients data access patterns, caching may or may not help. In all cases, allocating large amounts of main memory for preemptive disk caching is not cheap nor always possible without additional modification to most file systems and virtual memory managers.

Striping, and optionally software RAID artifacts, tend to add significant overhead to basic I/O operations. A 4+P RAID 5 stripe implemented on a set of generic SCSI drives and adapters requires 5 invocations of the basic disk I/O protocol stack (SCSI disk driver, card driver and unique device head codes) all contending for system I/O bus bandwidth and main memory access. Data transfer is not really parallel due to the Von Nuemann nature of most machines; rather it is (hopefully) rapidly interleaved in such a way as to appear parallel in nature to the requester. Sitting on top of these multiple device drivers would be a "collection manager" who's role in life is to re-assemble the users data records out of the disk chunks read by the device drivers (or on a write operation perform the "chunking" of the data records), verify correct parity and pass the re-assembled records to the user and/or to the file system buffer space.

We have noticed that it required approximately 5% of a large UNIX servers compute cycles to sustain a 4-5 megabyte per second data stream at the raw interface[9]. With the addition of "collection manager" overhead and optionally a software RAID function, the I/O overhead actually experienced by the user would be higher. To hypothetically sustain a 50 megabyte per second striped SCSI stream at the raw interface may consume up to 50-60% of a typical servers available cycles; thus not leaving much for any useful work such

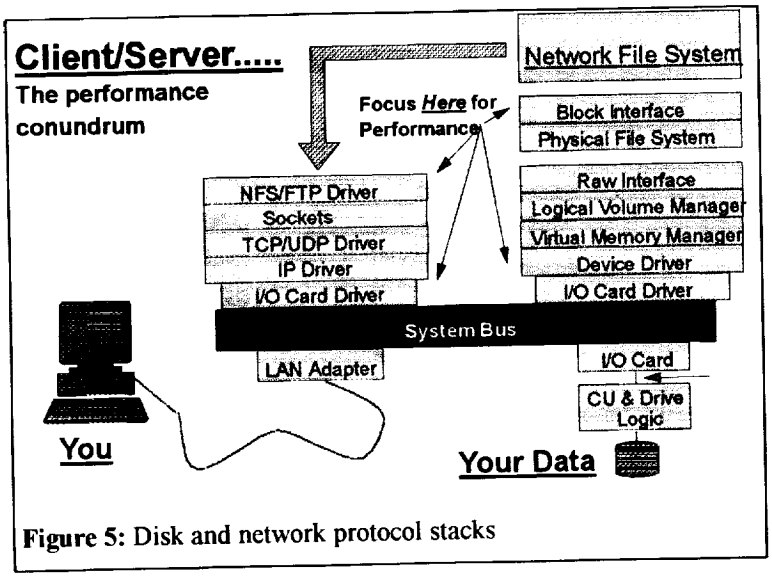


Figure 5: Disk and network protocol stacks

as actually delivering data to a client application. A better way to deliver data to the NFS server needs to be found.

The search for a better way

Based on what has been discussed previously, we recognized that using a general purpose UNIX (or other OS) compute platform as a "data bus" is inefficient, costly and may never deliver the performance required to satisfy the data absorption rate demanded by large HPC clients no matter what modifications we made to the software. Through-memory data transfer, system bus contention and general purpose I/O drivers were not designed to efficiently deliver high, sustained data rates and, when used in that capacity, deliver sub-optimal bit rates to your network clients.

Unfortunately, the worst is still to come. Assuming that the NFS server code finally gets the data it needs from the physical file system, the disk data blocks have to be sized to the users actual NFS request, packetized into datagrams and shipped over some fabric via IP protocol. Please see Figure 5 for a diagram of this protocol stack. As should be no surprise to the reader, another complete software protocol stack comes into play here further impacting the ability of a server to deliver meaningful data rates. Imagine all those little IP packets interrupting the I/O bus all the time, the OS frantically moving bits of data here and there through memory and the IP, UDP and LAN drivers all contending for precious CPU cycles. Performance problems are inevitable.

To eliminate these data bottlenecks you have to re-architect and completely re-define what a "server" is from the ground up. From our prior discussion, I think we can safely agree that it is not a workstation with lots of disk and some LAN cards. What it must be is a machine designed to manage and move large amounts of data efficiently and rapidly from disk storage to a fabric. Essentially, it should connect the disk subsystem directly to the network. It should scale (i.e. grow in usable bandwidth) as the clients and, as a second order, the request rate grows with no loss in performance. Since we are suggesting that many users entrust their crown jewels (i.e. data) to this machine, it should offer complete redundancy, virtually 7X24 access to the "jewels" and a bullet proof file system and backup scheme. A failure affects 10's to 100's of users, not just one or two. Since we are addressing high speed transfer of very large files on the order of multiple megabytes to gigabytes, file system corruption and/or physical disk storage failure could be catastrophic.

Client/Server network performance requirements definition

In order to solve the performance conundrum described above and design a file server capable of truly utilizing high bandwidth fabrics (e.g. ATM, FCS) you have to start with a

set of design points far above what has, to date, been deemed as acceptable. Some of these points that we picked for our initial design were as follows:

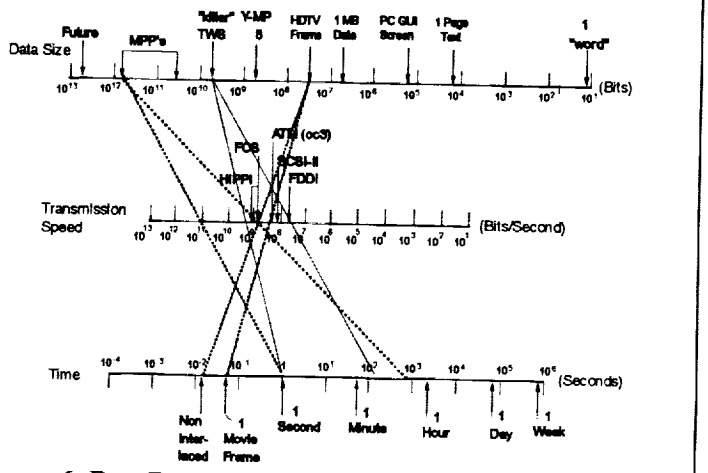


Figure 6: Data Demand by HPC Platforms and Applications

1. Sustained data delivery rates in excess of 50 Megabytes a second in order to utilize the bandwidth offered by FCS and/or multiple OC3 ATM links. As faster fabrics become available the server must be designed to accommodate them without extensive redesign of the architecture.
2. A scaleable design where the servers network bandwidth grows with the addition of more network ports vs. most current architectures where additional network ports deliver additional connectivity and fault tolerance but not necessarily more bandwidth[10].
3. Sufficient storage capacity to address the large data objects that graphical and "grand challenge" type applications tend to generate coupled to enough internal bandwidth to allow the server to access its storage subsystem fast enough to serve, for instance, multiple 155 megabit ATM OC3 links at rated speed.
4. 100% fault tolerance and 7x24 data availability for the reasons previously described.

Does anybody really need this?

Firstly, as sort of a reality check on the above specifications, we decided to more accurately understand if there is really strong market demand for extremely fast NFS/FTP servers. Earlier in this article, we discussed what was plaguing current server designs, but we did not discuss whether the user demand was 2X, 5X, 10X or whatever. Mark Seagert and Dale Nielsen at the Lawrence Livermore Computing Lab developed a model to illustrate the "data demand" of various compute platforms and/or applications. A version of that model, expressed as a Nomograph is shown in Figure 6[11].

The upper line, representing some common high performance computers aggregate data demand is compared to the lower line representing time (e.g., how long will a user wait for the data). Transmission speed (the middle line) can then be extrapolated from the size of the data object and the users "patience". We quickly realized that today's HPC demand is in the 100's of megabytes a second and growing fast.

We also interviewed most of our major customers and were able to identify four basic client/server application sets that had broad applicability in both the HPC community and the general commercial marketplace and required high sustained data delivery rates to perform well.

1. Animation: The studio standard for uncompressed, high resolution video is the D1 bit stream at 270 Megabits a second. Failure to deliver and sustain this isochronous bit stream will not allow for full motion playback of the digitized clip. All major studios, post-production and special effect houses are investing heavily in animation studios.
2. Simulation codes ability to accurately predict behavior improve as the number of points measured and the depth and width of the data stack associated with each point increases. From data preparation on workstations through large scale computing and eventual output display on frame buffers, massive amounts of data must flow through the network quickly and efficiently.
3. Data Mining: The "killer app" of the Ninety's says it all: Sifting through vast quantities of data to extract information useful to the client. Speed of data access (e.g. time to market) is everything.
4. Non-coded data storage and delivery: the collection of applications that concern themselves with the processing of non-coded (e.g. Video on demand, multi media, raw seismic data, high speed telemetry, image and pattern recognition etc.) data either deal with extremely large objects or deal with demanding isochronous data flow or, in many cases, both.

Based on the above and other market research we conclude that a large (and growing) segment of the client/server market could use a very high performance data server.

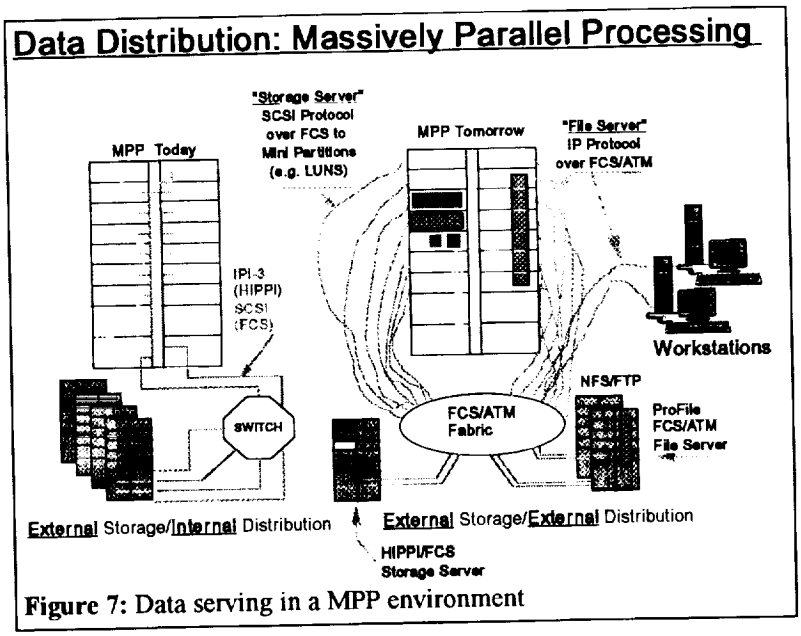


Figure 7: Data serving in a MPP environment

The MPP issue

The movement of massively parallel MIMD machines into the commercial sector coupled with the expected growth of full motion video data representations virtually guarantees that today's generation of servers will not be able to satisfy the data demand that these new applications and parallel processors will require to operate efficiently. Commercial applications tend to exacerbate the classic problems of delivering a large MIMD machine enough data to the correct node on a timely basis so as to actually utilize the massive compute power that it can bring to bear. Figure seven illustrates a conceptual idea of how very high bandwidth data serving might address this well known problem.

We are currently working with certain MPP vendors to further refine and validate the concept of massively parallel external data distribution.

Rising to the "grand" challenge

We realized that several challenges would have to be overcome to realize true high performance NFS and FTP bandwidth. Starting at the network access side, we recognized that we would have to provide multiple independent network ports which could be mixed or matched in any combination due to the heterogeneous nature of most users hardware install base. (Please see Figure 8.) HIPPI-IP, ATM-IP and Fibre Channel-IP were chosen to be the first three network interfaces supported for the following reasons:

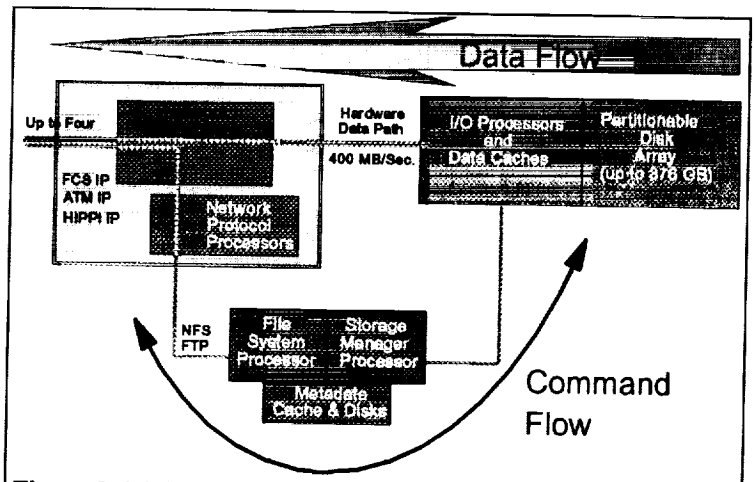


Figure 8: High Speed Server data flow ...

1. HIPPI, while somewhat costly and not as flexible as ATM and Fibre Channel, is here today, supports 800 megabit transfer speeds and is supported on most all high performance workstations, MPP's and traditional super computers.
2. ATM is rapidly developing into the high speed LAN/WAN of choice and, again enjoys near universal acceptance. While OC3 speeds of 155 megabits/second are lower that the full rated speed of our design, most clients cannot currently absorb IP data rates even that high. We recognized that as OC12 capable clients emerge, we would be well positioned to support that data rate.
3. The Fibre Channel Standard (FCS) supports gigabit transfer rates, is mature in its specifications and has been adopted publicly by IBM, HP and SUN. Other workstation vendors have told us that they plan to support this standard, at both quarter and full speed implementations, during 1995.

We considered FDDI and 10 megabit Ethernet but decided not to directly support these interfaces primarily because they lacked the bandwidth to support the marketplace we were interested in addressing and interconnection to these legacy networks could be handled by numerous vendors of routers and switches[12].

Outboard protocol processing

In order to achieve the scaleability criterion described earlier, we equipped each network port with its own integrated protocol engine to handle the IP, TCP or UDP protocol stacks completely within the attachment port. In addition, and modeled after some of the seminal work performed by the National Storage Lab (NSL) and others[13], we recognized the need to separate the command and control paths from the actual user data transfer paths so as to maximize the speed and efficiency of our internal data bus while providing for completely asynchronous and concurrent command flows.

Specifically, in the design we implemented, the outboard protocol stack engines strip the NFS and/or FTP payloads (RPCs) out of the network IP packets and route them off for processing by an independent dedicated filesystem processor. It is at least metaphorically

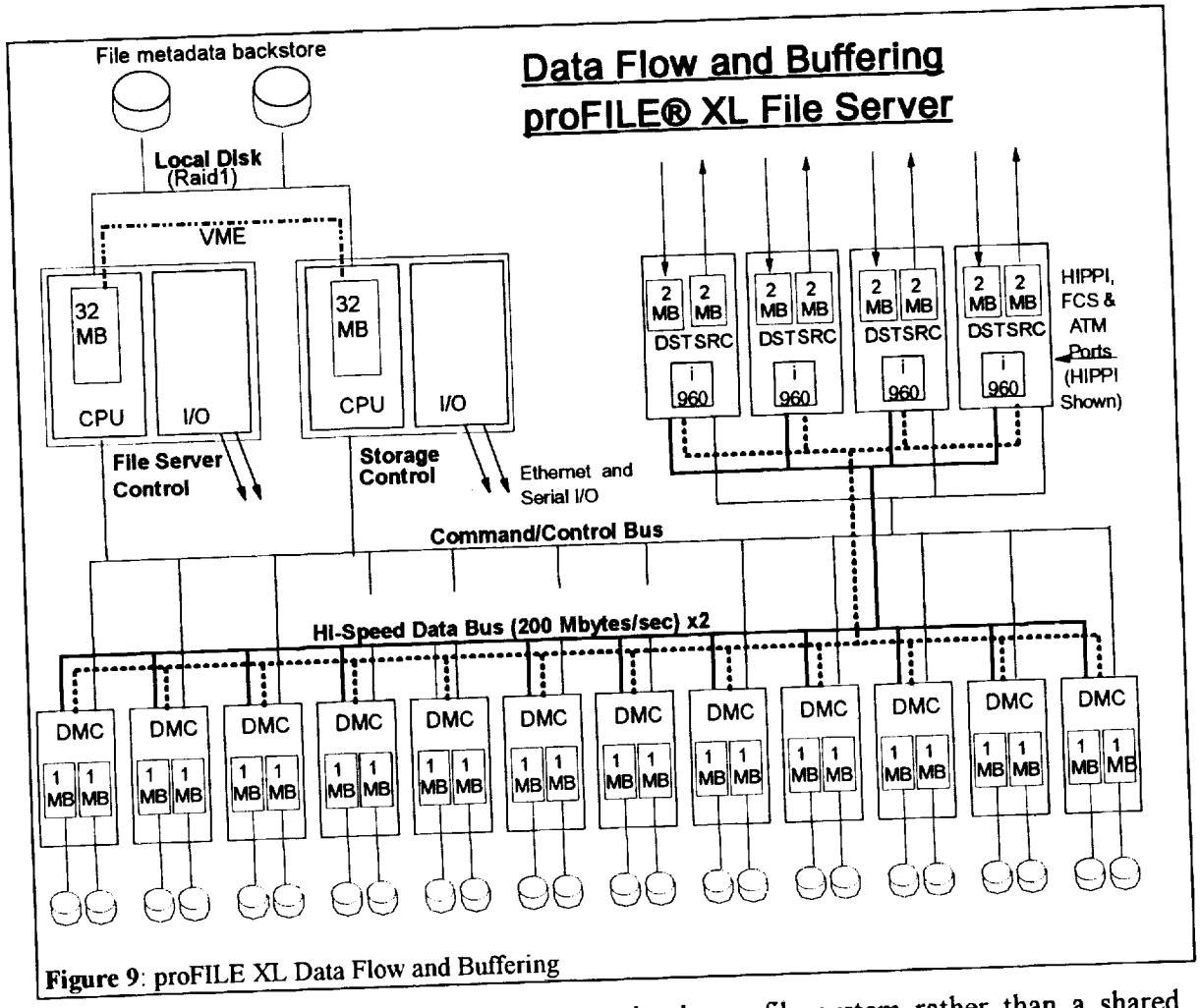


Figure 9: proFILE XL Data Flow and Buffering

correct to view the server's filesystem as a hardware file system rather than a shared software construct. A design of this nature allows the protocol engines to scale up as network connections are added to keep protocol handling from bogging down the server as users add connectivity and/or clients. This implementation addresses one of problems with today's fully software based servers; they do not scale well with connectivity and/or client load.

Filesystem processing and storage management

The file system processor, currently a Motorola 68060, accesses metadata (file identifier and i-node data) from a internal 32 MB local cache backed up by dedicated mirrored (RAID 1) metadata disks attached to both the file system processor and the storage manager processor on a local SCSI bus that is independent of the SCSI busses used to transfer user data. The local metadata cache is large enough to hold the metadata required to open and access approximately 50,000 user files under normal circumstances. Internal caching of filesystem metadata drastically reduces the time required to locate and access the appropriate user data disk in order to fulfill a NFS transfer request.

Connected to the File System Processor over a short inter-processor VME bus is a second, identical processor, the Storage Manager Processor, dedicated to managing the

physical organization of the user data and in setting up the transfers of the requested user data from the network ports either to disk or to the small write behind (fast write) caches located in the Device Module Controllers (DMC's). The DMC's are responsible for the attachment of the physical disk subsystem and can be considered to be "hardware" device drivers - please see Figure 9 This second processor, operating concurrently with the File System Processor manages the internal RAID 5 organization of the disk backstore, is responsible for management of the DMC write behind caches and controls any required recovery/rebuild processes should there be a failure of one of the DMC's and/or it's attached disks[14]. The Storage Manager Processor sets up, but does not manage, all data transfers from the DMC caches or disks to the appropriate network ports and vice versa. Over the command bus, the Storage Manager Processor instructs the DMC(s) to read or write the required number of blocks of data on/off each DMC's directly attached disk drives and transfer those disk blocks directly up to the network ports over the servers internal high speed data busses.

This function segregation between the File System Processor, The Storage Manager Processor and the multiple Protocol Processors has allowed us to not only scale the server as client connectivity and data demand grows but to tune each hardware process to efficiently implement just the functions that it was designed for and no others[15]. We refer to this design methodology as "MacroRISC(tm)" design: "Only those functions most needed shall be implemented on a processor and that processor shall be a RISC processor that efficiently implements those functions."

Internal data transfer bandwidth

The current design implements two (2) 200 megabyte/second redundant data transfer busses attached to the network ports and the DMC's via custom designed low latency chip sets. The replacement of through memory data transfer by internal "third party" transfers over 400 megabyte/second worth of hardware bandwidth eliminates another of the performance bottlenecks experienced by software only server architectures.

Each DMC is equipped with a Motorola 68020 processor that allows it to maintain its own queue of work and operate asychronously and concurrently with all other processes within the server. Under ideal conditions up to 24 simultaneous SCSI lower interface data transfer operations can be going on delivering an aggregate internal data transfer bandwidth of over 160 megabytes a second[14]. This 24-wide striped I/O subsystem allows for 100's of gigabytes of storage to be attached efficiently (e.g. no more than four LUNS on a SCSI bus) and could allow for the attachment of integrated tape backup systems, if desired, sometime in the future. Conceptually, what this distributed design does is to connect one or more disk drives directly to the network with no intervening software protocol stacks or memory bandwidth limitations.

Data availability

The File System Processor and the Storage Manager Processor are identical in hardware design and are designed to back each other up. They constantly monitor each others health and maintain mirrored metadata caches and system status latches. Should one of the two

processors fail, the other is capable of performing both the filesystem function and the storage manager function albeit at a significantly reduced level of performance. This takeover capability leads to increased levels of data availability as seen by the using clients.

A full UNIX-like system administration shell, implemented on its own administration processor, is provided for operational consistency with existing UNIX servers. A GUI interface for this shell is planned for mid '95 availability. Hot pluggable disks, imbedded RAID 0,1,3,5 hardware and N+1 power with an integrated uninterruptible power supply (UPS) complete the data availability aspects of the package.

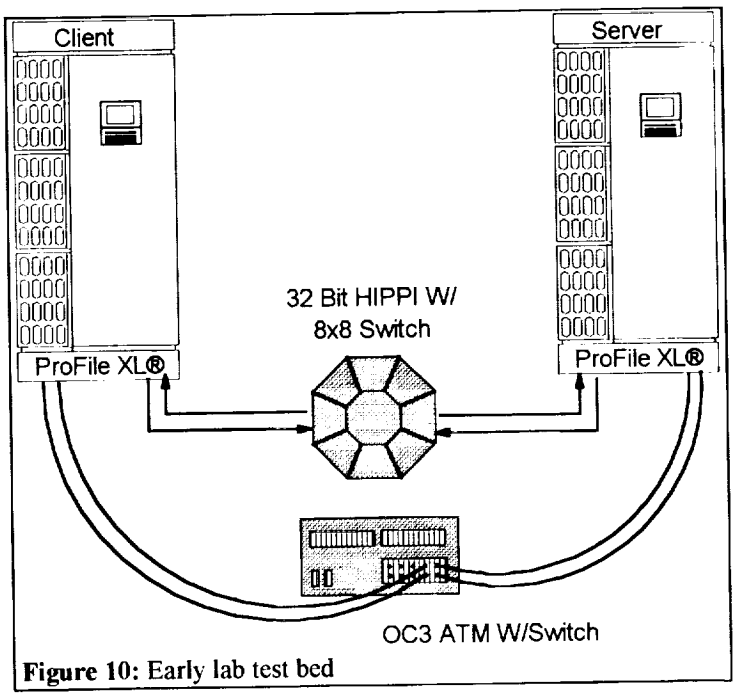


Figure 10: Early lab test bed

Measured performance and early customer experience:

When we set out to initially test the performance of beta level machines in our lab we rapidly realized that the existing "industry standard" NFS test suites based on LADDIS type workloads or the older NFS "stones" type of tests were not appropriate for this type of server. LADDIS and "stones" type tests are oriented towards measuring short, fast OLTP type workloads and not towards measuring sustained throughput of large files. Additionally, since we have optimized the server towards NFS-3 large datagram performance (although it also fully supports NFS-2 workloads) measuring short (e.g. 8K or less) requests would not allow us to test the sustained large datagram transfer rate. FTP performance was easier since measuring "throughput" is a simple matter of measuring how fast files of various sizes actually are transferred to various clients.

What to measure?

What we decided to use as a metric to represent data throughput was to measure the servers ability to deliver "N" Datagrams per Second, wherein datagrams can range in size from 8K (NFS-2 limit) to 60K (NFS-3). Sustained Throughput is the product of N and the datagram size.

During November and December of 1994 we were able to begin testing with beta level hardware and code. Recognizing that we did not have any client machines fast enough to drive the server to its limits we slightly modified one of our early servers to enable it to act as a fast client machine (See Figure 10). All initial measurements were taken utilizing 32

bit HIPPI channels. A later set was taken using OC3 ATM.[2] The graph (Figure 11) shows the relationship between throughput, datagram size and datagram delivery rate over a single HIPPI-IP port configured per the test bed shown in Figure 10 above. Several interesting items immediately come to light:

1. The server essentially has the capability to sustain a constant datagram delivery rate regardless of the size of the datagram packet. Values ranging between 800-1000 data delivery datagrams/second have been observed across all datagram sizes tested.
2. Because of observation one, delivered throughput is primarily a function of datagram size.

It should be noted that these tests were performed using a modified server as a "client" so as to remove, as much as possible, the "clients" effects on data rate. The strong relationship between datagram size and throughput may not be as linear with more traditional clients due to their potential inability to absorb high delivery rates of large NFS-3 style datagrams.

The specific initialization and opening state parameters for this test were as follows:

1. The file to be accessed had been opened and at least one request had been made so as to prime the read-ahead data caches and force the caching of necessary file and filesystem metadata.
2. Subsequent accesses were of a sequential nature.
3. The files accessed were 10's of megabytes in size or larger. Most of the small variances noticed are probably explained by file (request) size.

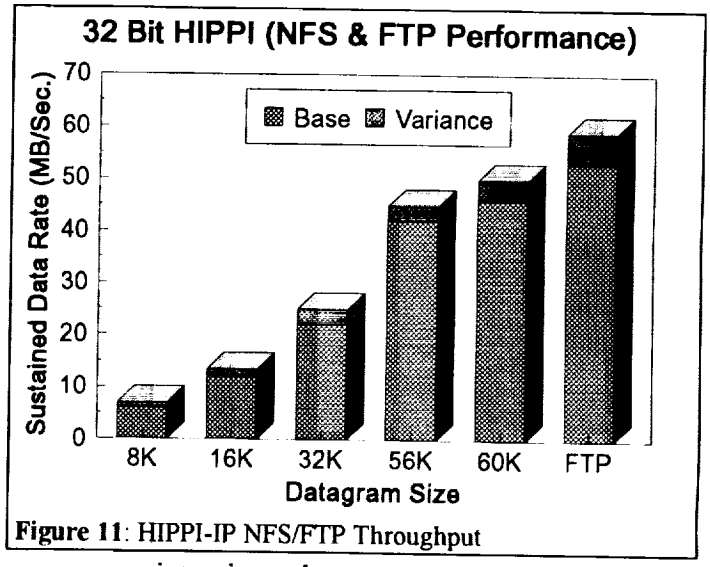


Figure 11: HIPPI-IP NFS/FTP Throughput

We felt that these initial state parameters were appropriate since the target use of this server is for applications where large files are to be accessed and the amount of data that the client requests is substantial. This test was designed to measure sustained data throughput to a client both requiring and capable of absorbing high speed IP traffic. Priming the data and metadata caches eliminates the initial latency of the "get attributes" sequences and most mechanical disk effects. Where file and request sizes are large, start-up latencies are essentially amortized over many megabytes of data transfer and become trivial. For small requests this is not the case and different start up states should be assumed for any kind of performance testing. We plan to perform more extensive performance testing over a wider range of workloads during the first half of 1995.

[2] **All results shown here should be considered preliminary due to the early levels of hardware and code used to perform these tests. Final data will be formally published in an update to this paper in the second quarter of 1995.**

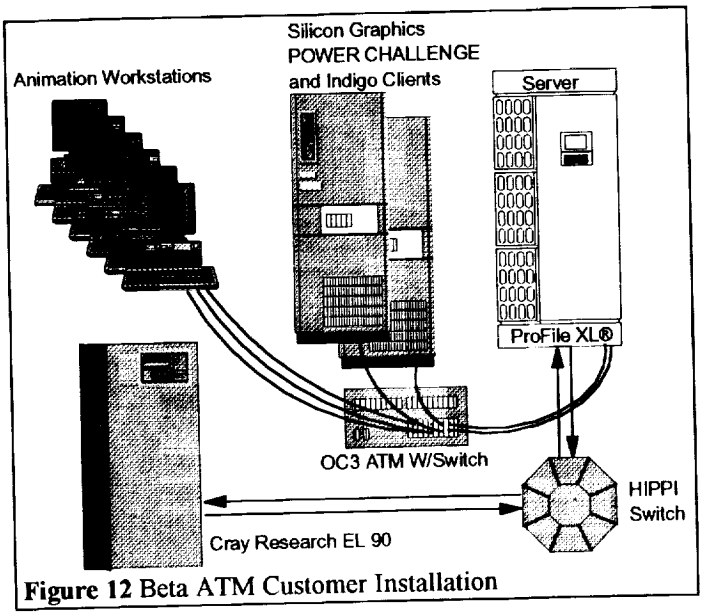


Figure 12 Beta ATM Customer Installation

ATM

ATM performance data is in the process of being developed more fully. Early lab results, obtained during December of 1994, have demonstrated the ability to sustain approximately a 12.5 megabyte/second (~100 Mbits) data stream over an OC3 (155 Mbits) ATM channel coupled to a FORE Systems *FORE-Runner*[tm] ATM switch. (See Figure 10) All switching and virtual circuit initializations were controlled by FORE Systems SPANS[16] interface code which we have implemented in the ATM versions of the proFILE server. The datagram size used to obtain these results was 56K and the initial state parameters were similar to HIPPI.

As of this date, December 1994, the maximum number of ATM channels that we have run simultaneously at this rate is two. No measurable degradation in performance was noticed. Both ATM channels sustained about 100 megabits/second of user data transfer. We expect that when all ATM performance tuning is completed sometime in the second quarter of 1995 that we will be able to saturate four OC3 ATM channels with large NFS-3 datagrams.

Early ATM Customer result:

In December of 1994 two proFILE HIPPI and ATM files servers were installed at a customer who's major application is various sorts of studio animation and video post processing[3]. The goals of this early beta site were fourfold:

1. Verify ATM-IP NFS operations when interconnected with FORE Systems products and Silicon Graphics POWER CLALLENGE XL(tm) and Indigo(tm) clients via the SPANS interface.
2. Verify HIPPI-IP (and HIPPI IPI-3) interconnect with Cray Research's EL and J-90 series of processors.
3. Measure what sustained performance could be achieved at various datagram sizes and application request patterns.
4. Insure proper operation to all clients in an NFS-2 and NFS-3 environment.

ATM & SPANS: Installing and setting up the ATM network went very smoothly. The ATM part of the network, with the exception of the ATM boards in the proFILE server which were designed by ourselves, was all supplied by FORE systems and operated well. A simple point to point star configuration was used for simplicity and guaranteed

[3] The results presented here are very preliminary and do not represent a production level environment; rather they represent interim results of an ongoing experiment.

bandwidth to each client. We and the customer specifically avoided complex mixed vendor fabrics due to the immature state of many ATM products and, more importantly, industry accepted specifications.

NFS-3: NFS-3, as implemented on pre-release versions of Silicon Graphics IRIX operating system Releases 5.3x and 6.1x[4] had some early-on stability problems and command/response state errors. This was not particularly surprising given that we were all working with non-released code and a completely new version of NFS. Fortunately, many of the NFS-3 problems were uncovered in our labs prior to install which made the installation far less painful that it might have been. Silicon Graphics was very helpful and responsive working with us to address any NFS-3 glitches in IRIX and our server code. Based on our progress to date, we expect that by the second quarter of 1995, NFS-3 will be ready for general availability and production use.

HIPPI-IP and Cray "big block" NFS: Cray Research, recognizing the performance limitations of NFS-2 years ago, implemented a proprietary Cray-to-Cray extension to NFS-2 that allowed the use of large datagrams up to 60K. This has proved to be very effective in speeding up interprocessor IP communication between Cray platforms. Peter Haas, at the University of Stuttgart, has measured sustained Cray NFS traffic up to 7.5 MB/second between a Cray Y-MP/2E server and a Cray C-94 client[17].

Datagram Size	Observed Data rate
8K	~1.8 MB/sec.
16K	~3.2 MB/sec.
32K	~5.4 MB/sec.
56K	~5.4 MB/sec.
60K	~5.4 MB/sec.

Table 1: HIPPI NFS Performance

When we initially installed the proFILE server on the Cray EL, it was configured as a storage server utilizing the IPI-3 protocol over HIPPI. Everything worked well, with data rates observed in excess of 50-60 MB/second. After determining that IPI-3 disk protocol operated correctly with the EL, we upgraded the proFILE to full file server mode and ran some initial NFS test runs at various datagram sizes.

The following table (Table 1) shows achieved data rates as a function of datagram size. We discovered that UNICOS 8.0 (Cray's operating system) would not generate a packet larger than 32K even though it was configured to do so. This problem was confined to the EL series and has been identified and corrected by Cray. Because of this, there was no throughput improvement above 32K datagram sizes. We plan to publish updated performance numbers when we retest with updated proFILE and UNICOS server code in early '95. We expect to see substantial improvements at that time.

ATM NFS Performance: As previously mentioned we achieved effective ATM saturation rates of over 100 megabits/second of user data when running two proFILE platforms as client and server respectively. (See Figure 10) When configured as per Figure 12 (proFILE to SGI Indigo) we achieved 15-18 Mbits/second sustained NFS transfer rates

[4] IRIX releases coded as "5.2x" support 32 bit hardware platforms. Versions coded 6.0x support 64 bit platforms. Release 5.2 and 6.0 are at the same basic function level. The "x" represents a non released version of the OS.

per physical ATM channel using NFS-3 8K packets. IRIX 5.3x's support of NFS-3 does not yet support any datagram sizes larger than 8K nor the ability to configure significant additional quantities of UDP datagram buffers. We expect this situation to be corrected in 1Q95.

Given that the proFILE server can hold a constant datagram delivery rate regardless of datagram size and that the client seemed to be primarily gated by the virtual memory manager, IP protocol stack and the NFS RPC interrupt handler components, we can extrapolate performance in the range of 50-80 Mbits/sec. when IRIX fully supports large (56K) datagram sizes and sufficient UDP buffering is available. (e.g. The OS client components most involved in limiting datagram absorption rate will be executed far less frequently.) Updated information will be provided in a revision to this paper later in 1995.

While both we and the customer are pleased with these early results we feel that they are not indicative of the throughput that we can achieve with production level operating system code, some application tuning and, most importantly, experience.

Summary

The distributed, parallel server design implemented in the proFILE family of network data servers has promise to revolutionize and make practical the concept of file serving to truly high performance client machines. By eliminating software protocol stacks, system and I/O busses, memory accesses and operating system overheads inherent in most "servers" today, performance levels that used to be available only on a local file system can now be delivered (and potentially bettered) on a remote, shared file system. On the client and network side, the full implementation of the NFS-3 protocol suite and the availability of fast fabrics completes the picture.

For the first time, continual data starvation will become a thing of the past and the promise of high performance Client/Server computing will become a reality.

Endnotes and References:

[1] An informal count of physical attachments in use a few years ago was on the order of 35 or more. Some examples were: IBM BMX & ESCON, DEC Q-BUS, CI & BI BUS, Univac word channel, Boroughs A-Series disk interface, NCR direct connect, SCSI: (1&2, Fast, Fast/Wide, differential and open ended), IPI: (2 & 3, Voltage or Current mode and IBM's DFCI used on the AS/400), SMD, and numerous others.

[2] Typical installations of this sort may have utilized an IBM 7094 for arithmetic operations and one or more IBM 1401 processors as I/O support machines. Early CDC 6x00 installations were similar. Cray Research users often employed large IBM and Sperry mainframes dedicated to data preparation and input support and, more importantly, as permanent data repositories. The Los Alamos Common File System (a.k.a. "Datatree" - in it's commercial incarnation) was a well known example of this scenario.

[3] Examples of some common I/O channel to I/O channel "black boxes" were the Network Systems Corporation (NSC) "DX" (Data eXchange) family of interconnect products and, at a higher function level, the Ultra-1000 network hub offered by Ultra Network Technologies.

[4] CFS was brought up in 1979 on an IBM 370/148 processor running the MVS (Multiple Virtual Storages) operating system. Datatree, released by General Atomics is functionally equivalent to CFS release 56.

[5] Most peer to peer LANS and WANS today employ one of two generic schemes to allocate bandwidth to multiple users at the physical level: 1) CSMA/CD (Carrier Sense Multiple Access/Collision Detect) which is employed by Ethernet type LANS wherein the client "listens" to the network and, if it seems to be free, transmits its data. Obviously, collisions (and retries) are common during heavily loaded periods, or 2) token controlled access, (Token Ring, FDDI, etc.) wherein a user must have access to a "token" to transmit data. Controlled access fabrics rarely have collision problems, but suffer from the higher overhead required to manage and share the token. See ANSI standard document 802.xx for further reading.

[6] There was one major exception to this statement. Under the direction of Dr. Richard Watson, the National Storage Lab (NSL) located at the Lawrence Livermore National Laboratory (LLNL) directed it's focus towards serving HPC platforms at very high speeds primarily via utilizing a construct called Third Party Transfer over HIPPI networks. (See reference 13) All data transfer was under control of a modified version of Unitree (NSL-Unitree) and is commercially available from IBM's Federal Sector Division.

[7] SCSI data rates delivered to various points in the protocol stack (driver level, raw, and block interfaces) were obtained via the use of an IBM tool "perfmon" running on an RS/6000 98B with AIX 3.2.5. There is no guarantee that any user can or will obtain these results. They are presented for illustrative purposes only. Please see IBM publications GA23-2704-00 and GA23-2708-00 for similar information concerning achieved data rates over HIPPI channels. Interestingly, while the numbers are different, the ratios hold.

[8] Ruwart, T. M. and O'Keefe, M.T. 1993. "*Performance of a 100 megabyte/second disk array*" (Preprint 93-123), University of Minnesota, Minneapolis M.N.

[9] This approximation was developed by measuring the cycle consumption required for an IBM RS/6000 980 server to sustain a 50 MB/Second data rate over a HIPPI channel utilizing the IPI-3 protocol. It required approximately half of the available processor cycles to achieve this rate thus allowing us to extrapolate that every 5 MB/sec of data rate required 5% of the processor. Striped SCSI, due to the larger number of small I/O chunk requests and the need to re-assemble such chunks would require more. For furthur reading please see:

- Arneson, D., Beth, S., Ruwart, T. and Tavakley. 1993 "*A testbed for a high performance file server*" Procedings of the 12th IEEE Symposium on Mass Storage Systems, April 26-29, Monterey C.A.
- Chen, P.M. and Paterson, D.A. 1990. "Maximizing performance in a striped disk array" Proceddings of the 1990 International Symposium on Computer Architecture, pp. 322-331.

[10] In certain extreme cases, the addition of additional I/O ports on UNIX workstations configured as servers may actually have the effect of reducing the overall throughput of the server. This counter-intuitive phenomena results from the higher multi-programming level necessary to manage the increased number of I/O ports and data movement operations that result from such additions. The increased interrupt rate across the system bus and within the OS can lead to diminished overall throughput. Data supporting this

observation, developed on an IBM RS/6000 980 server driving multiple HIPPI channels is available from the author on request.

[11] The original Nomograph upon which the representation shown in this paper is based was developed by Dr. Mark Seagert's and Dr. Dale Nielsen, both at the Lawrence Livermore Computing Lab, as a method of estimating the data demand and transfer speeds required to feed future generations of processors envisioned at LLNL. Additional data points relating to ATM and video frame transmission were added by the author.

[12] Vendors that we are aware of today who have either announced products or announced their intentions of developing products to interface HIPPI, FC and/or ATM to existing Ethernet and FDDI networks consist of Netstar Inc., Essential Communications, Bay Networks and FORE Systems. Additional vendors have announced intentions to enter this market in some form or another.

[13] Hyer, R., Ruth, R. and Watson, R. 1993. *"High performance direct data transfer at the National Storage Lab"* Procedings of the Twelth IEEE Symposium on Mass Storage Systems, Monterey, C.A. April 26-29, 1993.

[14] Wood, L. C. 1994. *Gen 5 Storage Server - General Information.* Maximum Strategy Inc., Milpitas CA.

[15] The author would like to recognize the invaluable contribution of John Lekashman, Bruce Blaylock, Bob Ciotti, and many others at NASA-Ames for assistance in the design and validation of the specific function splits described in this paper. Without their help in measuring and understanding the choke points in NFS data flow we would not have been able to accomplish this project in the time frame required.

[16] SPANS (Simple Protocol for ATM Network Signalling) is a proprietary API developed by FORE Systems as a method to set up and control FORE's family of ATM switches. The current lack of a complete standard for ATM has led to the development of several competing proprietary access and control schemes; most of them not compatible with other vendors switch and interface hardware.

[17] Haas, P. 1994. "Optimal UDP buffering for UNICOS 8.0 NFS" University of Stuttgart, Stuttgart, Germany. (an unpublished work)

A Kinetic Study of Hydrolysis of Polyester Elastomer in Magnetic Tape

K.Yamamoto, H. Watanabe
SONY Corporation Sendai Technology Center
Sakuragi 3-4-1, Tagajyo-shi, Miyagi-ken, 985, Japan
kunibo@rdds.smp.sony.co.jp
Tel:+81-367-2338
Fax:+81-367-2778

Abstract

A useful method for the kinetic study of the hydrolysis of polyester elastomer is established which uses the number-average molecular weight. The reasonableness of this method is confirmed and the effect of magnetic particles on hydrolysis is considered.

Introduction

Long archival lifetime is an essential property of magnetic recording tape for data storage. It is well-known that the archival life of tape depends on various factors, all of which may be important. This paper is a basic study on estimating the life of magnetic recording tape as affected by degradation of the binder. Polyester elastomer is used as the binder in magnetic recording tape, and one of the factors of tape degradation is hydrolysis of the binder. Hydrolyzed binder is adhesive and the tape with hydrolyzed binder may be sticky.

This paper first describes a method for the kinetic study of the binder's hydrolysis. Following that explanation, the appropriateness of this method is discussed, together with the influence of the magnetic powder.

Method and Materials

Determination of the reaction rate is necessary for estimating the tape life. Ester hydrolysis is a second order reversible reaction in which ester group and water are involved. Since there is a large quantity of water in the air, it may be assumed that the amount of water in the air is constant. Thus the rate equation of ester hydrolysis can be expressed as a function of ester concentrations only (1).

$$C_e/C_{e0}=\exp(-k't) \qquad (1)$$

C_e : Ester concentration after storage.

C_{e0} : C_e at T=0 (before storage)

k' : Rate constant.

t : Storage time

The reaction rate is calculated using the changing rate of ester concentration, but it is difficult to measure the ester concentration in polymers. The rate equation for this case can be expressed using the molecular weight of polymers.

Defining the number of molecules in unit weight as N and ester concentration in unit weight as C_e, the relationship between N and C_e may be shown as in figure 1 and represented by the following equation (2).

$$C_e=N_e-N \qquad (2)$$

C_e : Ester concentration in unit weight

N : Number of molecule in unit weight

N_e : N after storage at $C_e=0$

Combining equations (1) and (2) leads to (3).

$$N=N_e-C_{e0}\exp(-k't) \qquad (3)$$

Defining the number-average molecular weight in unit weight as M_n, the relationship between N and Mn may be represented by the following equation (4).

$$N=1/M_n \qquad (4)$$

Combination of equations (3) and (4),

$$1/M_n=N_e-C_{e0}\exp(-k't) \qquad (5\text{-}1)$$

(5-1) at t=0 gives (5-2)

$$1/M_{n0}=N_e-C_{e0} \qquad (5\text{-}2)$$

M_{n0} : M_n at T=0 (before storage)

Eliminating N_e from equation (5-1) and (5-2),

$$1/M_n-1/M_{n0}=C_{e0}\{1-\exp(-k't)\} \qquad (6)$$

Approximating exp(X) when X<<1 leads to (7).

$$1/M_n-1/M_{n0}=C_{e0}k't \qquad (7)$$

Finally, redefining $C_{e0}k'=k''$,

$$1/M_n-1/M_{n0}=k''t \qquad (8)$$

Equation (8) is the rate equation of ester hydrolysis expressed by number-average molecular weight (M_n) of the polymer and it is used in estimating reaction rate. Equation (8) coincides with the empirical equation of Huisman [1].

The sample used in this study is a normal chain polyester binder and initial molecular weight varies from 30,000 to 40,000. This simple structure is chosen for a basic study. Thin film made from this binder is stored in accelerated aging conditions, that is, high temperature and high relative humidity. After a few weeks' storage, the film is dissolved in tetrahydrofuran (THF) and the molecular weight measured by gel permeation chromatography (GPC). The conditions of GPC are as follows:

System : Waters GPC system
Columns : Waters Ultrastyragel 500 angstrom and Linear 10^6 angstrom
Effluent : Tetrahydrofuran (THF)

Detector : Differential refractometer (RI)

Results and discussion

Figure 2 shows plots of $(1/M_n-1/M_{n0})$ vs time for storage at 30 C/90% RH, 50 degrees /90% RH and 65 C/90% RH. These plots show that $(1/M_n-1/M_{n0})$ is proportional to time and confirm the rate equation (8). The reaction rate constants which are the slopes of the plots increase as the temperature increases. Table 1 shows the rate constants at 90% RH which are calculated from the plots of Figure 2.

Figure 3 shows the Arrhenius plot of the rate constants. An activation energy is calculated from Arrhenius' equation and it is about 110kJ/mol. Now we can estimate the rate constants at various temperatures when the relative humidity is constant at 90%.

Figure 4 shows Arrhenius plots in other humidities. Activation energy is not dependent on relative humidities.

Figure 5 shows the relationship between rate constants and relative humidities at 65 C. Relative humidities and rate constants are in proportion. The reason for this is that the sample films are so thin that moisture diffuses rapidly. Equation (9) derives from (8) in consideration of this effect.

$(1/M_n-1/M_{n0})/H=k*t$ (9) H : Relative humidity

Rate equation (9) exhibits the effect of relative humidity.

The effects of storage temperature and humidity on the hydrolysis of polyester are clarified. Thus we can estimate the reaction rate in every environment. Half value periods of molecular weight can be estimated.

Table 2 shows rate constant and predicted half value periods of molecular weight. The hydrolytic speed of 65 C/90% RH is about 1000 times that of 20 C/65% RH. For example, half value periods of molecular weight of this sample are estimated to be about 100 days at 65 C/60% RH and 50 days at 65 C/90% RH. The number of days was confirmed by storing until the molecular weight decreased to half value. Figure 6 and table 3 show the data.

Binder and magnetic particles are principal ingredients of the paint of magnetic tapes. We found that the presence of magnetic particles reduced activation energy. Figure 7 shows comparison of Arrhenius plots of polyester binder with metal particles, with oxide particles and without particles. Activation energy decreases when magnetic particles are mixed with polyester.Catalytic action by magnetic particles is shown by these data. This shows that ester hydrolysis is accelerated by catalytic action of magnetic particles.

Catalytic action disappeared in the case of binder with magnetic particles covered with adsorbate citric acid or the like, shown in figure 8. The decrease of activation energy is not observed for the binder with magnetic particles covered with citric acid. It is supposed that such catalytic action occurs by interaction of activation points of the magnetic particles and binder adsorbed at the points. Thus the catalytic action disappeared when the activation points were covered with adsorbate.

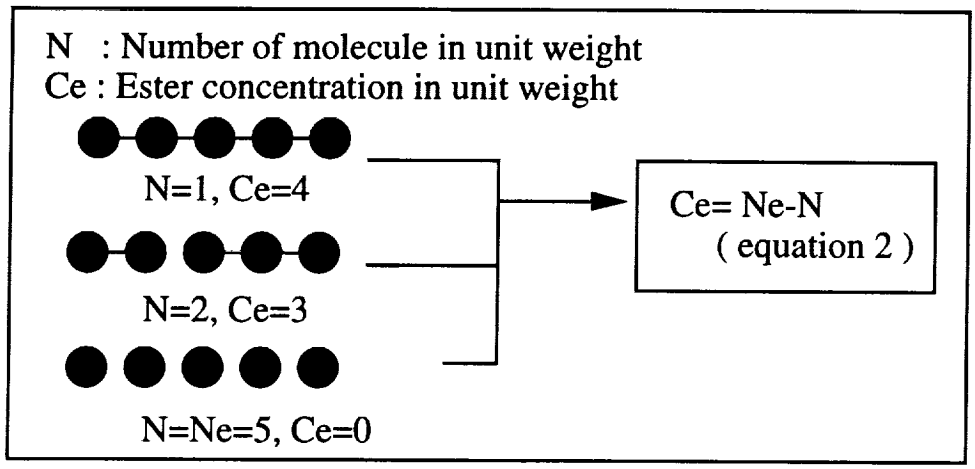


Fig.1 Relationship between N and Ce

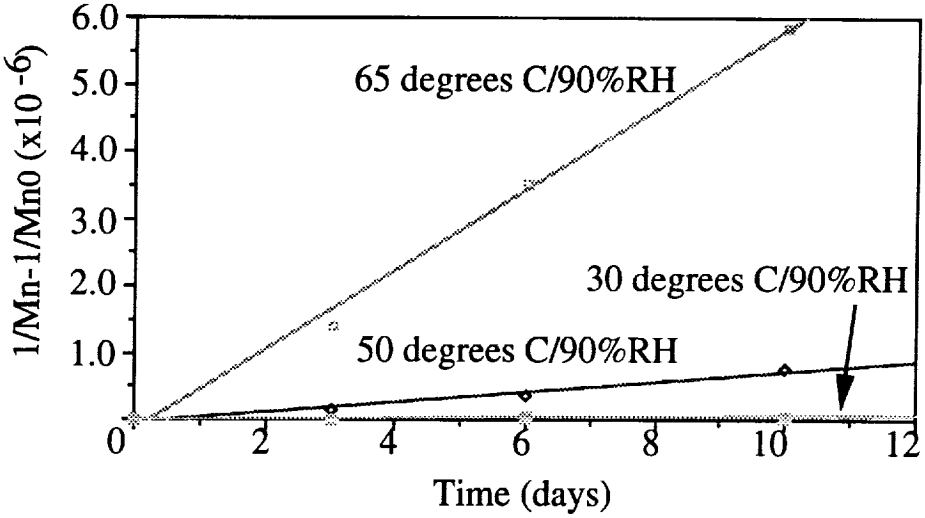


Fig.2 Variation of molecular weight in 90%RH

Table1. Rate constants at 90%RH

Temp. (degrees C)	Rate constants k" (1/days)
30	5.0×10^{-9}
50	7.6×10^{-8}
65	6.0×10^{-7}

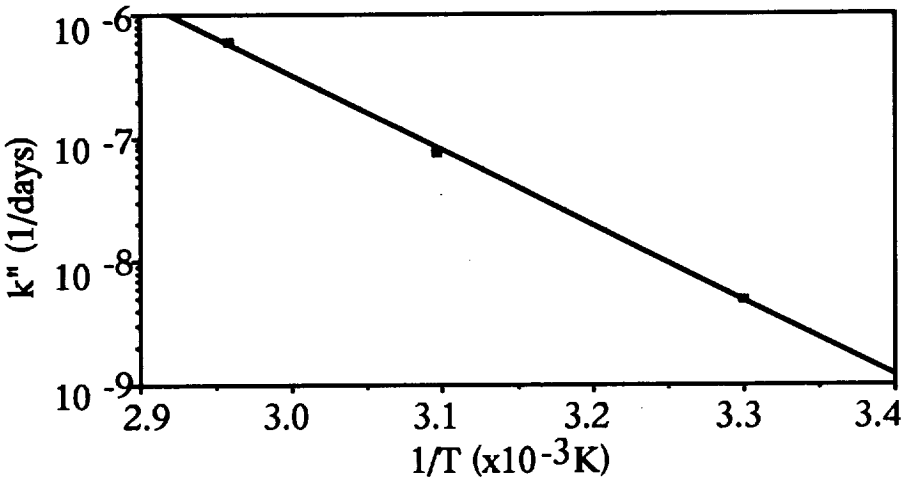


Fig.3 Arrhenius plot of polyester binder in 90%RH

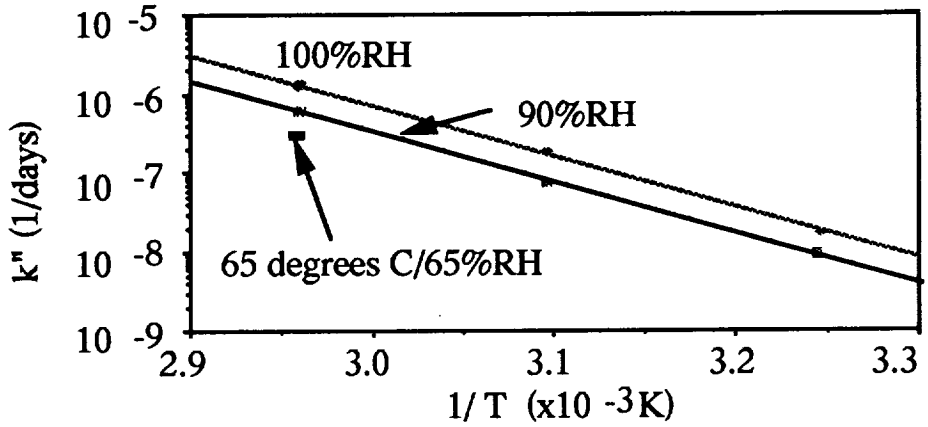


Fig.4 Arrhenius plots of polyester binder in various humidity.

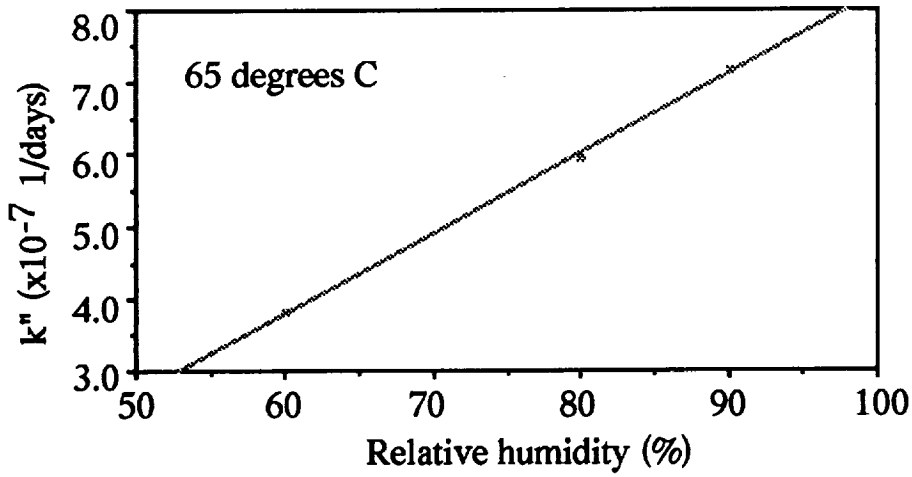


Fig.5 Effect of relative humidity on rate constants (k")

Table2. k",k' and Half value periods of molecular weight (Mn)

Temp.	%RH	k" (1/days)	k* (1/days)	Half value period (years)
20	65	8.0x10 -10	1.2x10 -11	110
30	60	4.1x10 -9	5.0-7.0 x10 -11	15.0-25.0
	90	4.9x10 -9		
40	60	1.7x10 -8	2.0-3.0 x10 -10	4.0-5.0
	90	2.0x10 -8		
50	60	6.3x10 -8	8.0-11.0 x10 -10	1.0-1.5
	90	7.6x10 -8		
65	60	3.9x10 -7	6.0-7.0 x10 -9	0.1-0.2
	90	5.9x10 -7		

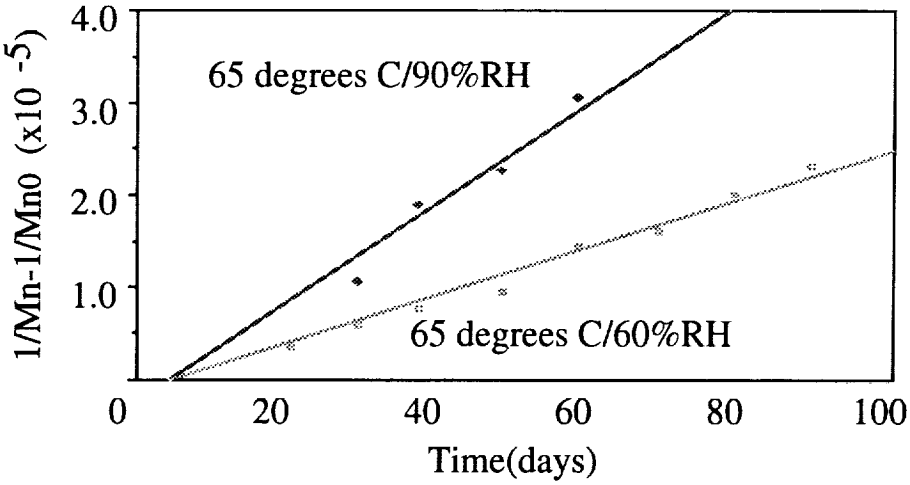


Fig.6 Confirmation of half value periods

Table3. Molecular weight of binder stored until half value period

Time (days)	Molecular weight (Mn)	
	65 degrees C/60%RH	65 degrees C/90%RH
0	39041	39041
9	35847	37367
29	31667	27620
37	29975	22419
48	28495	20730
58	24909	17781
69	24002	
79	21880	
89	20512	

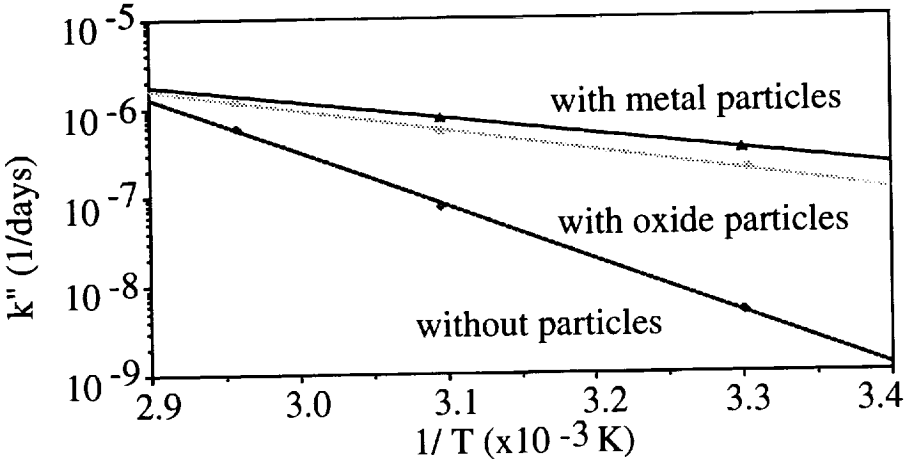


Fig.7 Comparison of Arrhenius plots of k" between with magnetic particles and without magnetic particles

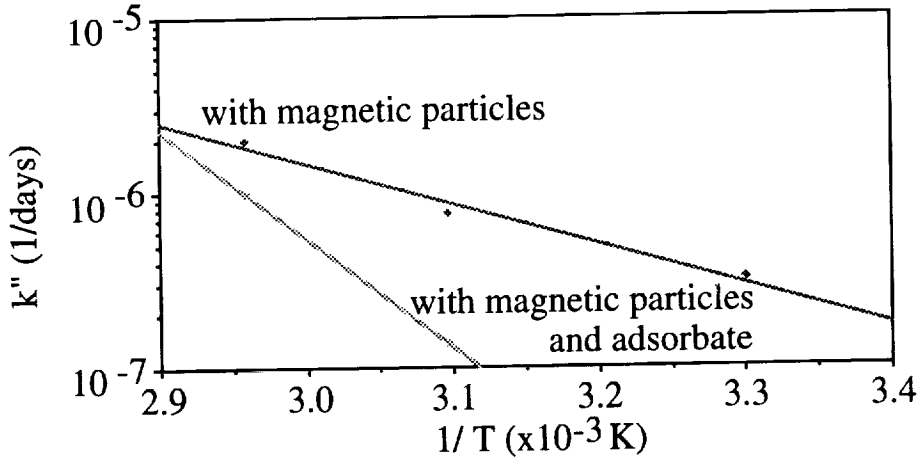


Fig.8 Comparison of Arrhenius plots of k" between with adsorbate and without adsorbate

Conclusions

1. We established a useful method for the kinetic study of the hydrolysis of polyester elastomers in magnetic tapes. Using number-average molecular weight (M_n), the rate equation of polyester hydrolysis led to the equation $(1/M_n-1/M_{n0})/H=k*t$.

2. Catalytic action by magnetic particles is demonstrated and it is supposed that such catalytic action occurs by interaction of activation points of magnetic particles and binder adsorbed at the points. The catalytic action disappeared when the activated points were covered with adsorbate.

3. We make use of this method and these results to estimate the life of magnetic tape.

References

H.F.Huisman.,et.al. "Aging of Magnetic Coatings" PD Magnetics B.V., Oesterhout, NLD, 16(2), 177-195(1988) [1].

Digital Linear Tape (DLT) Technology and Product Family Overview

Demetrios Lignos
Quantum Corporation
333 South Street
Shrewsbury, MA 01545
+1-508-770-3495
lignos@tdh.qntm.com

Introduction

The demand that began a couple of years ago for increased data storage capacity continues [1]. Peripheral Strategies (a Santa Barbara, California, Storage Market Research Firm) projects the amount of data stored on the average enterprise network will grow by 50 percent to 100 percent per year. Furthermore, Peripheral Strategies says that a typical mid-range workstation system containing 30GB to 50GB of storage today will grow at the rate of 50% per year. Dan Friedlander, a Boulder, Colorado-based consultant specializing in PC-LAN backup, says "The average NetWare LAN is about 8GB, but there are many that have 30GB to 300GB....."

The substantial growth of storage requirements has created various tape technologies that seek to satisfy the needs of today's and, especially, the next generation's systems and applications. There are five leading tape technologies in the market today: QIC (Quarter Inch Cartridge), IBM 3480/90, 8mm, DAT (Digital Audio Tape) and DLT (Digital Linear Tape). Product performance specifications and user needs have combined to classify these technologies into low-end, mid-range, and high-end systems applications. Although the manufacturers may try to position their products differently, product specifications and market requirements have determined that QIC and DAT are primarily low-end systems products while 8mm and DLT are competing for mid-range systems applications and the high-end systems space, where IBM compatibility is not required. The 3480/90 products seem to be used primarily in the IBM market, for interchangeability purposes.

There are advantages and disadvantages for each of the tape technologies in the market today. We believe that DLT technology offers a significant number of very important features and specifications that make it extremely attractive for most current as well as emerging new applications, such as Hierarchical Storage Management (HSM). This paper will demonstrate why we think that the DLT technology and family of DLT products will become the technology of choice for most new applications in the mid-range and high-end (non-IBM) markets.

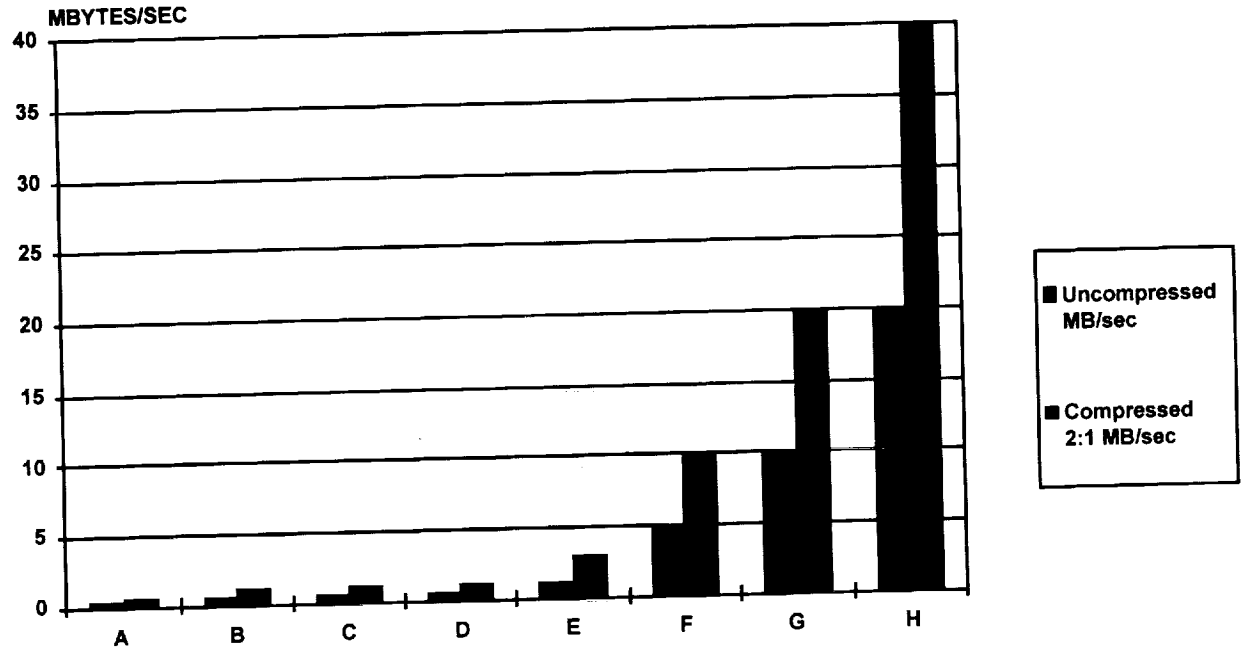
DLT Technology — Media, Mechanics, and Electronics for Performance and Reliability

The choice of using Digital Linear Technology (versus analog and/or helical scan) to develop our tape storage products was made after an in-depth analysis of the tape media and head technologies available in the late 80's. We decided on metal particle (MP) media and a tape cartridge that permits the creation of several generations of DLT products [2].

The DLT engineering development team recognized the potential of MP media early on. Products using MP technology were already using MP tape when the first DLT product was introduced into the OEM market in December, 1991, but the origin of 8mm technology was actually a consumer product that was already designed to use a consumer grade version of 8mm tape. We chose MP after an exhaustive set of tests with all of the then-available types of media, including SVHS, Barium Ferrite, Chromium Dioxide, and MP, because our testing proved to us that MP was to become technology's media of choice.

Initial reaction from a number of industry experts was that we had made the wrong decision. The pending announcement of IBM's NTP (New Tape Product) and the recent announcement of STK's REDWOOD product (both designed for high capacity and performance) are solid proof that our choice of MP media for our DLT products was correct. In addition, both the 8mm and DAT media products already depend on MP media for their newest and future generation products.

We chose linear recording technology (vs. helical scan), because, with the help of Digital Equipment Corporation system architects, we were able to foresee that transfer rate, which was not important until the early 1992 time frame, was going to be increasingly important in the future. We began with a 2-channel head design (using ferrite head technology) for the first four DLT family members. It has been established that linear recording technology allows for the increase of read/write channels with their corresponding increase in the transfer rates. Figure 1 illustrates the transfer rate potentials for the leading 4mm/8mm technologies versus linear recording technologies such as QIC and DLT. The graph illustrates ability of DLT technology to continue increasing the transfer rate of subsequent generation products by adding more parallel channels (4 channels, 8 channels, etc.).



A = 4mm, standard speed
B = 4mm, high speed
C = 8mm, 1 channel
D = 8mm, 2 channel
E = 2 channel DLT
F = 4 channel DLT
G = 8 channel DLT
H = 18 channel DLT

Figure 1: Data Transfer Rates of Competing Tape Technologies
(Based on First Generation MP Media Products)

We chose a 4" x 4" x 1" single-reel cartridge that could handle as much media as possible in a tape drive that could fit in the 5.25 inch form-factor envelope. We have already demonstrated on an earlier generation DLT product (the TZ30), that even a half-height, 5.25 inch form-factor product is possible using the DLT cartridge. The cartridge size and the half inch tape (versus quarter inch or other sizes) ensures that whatever capacities other technologies accomplish with new media (MP1, MP2, BaFe, ME etc.) the DLT products can surpass from 8 to 16 times, because of the amount of physical media area available inside the cartridge.

Figure 2 illustrates the capacity potential of various technologies. The bars indicate the physical area that the media from each of the cartridge technologies would occupy, if it was just laid out on a flat surface.

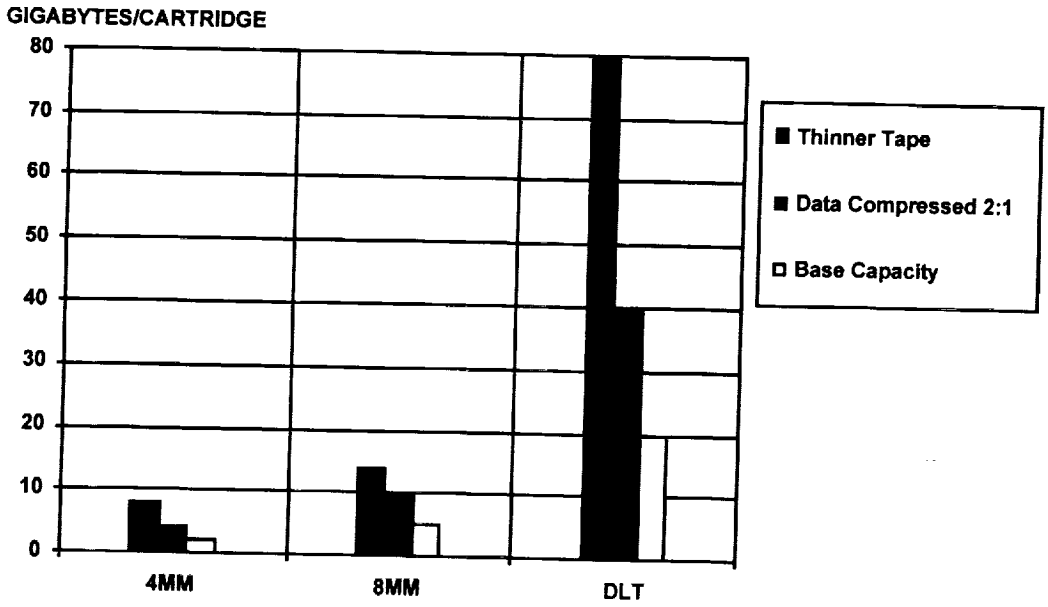


Figure 2: Capacity Potential of Various Technologies

The combination of capacity per cartridge and transfer rate, coupled with industryleading reliability and data integrity make DLT a technology ideally suited for meeting the rising demands for data storage and the clear choice of products for the balance of this decade and, possibly, well into the next.

The DLT's design features illustrate its robust nature. Mechanics, electronics, and interface have been developed to provide a platform for performance and growth.

The heart of the DLT mechanical design is the Head-Guide Assembly (HGA). The HGA is basically the tape path, with the head mounted on a head bracket in an integrated sub-assembly. The tape path is comprised of six rollers, three on each side of the head. The head bracket sits on a stepper motor lead screw that positions the head in a horizontal/vertical motion only, allowing for random access operation. The DLT uses 128 tracks, addressed in pairs by the 2-channel ferrite head in the DLT2000 product.

The primary strategy for the DLT mechanical design was to create a platform capable of multi-generation products. The original HGA design resulted in a number of patents for the basic mechanism. To achieve the tracking margin requirements [3], the off-track error budget elements are monitored and controlled continuously throughout the manufacturing process and via strict parts specifications. Furthermore, a stringent off-track test is performed on every DLT drive, prior to the"Confidence" and "Data Interchange" testing performed in manufacturing prior to shipping the product.

Because of the superior tape tracking and positioning accuracy of the HGA (Figure 3), so far there has been no need to introduce a closed loop servo control on any DLT product. Instead, the positioning accuracy throughout the entire length of tape is achieved by a combination of a pair of calibration tracks located ahead of the BOT coupled with an extensive adaptive calibration process and a series of adaptive positioning algorithms [4]. These calibration tracks, which also serve to detect the recording density of the drive, and, therefore, the specific DLT family member, are not pre-recorded. When a drive sees a blank tape cartridge, it automatically lays down the calibration tracks before any other operation takes place. From this point on, the cartridge will always indicate its recording density, thereby identifying how it should be read or written by any other DLT family member into which it is loaded. It is not possible to record multiple densities on the same cartridge.

The "buckling" mechanism is a self-threading mechanism whose reliability has been demonstrated in the over 600,000 DLT products that have shipped since 1985, spanning two generations of DLT drives (seven DLT family members). Each generation benefits from the experience of the previous generation, leading to perfection in this very critical part of the design. The need for robustness in the Load/Unload Mechanism is essential to withstand continuing punishment of the hardware in tape library environments. The DLT Load/Unload operation is heavily assisted by a number of firmware algorithms that guarantee the reliability of the DLT's mechanical operations.

There are no capstans involved in the design. The tape moves in and out of the cartridge via a precise servo control of the two reel motors. The servo control is designed to guarantee a constant 4.5 oz. tension in front of the head. The adaptive techniques mentioned earlier, however, can introduce automatic tension adjustments if the drive detects a soft area on the media or other reasons that may weaken the signal amplitude and/or resolution.

The DLT family's electronics support the design's requirement for performance and expansion. The read/write channel for most recent DLT family members (DLT2000 and DLT4000) is designed using RLL (2, 7) recording technique. The bit density of the DLT4000 is 82,500 bits per inch (bpi). The tape speed is constant at 110 Inches Per Second (IPS) during read and write operations. The best way to describe the sophistication of the electronics, is to discuss some of the areas of adaptive techniques in the design: servo and tape thickness adaptation, track positioning, head media mechanics and electronics, and data position "learning".

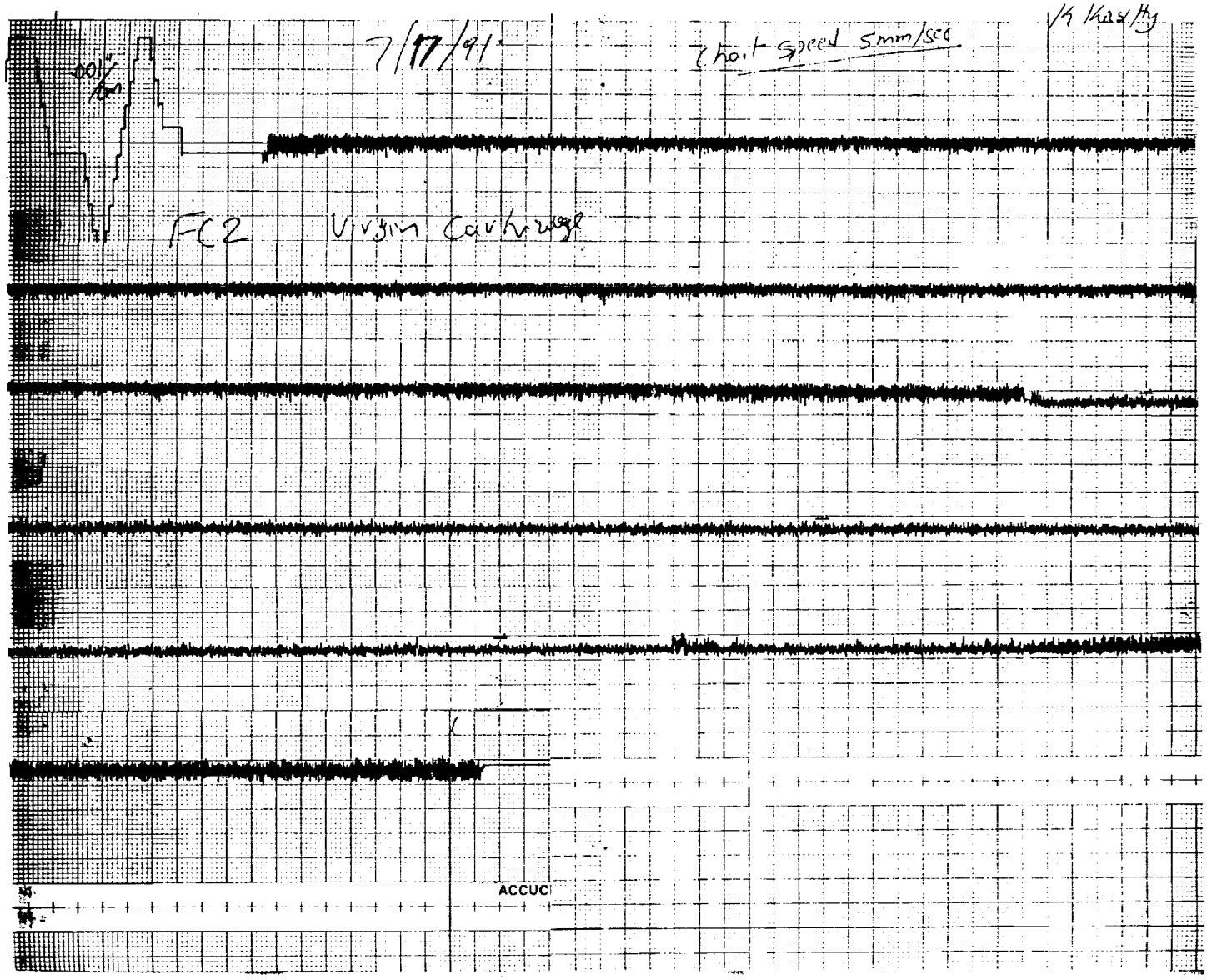


Figure 3. Signal Trace Showing Lateral Motion of Tape (HGA Tracking)

Mechanical variation of the media (which may result from manufacturing process tolerances, for example), if not properly compensated for, can result in operation failures or a control system that runs at a sub-optimal performance level. If this occurs, the tape thickness and other dimensions are actually measured by the drive. The results of these measurements are used for the various servo optimizations.

As indicated earlier, newly-purchased tapes are completely empty of data. The drive will detect a blank tape and write on it a pair of calibration tracks. These calibration tracks are written only once. The drive uses these tracks (located ahead of the BOT physical hole), much the same way as a disk drive uses its servo tracks (a detailed description of the calibration tracks' location and handling is provided in the DLT ANSI and ECMA standards.) Accuracy in tape path design and manufacturing assembly is essential to guarantee interchangeability without the need for additional servo information recorded anywhere on the 1,100+ feet (DLT2000 cartridge) or 1,700 feet (DLT4000 cartridge) of tape.

The DLT position algorithms are extremely accurate. They are able to determine the track centerline to within 100 micro inches accuracy using extensive filtering and various interpolation techniques.

At current DLT product densities (82.5 KBPI), any manufacturing process variation can result in loss of signal. To minimize tolerance sensitivities and improve manufacturing yields, our DLT design utilizes a completely adaptive channel. Every time a previously written cartridge is loaded, the drive adjusts its read/write electronics to ensure optimum operation. These adjustments are not a simple correction but a very complex estimation theory based on past experience with adaptive techniques.

The following parameters are automatically adjusted: write current, operating controls (tension, position, etc.), mechanical offsets (head adjustments, etc.), and automatic gain controls/channel control/various responses, etc.

The DLT drive's intelligence actually extends into "learning" the data on the tape . Information such as "end of data" (EOD) location and Tape Mark (TM) counts allow the DLT to find data boundaries at very fast access times by using diagonal searches. The DLT2000 and DLT4000 drives can search for Tape Marks at 150 IPS (a 300+ Mbyte/sec. equivalent search speed.) The Tape Mark directory is totally transparent to the user and is maintained and updated automatically following the completion of a write operation.

The DLT calibration process completely replaces the familiar electrical and mechanical factory adjustment that most of today's tape drives require: There are no pots, capacitors, head adjustments or any other fine tuning. All adjustments are done by the two on-board microprocessors during the calibration stages.

One of the unique (and well patented) features of the DLT design is its superior head and media interface implementation. The DLT's unique head design alone deserves a separate paper. The head is ferrite with MIG. It has six elements (2 x W-R-W channels operating simultaneously). Figure 4 shows the head geometry configuration.

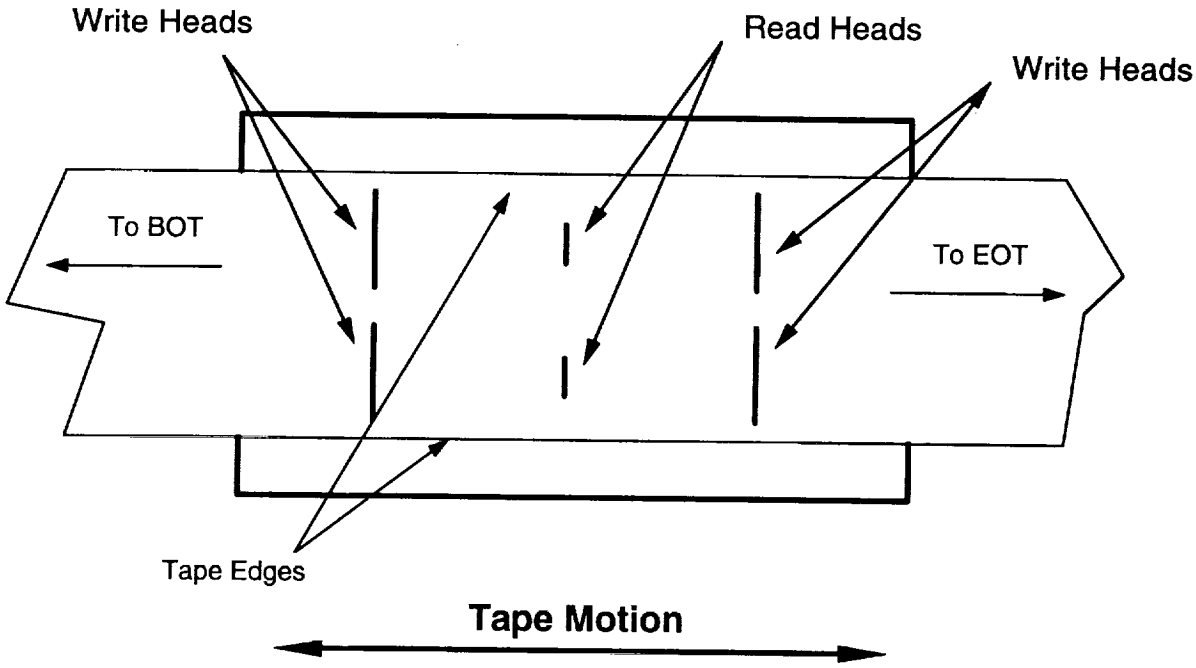


Figure 4: DLT Head Geometry Configuration

The write-read-write placement of the gaps allows for "read while write" operation in both directions. This unique design of head elements and contour combine to give the DLT products great areal density capability as well as self-cleaning behavior. The unique contour design virtually eliminates any separate head cleaning operation. A cleaning cartridge is available, but only to be used when the drive illuminates the cleaning indicator on its front panel.

The combination of the head contour design and low tape tension results in a head life that exceeds 10,000 hours (at 100% duty cycle). The length of DLT head life is a significant advantage over other technologies when robustness is necessary for high duty applications. The gentleness and accuracy of the tape path design, coupled with the head contour design, low tension and the quality of the MP media itself contribute to a tape durability that exceeds 500,000 passes (a "pass" is defined the movement of a single segment of tape under the head). Assuming a worst case scenario, one cartridge can be used 10,000 times to completely write or read all 128 tracks (64 pairs) in the serpentine mode.

Our data on life and durability of the DLT tape shows that to date we have been unable to find a measurable end of life for the tape. Our tests in environmental chambers have been designed to simulate 10 years of actual, continuous drive operation, and the tests are still running. The number of passes the still-readable tape has made over the head is now approaching 1,000,000. One of the original requirements for the DLT product family, was to implement an "Industrial Strength" class of data integrity and reliability algorithms. The results of our design approach are an unsurpassed combination of data detection and correction algorithms that produce a "Hard Error Rate" of 1×10^{-17} bits read and a combined theoretical overall undetected error rate of 1×10^{-30} bits read.

Figure 5 shows the data format for the DLT2000 and DLT4000 products. The format consists of multiple "entities." An entity is comprised of 16 x 4K data blocks and 4 x 4K ECC blocks. Within the entity, the format supports record sizes varying from 1 byte to 16 Kbytes, as Figure 5 indicates. At the end of each record (regardless of size), the drive records a 16-bit cyclic redundancy check (CRC). At the end of each physical 4K block, the drive records a 64-bit CRC that checks the entire 4K block with as many records as it contains. The entity is protected by a "Block-Level Interleaved Reed-Solomon ECC" code, that occupies the last four 4K-blocks of the entity. The ECC algorithm is capable of correcting any four 4K-blocks at any place within the 20 block entity (including the ECC field itself). In terms of physical tape space, it is possible to remove a half-inch section of the tape and the drive will be able to accurately reconstruct the missing information. There is a detailed description of the DLT format and ECC/CRC algorithms in the applicable ANSI and ECMA Standards documents.

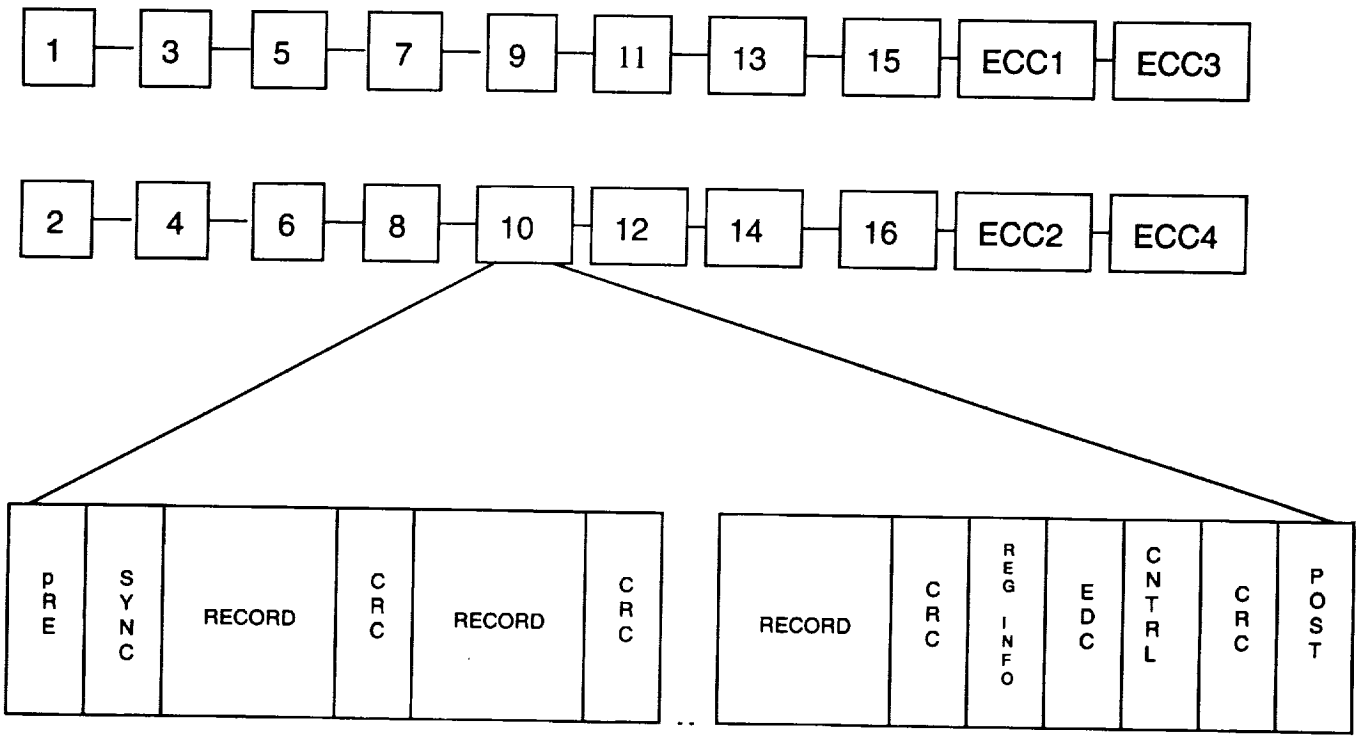


Figure 5: DLT2000/DLT4000 Media Format

In addition to the ECC and CRC error detection and correction features, the DLT drives are capable of using "track centerline offsets" (like disk drives) to attempt to recover the data as part of the automatic hard error recovery procedure. No software intervention is needed for the hard error recovery process to be invoked.

The Data Compression Algorithm chosen for the DLT2000 and succeeding products, is a variant of Lempel-Ziv (LZ1). The DLT Engineering Development Group chose LZ1 (versus IBM's IDRC), after prototyping both algorithms with identical DLT Tape Drives in the Engineering Labs [5]. It was expected that the IDRC prototype would out-perform the LZ1 prototype because it had, potentially, almost twice the data throughput. It was also expected that the two competing algorithms would have roughly the same compression ratios, with the LZ1 ratio being only slightly higher. Test results showed, however, that in the DLT environments the LZ1 consistently exceeded the IDRC performance in both metrics. The IDRC compression efficiency results were also confirmed by benchmarking against other tape products that use the IDRC algorithm.

Figure 6 shows the measurements of compression ratio on VMS and UN*X systems. The difference in compression ratio between the LZ1 and IDRC prototypes show that the LZ1 prototype had significantly higher compression ratios for all the data types that were tested.

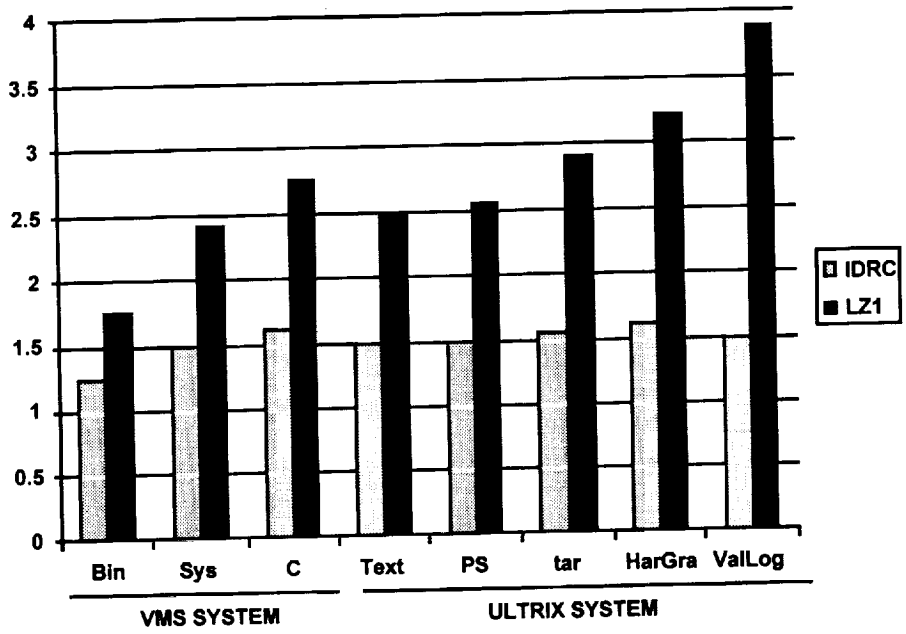


Figure 6: Operating System Compression Results

Figure 7 shows the SDS-3 based compression test results. The first four data types show the LZ1 prototype averaging around 2.4:1 and the IDRC prototype around 1.5:1. For the paintbrush bitmap file, both versions compressed at about the same efficiency.

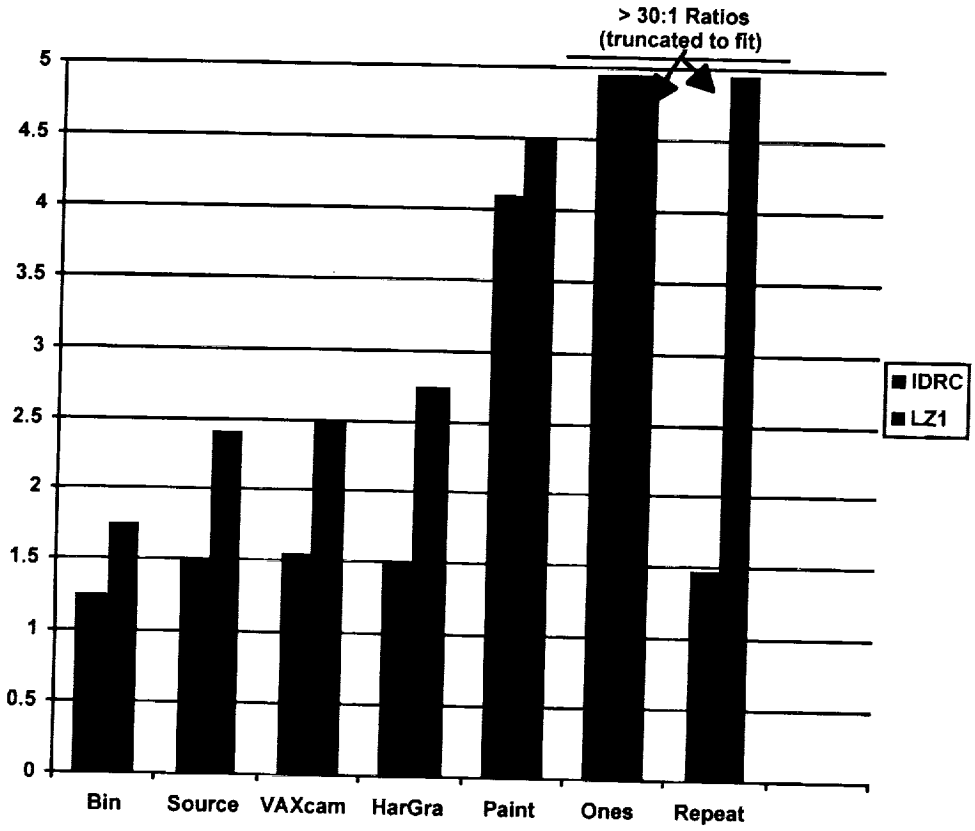


Figure 7: SDS-3 Data Compression Results

The lab test results showed that, on average, the LZ1 efficiency was at a 2.5:1 ratio vs. the IDRC's 1.5:1 ratio. Quantum Corp. has a white paper available that provides the details of these tests.

The DLT design supports both data compression and compaction. The advantage of the compaction algorithm is that there is no loss of recording space on the tape. Even if the file ends anywhere within a given entity (see Figure 5), the first record of a new file will begin immediately after the end of the previous file without any loss of media space. The drive automatically regenerates the ECC algorithm to cover the new information within the entity.

To summarize, the description of the major design areas of the DLT given above, although brief, exemplify a product designed for maximum reliability. The very gentle head-to-media interface (HGA design), the self-cleaning properties of the head, the extensive use of adaptive techniques, and the very long media and head life tests under extreme environmental conditions, all contribute to the reliability and robustness of the DLT products. Using the "HP Method" of recording failures in the field (i.e., all types of failures are taken into account during a power on period of 24 hours a day, seven days a week), the DLT products are exceeding their specified mean time between failure (MTBF) rate of 80,000 hours by 25%, independent of duty cycle. Two of the major contributors to this field MTBF performance are the 10,000 hour head life and media durability.

THE DLT PRODUCT FAMILY AND APPLICATIONS

Until the last couple of years, the primary function of the tape drive has been to backup and archive data. As data storage requirements have been increasing at an almost exponential rate, the need for a balance of capacity and performance has become much more critical. With 20GB of native data recorded on a single DLT4000 cartridge, the user needs to transfer the data in the shortest time possible. That is the reason the DLT product family emphasizes overall performance as much as capacity per cartridge. Figure 8 shows the DLT Product Family (second generation), beginning with the DLT260, which was first introduced in November, 1991, as a product aimed at the OEM market. The DLT260 was followed by the DLT600 in mid-1992, and the DLT2000 (our current high volume product), introduced in the third quarter of 1993. The DLT2000, with a capacity of 10GB per cartridge (native) and 1.25MB/sec. (native), is today's industry leader for this class of products. The new DLT4000, with volume production starting early in calendar 1994, stretches the DLT technology leadership that much further (note especially the considerable improvements in load times)..

	DLT260	DLT600	DLT2000	DLT4000
Data Rate (MB/s, Native)	.800	.800	1.25	1.5
Capacity (GB, Native)	2.6	6	10	20
Bit Density (bpi)	42,500	42,500	62,500	82,500
Track Density (tpi)	96	224	256	256
Media Type	MP-1	MP-1	MP-1	MP-2
Media Length (in feet)	1100	1100	1100	1700
Recording Channels	2	2	2	2
Data Compression	No	No	Yes	Yes
Load Time (in seconds)	60	60	45	33

Figure 8: Quantum DLT Drive Product Family

By intent, the transition from the DLT2000 to the DLT4000 product was an evolutionary development effort. As Figure 8 shows, the primary changes were the combination of the thinner MP media (MP2) and a higher bit density (82.5 KBPI). Minor modifications to the head were necessary, as well as the incorporation of a flex circuit containing the read pre-amp in much closer proximity to the head. Specifically, the head core geometry was slightly changed, but the contour and all other electrical and mechanical parameters remained fairly close to the DLT2000 configurations.

Quantum offers not only the drive itself: The DLT product family includes a 7-cartridge loader (half-rack form-factor, for rackmount applications) and a compact 5-cartridge loader designed for table-top applications. The design concept for the two loaders has been to enable the replacement of the drive only (inside the loaders) in the field by a skilled technician.

In addition to loaders, a number of third-party robotics companies have announced and are shipping both large and small library configurations. Figure 9 shows the vendors that offer libraries for the DLT family of products. It has become clear in the marketplace that the primary growth in the tape industry is in these library configurations. A number of tape manufacturers of various technology products (DAT, 8mm, 3480/90, etc.) today offer an assortment of library products extending from 28 cartridges to 900 cartridge robots. The library vendor differentiation is in terms of the number of cartridges in the library and the ratio of cartridges to drives that the robot can handle. A number of additional library vendors are developing DLT-based products, anticipating even more capacity and higher performance DLT products.

COMPANY	LIBRARY	DLT PRODUCT	DESCRIPTION	CAPACITY*
Quantum	DLT2500	DLT2000	5 Cart.Loader, 1 drive	50 GB
	DLT2700	DLT2000	7 Cart.Loader, 1 drive	70 GB
	DLT4500	DLT4000	5 Cart.Loader, 1 drive	100 GB
	DLT4700	DLT4000	7 Cart.Loader, 1 drive	140 GB
ATL/Odetics	ACL2640	DLT2000	264 Cartridge Library, 3 drives	2.64 TB
Breece Hill Technology	Q7	DLT2000	28 Cartridge Library, 3 drives	280 GB
	Q47	DLT2000	60 Cartridge Library, 2 - 4 drives	600 GB
Digital Equipment Corporation	StorageWorks TL820	DLT2000	264 Cartridge Library, 3 drives	2.6 TB
Overland Data	DLT Multilibrary	DLT2000	24 - 120 Cartridge Library, 1 - 8 drives	240 GB - 1.2 TB
	DLT TA200 Tape Array Subsystem	DLT2000		100 GB
Metrum	D900	DLT2000	900 Cartridge Library, 20 drives	9 TB
	D360	DLT2000	360 Cartridge Library, 8 drives	3.6 TB
	D480	DLT2000	480 Cartridges (add on to D360)	
	D28	DLT2000	28 Cartridge Library, 4 drives	280 GB
	D60	DLT2000	60 Cartridge Library, 2 - 4 drives	600 GB

* All capacities are native.

Figure 9: Quantum DLT Drive-Based Tape Libraries

COMPANY	LIBRARY	DLT PRODUCT	DESCRIPTION	CAPACITY*
ADL - Media Logic	SLA-Dbase	DLT2000/4000	1 - 2 drives, 7 or 14 cartridges	70 GB - 280 GB
	SLA-Dplus	DLT2000/4000	2 - 4 drives, 7/14/26 cartridges	70 GB - 520 GB
	SLA-Dmax	DLT2000/4000	2 - 7 drives, 7/14/26/50 cartridges	70 GB - 1.0 TB
ADIC (Applied Digital Inter. Corp)	N/A	DLT2000	1 - 8 drives, 24 - 120 cartridges (12 cartridges per magazine)	240 GB - 1.2 TB
APP/Grau	ABBA/2	DLT2000	Up to 100,000 cartridges (mixed media environment)	
	ABBA/E	DLT2000	Up to 12,000 cartridges (mixed media environment)	

* All capacities are native.

Figure 9: Quantum DLT Drive-Based Tape Libraries (Continued)

The increasing popularity of the DLT has led an impressive list of third-party software vendors to support the DLT family of products and options. Figure 10 shows a partial list of software companies that support DLT options under all the major operating systems platforms. There is a considerable list of additional software companies who are currently developing support for the DLT drives and options (in both Loader and Library configurations).

Operating System	**Application Software**
Netware	Avail 2.0 Cheyenne Arcserve NLM 4.02 & 5.01E Systems Enhancement V 1.95 Novastor Novanet Arcada HSM, Backup Exec Palindrome Backup 3.1, Network 3.1, HSM 3.1
Windows 3.1	Cheyenne ARCsolo for Windows Novastor Backup for Windows
Windows NT	Arcada Backup Exec for NT Microsoft Backup (NT 3.5)
DOS	Cheyenne ARCsolo for DOS Novastor Backup for DOS Palindrome Director for DOS 3.1 Palindrome Backup Network Archivist for DOS 3.1
OS/2	Novastor Backup for OS/2 Legato (SUNOS 4.1, Solaris 2.3, RS6000 AIX) Novastor SGI/IRIS>5.X, RS6000 AIX, ATT/GIS System 5 Novastor (Sun Solaris 2.3, HP9000/400/700, SunOS 4.1, .2, .3 Cheyenne ARCserve/open 2.0 (Solaris 2.3, RS6000 AIX)
HP-UX, AIX	Workstation Solutions/Quick Restore
SCO UNIX	SCO STP Driver
Apple System 7	Dantz (Retrospect 2.1A/Retrospect DLT Driver) Novastor NovaMac

Figure 10: DLT Software Connectivity Matrix (Partial List, Valid as of 11/30/94)

For those proprietary platforms into which DLT products have not yet been integrated by the systems manufacturer, a number of Value Added Resellers (VARs) and Systems Integrators have developed and offer emulations that enable the DLT products to be attached to these proprietary busses as well.

Considering that the popularity of Hierarchical Storage Management (HSM) is increasing (based on the premise that the data storage requirements at the system level are increasing dramatically every year), it is probably worth discussing the DLT's potential for making HSM work most efficiently and economically [6].

In a recent article on HSM in the "Client/Server Today" magazine (December, 1994), David Simpson states that "HSM has the potential to dramatically reduce storage costs and management hassles by migrating infrequently accessed or inactive files from expensive disk drives to less expensive storage devices." The value of HSM is that this migration happens automatically and is transparent to the user. To differentiate between the various HSM packages available in the market today, Peripheral Strategies developed a set of definitions for the five levels of HSM software. Depending on the application, a user can select the particular HSM level that incorporates the various storage devices and/or technologies most suited for use with the range of data.

DLT products are ideally suited to become the products of choice for HSM applications. They offer the highest capacity and performance combination in industry today for their class of products. In addition, a number of software companies offer HSM software support for the DLT products. In December, 1994, Cheyenne Software, a leading supplier of software products for all major systems platforms, announced support for the DLT2000 drives within its new HSM program in addition to on all its other software platforms via the company's ArcServe and ArcSolo software packages. Avail Systems, Arcada Software Inc., Axent Technologies, Epoch Systems, Legato System Inc., Novastor Corp. and Systems Enhancements Inc. have also announced DLT support on their HSM solutions as well as on their other software platforms.

The next step in the DLT family development is another 5.25 inch form-factor product with higher native capacity per cartridge and substantially higher native performance. This new product will use the DLT4000 cartridge and will, of course, be read/write compatible with the previous members of the DLT Family. This new product will be announced in the second half of 1995.

For future family members, the DLT Development Group is planning to take advantage of all head and media technologies that other tape manufacturers who are using smaller form-factor cartridges are bringing to market to keep up with the constantly increasing demand for much higher capacities and increased performance. Because of the physical dimenstions of its cartridge and the cartridge's designed-in ability to incorporate more tape, the DLT engineering team can continue to offer industry-leadership storage capacity and products, always able to embrace the advances other manufacturers make in tape media and head technologies.

Our DLT development team plans to take advantage of thin-film-head technology for its multi-channel head requirements. Metal Evaporated (ME) tape and/or Barium Ferrite (BaFe) tape technologies, which are now being developed for the QIC and 8mm applications, also show some potential for use in DLT technology. Our product map calls for products to be developed with 100GB of native capacity per cartridge with 10-15 MB/sec. native transfer rates by the end of this decade. At this point, we intend to continue to maintain, at a minimum, read compatibility with all prior generation DLT products.

Summary

DLT is a mature and robust technology that has finally been "discovered" by the computer market because it offers the capacities, performance, and reliability that today's systems applications require. There is no other technology or product set available today that offers so balanced a combination of capacity and performance with leadership in data integrity and overall product reliability. With those strengths and their cost of ownership, these products are the best tape storage solution in the industry.

Advancements in DLT technology guarantee that new family members will continue to be developed for the balance of this decade and well into the next. Unless a new technology emerges to obsolete tape media products, DLT will continue to be the tape industry leader for this class of products.

References

1. Articles ("Surveying the Highs and Lows of HSM" and "HSM Brings Back Quick Returns," written by David Simpson, issue of *Client/Server Today,* p 53, December, 1994.
2. George Saliba "A Tape Drive Architecture Of A Robust Technology That Delivers Very High Reliability and Data Integrity As Well As Very High Capacity And Performance." White Paper, February, 1993.
3. George Saliba/Kumar Kasetty "DLT HGA Tracking Capabilities" White Paper, May, 1992
4. George Saliba "DLT2000 Adaptive Calibrations" White Paper, circa 1993
5. David C. Cressman "Why Does the DLT2000 Tape Product use Lempel-Ziv Data Compression?" *Digital Technical Journal,* **6** (1993) 62.
6. Fred Richardson, "Cost of Ownership in Hierarchical Storage Management," *Computer Technology Review* (Spring/Summer, 1994), p29.

The MAMMOTH Project

Tim Gerchar
Senior Product Manager
EXABYTE, Corp.
1685 38th Street
Boulder CO 80301
Phone: +1 (303) 447-7342 FAX: +1 (303) 447-7501
e-mail: timger@exabyte.com

What is MAMMOTH?

On the surface MAMMOTH is a high performance 5.25-inch half-high 8mm helical scan tape drive that records a native 20 Gigabytes of data on Advanced Metal Evaporated media at a sustained throughput of 3 Megabyte per second over a high speed SCSI interface, that is scheduled for production in the second half of 1995. But it's much more than that. Inside its custom designed sheet metal enclosure lies one of the greatest technical achievements of its kind. Exabyte's strategic direction is to increase throughput and capacity while continuing to improve drive, data and media reliability of its products. MAMMOTH adheres to that direction and the description of its technical advances is described in this paper.

MAMMOTH can be broken down into four main functional assemblies: high-performance integrated digital electronics, high-reliability tape transport mechanism, high-performance scanner, and advanced metal evaporated media. All this technology is packaged into a standard 5.25-inch half-high form factor that dissipates only 15 watts.

High Performance Integrated Digital Electronics

There has been some confusion in the industry over what commonality Exabyte's 8mm products and 8mm video technology share. The only similarity between 8mm video technology and MAMMOTH is the cartridge shell dimensions.

MAMMOTH employs a single processor design anchored by the AMD 29200 processor. The top-down design methodology of MAMMOTH results in a highly-integrated system that includes seven unique custom ASICs. The low parts count lends itself to a highly reliable system. The electronics are surface mounted onto three printed circuit boards. The MRF (MAMMOTH Rigid Flex) is an 'L' shaped board that contains the processor and its supporting circuitry. The digital data path section which contains two large digital ASICs, RAM and its supporting circuitry are also mounted on the MRF. Also contained on the MRF are the electronics for servo control, which include driver ICs and a custom mixed signal ASIC. The MRC (MAMMOTH Read Channel) is a rigid board that supports the read and write operations of the drive. It contains custom ASICs and all of the discrete filter functions. The MAMMOTH SCSI interface offers a configuration of one of four different SCSI variants; single ended narrow (8 bits) and wide (16 bits), differential narrow (8 bits) and wide (16 bits).

Firmware design is accomplished by the use of 'C' code wherever possible. The code is stored in EEPROM that is programmable, by the use of a code load tape, over the SCSI interface, or with an Exabyte proprietary diagnostic program. The code is designed with an eye towards the future; many of the SCSI 3 features already exist and the layered firmware allows for easy migration to other interfaces such as serial SCSI and fiber channel. The SCSI code is a special area of focus for the MAMMOTH designers. It is optimized for high performance, minimal bus hogging and improved error recovery. This will provide for fast average and predictable worst-case timing values.

In keeping with EXABYTE's strategic direction of constantly improving reliability, the engineering design criteria has been more stringent than the product specifications allow. These criteria include such things as power supply margins, operating temperatures and component tolerances. All this adds up to high performance, highly reliable tape drive electronics and firmware.

High Reliability Tape Transport Mechanism

MAMMOTH's tape transport mechanism can be broken down into three subassemblies: the solid aluminum deck casting, the innovative capstanless design and the cartridge loading mechanism.

Exabyte's MAMMOTH drive uses a one-piece aluminum deck casting with a belt-drive system to load the tape path. The one-piece deck casting affords a very high degree of rigidity and precision tolerances for greater reliability. The belt drive tape loader mechanism that operates the two streamlined trolleys uses an angled motor/worm shaft to optimize gear mesh and reduce axial loads. The integrated overdrive springs eliminate any timing errors between the trolleys. The trolleys have been designed to hold very tight tolerances by insert molding the guide pins into the arms. The use of large 6mm tape guides and rollers aid in producing a very simplified and low stress tape path. The tape path can be manually unloaded without damaging the media in an emergency situation.

The innovative capstanless reel-to-reel design used in the MAMMOTH tape transport mechanism incorporates the circuitry for the supply and take-up motor controls and a custom motor driver chip. This is all packaged on a rigid flex mounted on a metal base plate to minimize interconnection and electrical noise. The take up motor's gear ratio allows accurate speed control at low tape speed. This ratio also provides for increased efficiency and low power consumption. The supply motor tightly maintains tape tension through a closed loop control system. The assembly is designed to effectively handle not only component tolerances but cartridge tolerances such as hub roundness.

The capstanless design used in MAMMOTH provides many benefits. Among those are minimized edge damage by using fewer edge guides and lower edge forces when recording on long thin smooth tapes such as AME, tape life is also extended by using fewer large diameter guides. The capstanless design also provides for faster backhitches, improved high speed search performance and faster load to ready times. By removing the capstan debris is not pressed into the media by the pinch roller, also there is less debris generated in the capstanless system.

Integrally mounted in the deck casting is the cartridge loader. It has been designed with a sturdy metal frame for smooth quiet motion of the cartridge, and doesn't allow the cartridge to be misloaded in the drive. The cartridge loader can also be manually operated for media removal without damage. The cartridge loader was designed for fast load/unload times

which are required in an automated environment. This allows for complete and simple library integration without modification of the drive.

The deck assembly is shock mounted to the three piece sheet metal enclosure to help in isolating the deck casting and tape path from the host system's enclosure. The electronics are mounted along the side and rear top of the casting to help in cooling and prevent particulate contamination from entering the tape path. The sheet metal enclosure has been designed to facilitate cooling while minimizing any susceptibility to external radiating sources. The SCSI interface is easily changed by removing one screw to remove the sheet metal cover.

All of the tape transport mechanism design features add up to provide a low stress tape path, tight control over tape speed and a library-ready cartridge loader which in turn offers not only a highly-reliable tape drive but one which also extends media reliability. The design also affords a simple, reliable and predictable manufacturing process.

High Performance 8mm Rotating Scanner

With the purchase of the Grundig scanner division, now known as EMG (Exabyte Magnetics Gmbh) Exabyte now controls another piece of the core technology required to effectively compete in the tape drive industry. Exabyte had been working with that division for more than two years before the acquisition to develop the high-performance scanner used in the MAMMOTH product.

The 47mm scanner was designed to maximize the utilization of tape area. The scanner's rotational speed of more than 5600 RPM, along with a proprietary upper drum design, provides precise airfilm control over the 4 dual azimuth read/write heads throughout the scan. The high head-to-tape speed allows the drive to easily attain the 3 MB/sec. sustainable transfer rate while reading and writing the MAMMOTH format. The 4 dual azimuth read/write heads employ the same type of read-after-write strategy that has been an EXABYTE trademark since the EXB-8200, Exabyte's first product. Head design and media characteristics combine to give the long head life Exabyte's 8mm products have traditionally enjoyed. The motor technology that drives the scanner is a three-phase brushless DC motor.

One of the very innovative features of the MAMMOTH scanner is the rotary transformer. This advanced proprietary transformer technology affords a much higher coupling coefficient than previous designs. This coupling coefficient not only makes the high data rates for MAMMOTH achievable but allows for easy migration to higher performance follow-on products. The transformer configuration also allows for excellent noise isolation to boost the SNR.

All of the design features incorporated in Exabyte's MAMMOTH scanner will provide for high data reliability while achieving a very high level of performance in an easy-to-manufacture product.

Advanced Metal Evaporated Media

MAMMOTH will utilize 160 meters of a high-performance advanced metal evaporated media to store 20 Gigabytes at a rate of 3 Megabytes per second. MAMMOTH will be able to read 8200, 8500, and the 8500c formats of the 8mm Metal Particle tape recorded by previous Exabyte 8mm tape drives.

The AME media is being developed by SONY Corporation in concert with the drive mechanism. The development goal is to meet or beat all previous Exabyte reliability specifications. In our internal testing the durability and archivability of the media are meeting and or exceeding expectations. That results in an initial specification of at least 1500 passes, and a storage life of at least 30 years. Exabyte is very confident that the media will meet expectations due to all of the design features built into the MAMMOTH tape transport mechanism.

Summary

MAMMOTH accomplishes Exabyte's strategic direction of increasing throughput, performance, and capacity while improving reliability by utilizing design features such as high-reliability tape transport system, high-performance digital electronics, a high-performance scanner, and the use of AME media.

It furthers Exabyte's commitment to the tape industry by extending 8mm technology. The first Exabyte product, the EXB-8200, was first produced in 1987. It gave the tape industry the shot in the arm that has brought about a multitude of new products and technological advances in the existing technologies. Exabyte followed the 2.5GB EXB-8200, that when released was specified at 20,000 mean time between failure hours, with the 5GB EXB-8500 in 1990. The EXB-8500 had a transfer rate of 500 KB/s and a MTBF specification of 40,000 hours, in that same year the EXB-8200's MTBF specification was doubled to 40,000 hours. In 1992 Exabyte introduced its second generation of products which were half high versions of the EXB-8200 and EXB-8500. The EXB-8205/8505 were designed in a co-developement with our deck supplier to provide additional drive and media reliability. As a result the EXB-8205/8505 were released with double the MTBF specification: 80,000 hours. Exabyte has recently released the 'XL' versions of the EXB-8205/8505 that again double the MTBF specification to 160,000 hours, and extend the capacity to 3 and 7 GB respectively. Along with the drive reliability specifications doubling, the head life expectation also has been increased to at least 16,000 hours.

I equate the MAMMOTH tape drive to the EXB-8200 - - it will also cause a resurgence of tape technology being utilized in many of the non-traditional tape applications such as video-on-demand and hierarchical storage management systems. MAMMOTH also establishes a new level of performance and reliability that is directly due to the technological advances described above. It will initially have reliability of at least 200,000 hour MTBF, and a large library population ready to upgrade to its capacity, performance and reliability level.

Influence of Technology on Magnetic Tape Storage Device Characteristics

John J. Gniewek and Stephen M. Vogel
IBM Corporation
9000 S. Rita Road
Tucson, Arizona 85744
(602) 799-2390
Fax: (602) 799-3665
jgniewek@vnet.ibm.com
svogel@vnet.ibm.com

Introduction

There are available today many data storage devices that serve the diverse application requirements of the consumer, professional entertainment, and computer data processing industries. Storage technologies include semiconductors, several varieties of optical disk, optical tape, magnetic disk, and many varieties of magnetic tape. In some cases, devices are developed with specific characteristics to meet specific application requirements. In other cases, an existing storage device is modified and adapted to a different application. For magnetic tape storage devices, examples of the former case are 3480/3490 and QIC device types developed for the high end and low end segments of the data processing industry respectively, VHS, Beta, and 8 mm formats developed for consumer video applications, and D-1, D-2, D-3 formats developed for professional video applications. Examples of modified and adapted devices include 4 mm, 8 mm, 12.7 mm and 19 mm computer data storage devices derived from consumer and professional audio and video applications.

With the conversion of the consumer and professional entertainment industries from analog to digital storage and signal processing, there have been increasing references to the "convergence" of the computer data processing and entertainment industry technologies. There has yet to be seen, however, any evidence of convergence of data storage device types. There are several reasons for this. The diversity of application requirements results in varying degrees of importance for each of the tape storage device characteristics listed in Table 1.

Table 1
Tape Storage Device Characteristics

- Reliability
  - Data Reliability
  - Device Reliability
- Procurement Cost
- Operating and Maintenance Cost
- Access Time to Data
- Data Rate
- Automation Compatibility
- Capacity
  - Cartridge Capacity
  - Automation Capacity
- Write/Read Ratio
- Form Factor

This diversity of requirements has continually reinforced the need for an economical storage hierarchy. Continuing advances in technology have enabled the development of new devices with enhanced capabilities. The acceptance of new devices is tempered, however, by the investment most users have in existing tape storage volumes. For removable tape storage systems, this therefore presents a dilemma. Significant (perhaps 5-10X) rather than incremental improvements in storage cost-performance assessments must be offered to make the conversion to a new system attractive. However, in order to obtain order of magnitude improvements, it is usually necessary to introduce significant changes in the technology components that may prevent direct device compatibility with previous devices in an economical manner. In this respect, removable media storage systems which often hold large quantities of archival data are unique compared to other storage devices and components such as semiconductor memory or magnetic disk storage.

This paper discusses the device attributes that may be obtained by using advanced technology components in an embodiment that is deemed most suitable for computer data storage applications requiring high reliability, flexibility of data processing operations, economical storage costs, economical maintenance and operations costs, and data rate parity with other members of the storage hierarchy.

Storage Hierarchy

Storage hierarchies exist primarily for economical reasons. In the extreme limit of unbounded advances in the price/performance characteristics of semiconductor memory or hard disk storage and electronic data transfer network technology, it might be concluded that the need for tape storage devices and removable tape media would almost entirely disappear. In almost all cases hard disk device characteristics would be preferred. In actual fact, however, in spite of the advances in both semiconductor and hard disk capabilities, there are five basic reasons why tape storage will remain an important member of the computer data storage hierarchy.

1) Magnetic tape storage will remain significantly less expensive than hard disk storage. While lower cost per storage will be a continuing trend for all technologies, it is expected that the cost ratios will remain intact.

2) The volumetric density of tape storage relative to other storage technologies will, for fundamental reasons, always be a large ratio.

3) With each advance in technology, the demand for data storage increases. Improved storage devices enable new applications that previously were not economical and this, in turn, leads to increased demand for additional storage.

4) Software-managed automated removable media storage libraries continue to evolve and will be common for all applications. With this in place, optimally-designed tape storage devices will provide a continuum of storage characteristics along with semiconductor memory, hard disk and optical disk storage.

5) Although significant advances in electronic data transfer communication networks can be expected in the next decade, because of band width limitations and telecommunication costs, data interchange via physical transport of removable media volumes will remain the most economical procedure for many applications.

Tape Storage Device Attributes

Because of the wide diversity of application requirements, there will undoubtedly continue to be several types of devices required in order to meet all needs economically. The best a user could hope for would be to reduce the number of types of devices that need to be supported. Setting this as a design goal, the development objective becomes one of utilizing advanced technology components in a design(s) that attempts to provide: 1) the greatest flexibility of uses; 2) at an affordable price; 3) without compromising reliability objectives and at 4) performance matched to system requirements.

The IBM 3480 technology introduced in 1985 has become the industry standard for high performance computer data storage users. The 3480 and the 3490 and 3490E follow-on devices have developed a well-deserved reputation for providing highly reliable operations. Those attributes, high performance and high reliability, were responsible for its widespread industry acceptance. In providing the technology base for the next generation of tape storage devices, it is desirable to build upon the strengths of the 3480/3490 class of devices and enhance those factors that are necessary to achieve the development objectives described earlier.

Analysis of the 3480/3490 device design reveals several key design features that contribute to the performance and reliability reputation that has been achieved. These features are as follows:

1) An enclosed Cartridge System Tape (CST) that prevents accumulation of handling damage, fingerprints, airborne debris, etc.

2) A gentle tape path and tape guiding system that minimizes or eliminates mechanical tape damage.

3) An 18 track linear recording technology using thin film ferrite heads with magneto-resistive (MR) read elements. The 3490E utilizes serpentine linear recording at 2X track density with second generation MR read elements and enhanced Error Correction Code (ECC). The format is arranged such that when writing full tapes, no tape rewind is required.

4) An ECC that utilizes the advantages of multiple track recording in a track format that minimizes the probability of concurrent track errors due to media defects.

5) A Head-Tape-Interface (HTI) that operates at low pressures ensuring low head wear rates, yet provides the closely controlled spacing required for data reliability.

6) A tape media with mechanically and chemically stable polymer binder system, magnetic particles and substrate that minimizes debris generation and provides stable tape motion.

7) Reel-to-Reel servo control for precise tension and velocity control. In combination with the fixed head design and an electronic buffer, both start-stop and streaming data recording/reading are supported without performance or reliability penalties.

These are laudable design features proven in almost 10 years of field operation. Incremental improvements have been introduced during this time as the cartridge capacity increased from 200 MB to 400 MB to 800 MB, uncompressed. Additionally, IDRC compression was introduced along with channel data rate enhancements and cost reductions. The ability to use lossless data compression, such as IDRC, provides benefits to both effective capacity and effective data rate. Because data compression ratios can be highly variable, it is very possible that the storage device can be driven into start-stop mode if the channel data rate becomes the gating factor. Hence, to get full benefit of data compression features it is highly desirable to have a storage device that operates in both start-stop and streaming modes.

As a result of numerous discussions with numerous customers, the following items were identified as desired improvements to the attributes of 3490E tape transports. This listing is from a diverse group of applications and is not to be interpreted that all items correspond to a single application. Using the 3490E experience as a base, an assessment was made of how advanced technology components could enable the achievement of the desired improvements listed in Table 2.

Table 2
Desired Improvements Relative to 3490E Technology

1) Higher Capacity
2) Higher Data Rate
3) Lower Cost
4) Smaller Form Factor
5) Maintain or Improve Reliability
6) Higher Drive Utilization
7) No Increase in Rewind Time
8) Faster Access to Data
9) Automation Compatible
10) Preservation of Automation Investment
11) Growth Path for Future Product Enhancement

Technology Factors

In discussing the influence of technology on magnetic tape storage device characteristics, it will be helpful to analyze the end objectives from the viewpoint of both the user parameters and the technologist/developer parameters. To this end, it is necessary to provide a translation between device functional parameters and base technology parameters. Such an analysis enables the developers of new devices utilizing new technology components to prioritize the required development activity in a manner that considers the interdependence of the various technology components. Table 3 illustrates this point.

Table 3

Translation Between End User Functional Parameters and Developer Technology Parameters

End User Parameters	Developer Parameters	Primary Technology Components
• Capacity (GB)	• Bit Areal Density (bit/mm^2) - Track Density (#/mm) - Linear Density (bit/mm) • Media Length • Recording Code	• Media • Heads • Head Positioning System
• Data Rate (MB/sec)	• Linear Density (bit/mm) • Head/Tape Velocity (m/sec) • Number of Concurrent Channels	• Heads • Media • Tape Path/Control
• Reliability a) Data Reliability (MBBE)[a]	• SNR Margin • Media Defect Level • ECC Design • Device ERP • HTI Stability	• Heads • Media •Tape Path/Control • Signal Processing Techniques/ Electronics
b) Device Reliability (MTBF)[b]	• Mechanical Design • Electrical Design	
• Access Time to Data	• Load/Thread Time • Search Time - Search Speed - Media Length	• Cartridge Loader Design • Cartridge Design • Tape Path/Control • Media
• Drive Utilization	• Load/Thread Time • Search Time • Read/Write Time • Rewind Time • Unload Time	• Cartridge Loader Design • Cartridge Design • Tape Path/Control • Media

(a) MBBE - Mean Bytes Between Error
(b) MTBF - Mean Time Between Failure

The End User Parameters - Capacity, Data Rate, and Reliability - need no elaboration. Access Time to Data and Drive Utilization, however, have importance beyond the obvious. These attributes serve as key elements in optimizing the price-performance of the complete storage subsystem, i.e. automation plus storage devices plus media. Different applications will provide different weighting factors to each of the sub-elements under these parameters. In the final analysis, a device with higher throughput (for whatever reason - data rate, search speed, load/unload time, etc.) requires fewer devices to achieve a given function. If there is not a cost penalty for such devices, this would then result in a lower total subsystem cost, hence a better price-performance rating. Applications such as "digital libraries" and network-attached HSM servers will be the major beneficiaries of such characteristics.

Technology Prototypes

In what follows, we will explain the methodology of the technology development process which began with choices for the various technology components, evolved with the assessment of their interoperability, and culminated with building and testing of prototype devices used both to validate the chosen operating points and to serve as an input to the product development decisions.

Technology Components

With reference to Table 3, there are a number of technology components that must be assessed and evaluated both on a stand-alone basis and in an integrated interactive environment. We identify the following as the key technology elements.

A) Media
B) Heads
C) Tape Path
D) Servo Systems
E) ECC
F) Device Electronics

A brief discussion of the factors involved in assessing the merits of each of the technology components is covered in the next section. In the following section we describe the technology elements to be incorporated into two different technology prototype devices and their resultant device characteristics. Finally, these characteristics are compared against the Desired Improvements listed in Table 2.

A) Media

The laws of physics determine the ultimate areal density that a recording medium can support. Numerous other factors influence the practical limits. Factors determining the ultimate recording density are identical for both disk and tape recording media. Practical limits, however, are significantly different due to significant differences in the other factors, such as track guiding, fly height, defect mapping stability, etc.

Because capacity and data rate are directly dependent upon the areal density capability of the recording medium, choice of the recording medium is of prime importance. Numerous investigations have assessed the recording density capability of various types of media [1] [2] [3]. It is generally agreed that a thin film medium of a few hundred Angstroms thickness with coercivity of ~1000 Oe provides superior areal density capability compared to any single-layer particulate recording medium. Thin film is the recording medium benchmark against which particulate medium improvements are measured. Indeed, current high performance high density hard disk recorders almost exclusively utilize thin film media. For saturate recording, which is

used in thin film disk recorders, the areal density capability is proportional to $H_c/(M_r t)$, where H_c is the coercive force of the medium, M_r is the remanent magnetization, and t is the media thickness [4]. Particulate tape media generally are utilized in a non-saturate recording mode where the coating thickness is usually greater than the recording depth. The effective recording depth is determined by the gap length of the recording head and the media coercivity. In this case, effective areal density can still be considered to be proportional to H_c/M_r. This explains the trend from ~350 Oe iron oxide recording media in the 1960s and 1970s to 550 Oe chromium oxide media and 900 Oe cobalt doped iron oxide media in the 1980s to 1500 Oe metal particle (MP) media in the 1990s. In addition to the increasing use of MP media, smaller quantities of barium ferrite media have appeared in floppy disk applications as well as thin film metal evaporated (ME) media for consumer video applications.

It should be cautioned that within a given media type, there are many variations possible, including many varieties of particle size, shape, and chemical composition, polymer chemical composition, mixing and coating processes, etc. In other words, all media of a given generic name type, e.g. MP, are not equivalent.

Another law of physics indicates that one parameter affecting the media signal-to-noise ratio (SNR) is the number of particles (n) in the recording volume in a manner such that SNR ~ [5]. This indicates that SNR can be improved through use of a medium with a larger number of smaller particles. Numerous other practical factors influence the final choice of particle size to be used.

Finally, it is realized that volumetric density is directly affected by the substrate [6]. Thinner substrates provide a higher volumetric density, however this desirable attribute must be balanced with durability and tape guiding issues, the latter factor obviously being dependent upon the tape path and the tape transport system.

B) Heads

Writing and reading data is accomplished via an intimate compatible relationship between the magnetic recording media and the magnetic recording/reading head. There are numerous design parameters that the head design engineer has at his disposal for optimizing the performance for a given type media. The head indeed plays a central role in providing a suitable transfer function between the write/read electronic circuits and the writing and reading of magnetic flux transitions in the recording media. In addition to the function of writing and reading of data, head design involves factors which influence a stable Head-Tape Interface (HTI) which is necessary for data reliability and tribology factors involving wear (of both head and media) and friction. For advanced recording systems, new media and head technology components are developed together to allow for intelligent design trade-offs and optimized system performance.

Write heads utilize an inductive coil of various designs coupled with a suitable magnetic pole tip material and write gap design that provides sufficient magnetic field strength to efficiently write flux reversals in the recording medium. IBM 3480/3490 technology utilizes a nickel-zinc ferrite (~3000 Gauss) material for the recording head pole pieces. While this is suitable for efficiently writing chromium oxide (H_C ~550 Oe) media used in 3480/3490 systems, this material is not capable of adequately writing 1500 Oe MP media. The ability to write higher coercivity media to get the benefits of higher areal density recording capability therefore requires the development of a new head design employing higher saturation magnetization pole tip material. A generic class of Metal-In-Gap (MIG) head designs has evolved for this reason. Most have been developed for rotary head type recorders. The ability to utilize MIG-type designs in a 3480/3490-like embodiment, i.e. stationary head, multi-track longitudinal recording, requires some unique features in both materials and fabrication processes.

Read heads may employ either inductive or magnetoresistive (MR) read elements. For inductive read elements, the signal amplitude is proportional to N d /dt, where N is the number of turns and d /dt is the rate of flux change. The rate of flux change is, in turn, proportional to the relative head-tape velocity. All rotary head recorders utilize inductive read heads. Unlike inductive heads, magnetoresistive heads provide a signal amplitude that is velocity independent. Furthermore, MR elements can be designed to provide high output signals with little required real estate or process complexity required for thin film inductive heads with a large number of turns. This is particularly important for 3490-like recording technology that employs 18 read and 18 write elements in each of the 2 modules required to make a recording head. The MR design enables a simplified thin film semiconductor-like process capable of fabricating the 36 elements per module economically. As track densities increase, the benefits of the MR design become indispensable. MR read elements were introduced in the 3480 product in 1985. A second generation improved MR design was introduced in 1991 with the double track density 3490E design. A new inductive thin film write head with third generation MR read head design was developed for the technology prototype devices described here.

C) Tape Path / Tape Transport

Magnetic tape storage recorders are electro-mechanical devices. The reliability of electro-mechanical devices is usually gated by the mechanical components. Hence, in order to achieve high reliability in a high performance computer data storage device, special attention must be paid to the mechanical portion of the tape path/tape transport design. Some of the requirements expected of the tape transport are reviewed in [7]. Suffice it to say that a reliable tape transport requires accurate velocity, tension, and tape guiding control while maintaining HTI stability and without damaging the media under repeated accesses in both write/read and high speed search/rewind modes. Features built into longitudinal recording stationary head devices enable highly reliable streaming or start-stop operations. These features include economical electronic buffers of sufficient size to totally mask the acceleration/deceleration times of tape movement.

For advanced operating point devices employing higher track densities, it is necessary to further improve the tape guiding envelope and, for interchange applications, the accuracy of the initial alignment of the head to the recorded tracks. As the advanced media and head components allow ever narrower tracks, the mechanical registration tolerances become the dominant issue limiting track density. At some point it becomes economically advantageous to introduce a head servo capability in lieu of ever more precise (and expensive) mechanical tolerances required to achieve the goal of increased track density and hence capacity.

D) Servo Systems

Servo systems of varying function are utilized in high performance tape storage devices. These functions include velocity and tension control in reel-to-reel driven transports such as 3480/3490 type, and synchronization of scanner speed with tape velocity for accurate tracking in various rotary head recorders. The use of a reel-to-reel tape transport servo system also enables a simple means of obtaining high-speed search capability.

While hard disk storage devices have employed active head positioning servo systems for many years as a means of reducing misregistration tolerances between the head and the written track, such active head positioning servo systems have yet to be employed in longitudinal recording format tape storage devices. Based on the discussion in the previous section, the ability to utilize head positioning servo systems in combination with accurate guiding tape transports, enables the attainment of higher track densities (hence higher capacities) without having to resort to longer length media and concomitant increases in search/rewind times.

E) Error Correction Code (ECC)

Error Correction Codes are employed to provide an uplift to the data reliability achievable from practically available media quality and device performance. Raw bit errors can be a result of a variety of causes, including a) local media defects such as coating contaminants, b) global media defects such as damaged edges, creases, adherent debris, c) adherent head contamination, d) transient non-adherent particulate debris at the head/tape interface, e) clocking errors due to velocity transients, and f) electronic noise.

The different types of error sources usually have a distinctive signature of raw errors which includes parameters such as error length, error reproducibility, error coincidence across multiple tracks, etc. The measured raw bit errors are of course critically dependent upon the data channels employed and the optimization of equalization and detection schemes for the variety of defect signals experienced.

In order to meet some end user data reliability target, the developer must first characterize and specify the media quality in terms of both average bit error rate (*ber*) and error distribution (error lengths and coincidence) as input to deciding which ECC scheme will meet the required objectives. It is meaningless to talk in terms of an average *ber* only without addressing the defect spatial and temporal distribution. A robust recorder design must also allow for possible increases in errors during either storage or heavy use of the media in order to achieve the desired performance objectives. Many claims that are made about corrected *ber* in various product advertisements do not define what is or is not included in the claim, frequently leading to "apples and oranges" comparisons.

The effectiveness of the ECC is intimately connected with the spatial distribution of the data format written on tape. Multi-track recording heads enable a more robust ECC since data is distributed laterally across the width of tape as well as linearly (temporally) along the length of tape. In systems utilizing a large number of concurrent channels, ECC robustness is enabled by reducing the probability of encountering extended concurrent defects of duration sufficient to defeat the ECC. This is extremely important for reliable data recovery from marginally recorded or degraded archive media. 3480/3490 devices utilize 18 concurrent channels. Data recovery is possible with 1 or 2 data read-back channels completely disabled. Obviously this would not be possible in 2 channel recording systems.

Thus multi-track recording formats can be used advantageously for enhancing data reliability. It is also apparent that performance (i.e. data rate) enhancements result from the use of multi-track formats. This ability to obtain both reliability and performance enhancements is what has been touted for RAID disk technology. In effect, tape storage devices have been utilizing these concepts for many years with the paradigm shift that the "Inexpensive Device" is the recording element in the multi-track head rather than a complete device. Indeed, because of the differences of tape devices compared to disk devices in achieving device synchronization and the stability of defects, we believe it is more practical to consider RAID type benefits for tape devices as occurring at the multi-track head level rather than schemes employing multiple devices. There is of course a cost associated with the benefits obtained by the use of multi-track recording technology. This includes a more expensive head and the additional cost of the additional read/write channel electronics.

F) Device Electronics

The increased density and decreased cost per circuit for semiconductor chips during the past decade has been nothing short of phenomenal. There is no indication that this progress will not continue. The power of using advanced recording media and heads combined with the advanced

semiconductor components enables significant recording device performance improvements while maintaining simple mechanical components (i.e. high mechanical reliability at low maintenance costs) and similar or reduced acquisition costs. These concurrent technology advancement trends have strongly influenced the decisions involved in the design, assembly and testing of the technology prototype devices.

Technology Prototype Devices

I) Technology Component Selection

Based on the analyses in the previous sections, the technology development team selected and developed the following technology components for incorporation into technology prototype devices.

A) Media - MP (1500 Oe class)

- Metal particle type chosen for optimal balance to meet both SNR and archival stability requirements
- Polymer binder system - uniquely developed to meet stringent performance/reliability requirements

B) Heads - multi-track linear recording

- Third generation magnetoresistive (MR) read elements
- Inductive thin film write elements
- New thin film shield/write pole tips materials/design needed to meet write performance and head-wear lifetime requirements

C) Tape Path - varies by technology prototype design

D) Servo Systems

- Reel-to-reel servo for velocity/tension control
- Active head positioning actuator/servo

E) ECC - Reed-Solomon

- Enhanced and scaled with areal density increases

F) Device Electronics - per performance and form factor objectives of technology prototype

Upon reviewing the eleven items listed as Desired Improvements in Table 2, it became apparent that a single prototype design would not be able to address all the items on the list. Therefore it was decided to build and test two different designs utilizing a common advanced technology base. In combination, the two different designs are able to address all eleven items.

II) Technology Prototype I

Of paramount importance in the selection criteria for the Prototype I design was preservation of automation investment. This requirement translated into the use of a 3480 CST type cartridge for

compatibility with the IBM 3494 and 3495 tape libraries. Use of a CST type cartridge would thereby provide for coexistence of 3490 and an advanced function drive type to enable both investment protection and migration capability to new technology.

Following this decision, it was decided to utilize the basic 3490 tape path since this design has been proven with approximately ten years of field experience since the introduction of 3480 in 1985. To obtain maximum benefit of the field experience, it was further decided to utilize approximately the same media thickness, and hence media length, as is utilized in the 3490E extended length cartridge. This ensures similar media mechanical properties favoring the establishment of a stable and reliable HTI with minimal development effort. Since the stated objectives were to increase capacity, including track density, early investigations were made to understand the various factors controlling the tape guiding envelope. The result of this has been to introduce into the tape path design subtle improvements to both the 3490 tape path and cartridge design that are designed to reduce the tape guiding excursions and therefore enable higher track densities.

Many media types were investigated with compatible read-write head designs to assess their storage capacity capability. Possible extensions of the 550 Oe chromium oxide media used in 3480/3490 devices were judged not to be of sufficient magnitude to lead to an attractive design point. MP media provided the best overall capability, but led to the requirement of being able to develop a compatible read-write head design capable of meeting all functional requirements including low wear and long operational lifetime. This was a major development checkpoint that, once achieved, committed the prototype design to MP media. Head and media were then co-developed to optimize their combined performance.

With continued co-development of head and MP media technology components, it was assessed that greater than an order of magnitude areal density improvement, relative to 3490E technology, could be obtained without an active head positioning actuator/servo technology. However, in order to provide for future capacity enhancements using the same cartridge, and to provide means to ensure data integrity and protect against neighboring track overwrite encroachment at high track densities, it was decided to incorporate servo tracks written on tape and to incorporate an active head positioning servo system. Such a system is new to linear tape recording systems, but borrows from the extensive technology developed for disk systems. The utilization of an active head positioning system reduces track misregistration (TMR) errors without the need for expensive, high precision mechanical components, and therefore enables the attainment of higher track densities.

Given these choices and the appropriate ECC and channel electronic circuitry, a design target of 10GB cartridge capacity with 9MB/sec data rate was established [8]. The 12.5X (relative to 800MB 3490E technology) capacity increase is obtained by operating at approximately 4X track density and 3X linear density. All values are uncompressed. It is judged that enhancement of the chosen technology components would provide for additional 2X-4X multipliers to both data rate and capacity without compromise of data reliability. Should the constraint of using the same media be removed, it is possible that even greater enhancements could be achieved.

Performance and capacity enhancements could always be obtained by reducing operating margins that relate to robustness of the system data reliability. The design point chosen for Prototype I and the expected possible extensions utilize the new technology components in a manner that does not compromise data reliability.

A comparison of the Prototype I design point to the list of eleven desired improvements indicates that seven of the eleven objectives are achieved. They are listed in Table 4.

Table 4
Prototype I
Functions Achieved Compared to Design Objectives

1)	Higher Capacity	10GB (0.8GB)
2)	Higher Data Rate	9MB/sec (3MB/sec)
5)	Maintain/Improve Reliability	Yes
7)	No increase in Rewind Time	Yes
9)	Automation Compatible	Yes
10)	Preservation of Automation Investment	Yes
11)	Growth Path	Yes

III) Technology Prototype II

Items 3, 4, 6, and 8 that were not achieved by the Prototype I design became key focal points in defining the design objectives for Technology Prototype II. The design was based on most of the same base technology elements, including the media and head, that were used in the Prototype I design. The major change that was made in the Prototype II design point involved the design of a new tape cartridge and tape path. Such changes were deemed necessary to achieve the design goals of items 3, 6, and 8. Several of the design objectives set for Prototype II are similar to those set in an earlier development effort [9].

For many of the emerging data storage applications involving network hierarchical storage management (HSM), digital libraries, and "parking garages on the information superhighway," current tape storage devices have several deficiencies. Key among the missing attributes is fast access time for data retrieval. There are two aspects to obtaining fast access time to data. The first is what may be termed the human factors aspect gauging user satisfaction against system response time. The second factor involves the price-performance aspects of a storage subsystem. A fast-response tape device with short rewind time leads to higher device utilization, i.e. the throughput rate per device is higher. This leads to fewer devices needed to perform the storage subsystem function and hence to overall lower storage subsystem costs.

Any tape storage device will still have orders of magnitude slower response time compared to a disk storage device, however the storage cost for tape will have a couple orders of magnitude advantage. This is enough incentive to employ a hierarchical storage system. The Prototype II design was developed to provide significant advantage over existing devices for these applications.

For both form factor reasons as well as access time and drive utilization reasons, it was desirable to have a high areal density recording technology. This would enable high capacity on a shorter length of media. Hence MP media, compatible head technology and active head positioning actuator/servo become the key enablers of such a prototype design. The next key design factor was the selection of a 2-reel cartridge with a self-contained tape path. This provided the ability to improve access time and drive utilization by not having to extract the tape from the cartridge in order to engage the head. Of equal importance, this design has the added benefit of improved mechanical reliability. By defining the Logical Beginning of Tape (LBOT) at the Physical Middle of Tape (PMOT) additional improvements are achieved in both access time and drive utilization.

A 5 1/4" form factor compliance was set as a goal, thereby setting an upper limit for the cartridge size. Other aspects of the objectives criteria refined the constraints on cartridge size further. Factors affecting ECC design, available electronic circuitry and data format were common with the Prototype I decisions.

Table 5 summarizes which of the desired objectives listed in Table 2 were achieved in the Prototype II design using the technology components previously described.

Table 5
Prototype II
Functions Achieved Compared to Design Objectives

1)	Higher Capacity	Yes 5GB (0.8GB)
2)	Higher Data Rate	No[a] 2.2MB/sec (3MB/sec)
3)	Lower Cost	Yes
4)	Smaller Form Factor	Yes
5)	Maintain/Improve Reliability	Yes
6)	Higher Drive Utilization	Yes
7)	No Increase in Rewind Time	Substantial Reduction 8 sec (30-50 sec)
8)	Faster Access to Data	Yes 8-10 sec (30-50 sec)
9)	Automation Compatible	Yes
10)	Preservation of Automation Investment	No[b]
11)	Growth Path for Future Enhancements	Yes

(a) Design is family compatible with higher data rate capabilities.
(b) Drive/Cartridge design enables compatibility with new high speed automation systems.

Conclusions

Advanced technology elements indeed enable advanced tape storage device capabilities. How such technology elements are utilized in particular device embodiments is highly dependent upon the application solutions that are targeted. The functions achieved in the Prototype I design were targeted to provide evolutionary, albeit saltatory, performance extensions to the 3480/3490 type products for their historical tape processing applications. Functions achieved in the Prototype II design are directed to providing solutions for a) cost-effective, lower performance historical applications and b) the putative new emerging applications. Both designs utilize a common technology base. This divergence in device designs is not unique in the storage industry. In the disk storage business, utilization of new technology capabilities has resulted in smaller disk files with higher capacity than their predecessor larger size disk products. The very reasons that provided those decisions for disk products, as opposed to simply increasing capacity on a large disk, will serve to guide the future direction of expected new tape storage devices. More than ever before, it is necessary to incorporate into the device design objectives, the performance objectives of the total storage subsystem rather than treating the device by itself.

Acknowledgements

The activity described here clearly represents the result of a development team effort composed of diverse technical skills. The authors acknowledge the contributions of these development team members, too numerous to credit individually, that contributed to the design, assembly and testing of the two prototypes.

References

1. Inaba, Hiroo, et al, "The Advantage of the Thin Magnetic Layer of a Metal Particulate Tape," *IEEE Trans on Magnetics*, **29** (1993) 3607.

2. Speliotis, Dennis E, "Double Layer Particulate Magnetic Recording Tapes," *IEEE Trans on Magnetics*, **29** (1993) 3613.

3. Williams, Peter, "Recent Developments in Particulate Recording Media," *IEEE Trans on Magnetics*, **24** (1988) 1876.

4. Middleton, Barry K, "The Recording and Reproducing Processes," in *Magnetic Recording, Volume I: Technology*, C. Denis Mee and Eric D Daniel, ed. (New York: McGraw-Hill, 1987).

5. Jorgensen, Finn. *The Complete Handbook of Magnetic Recording*, p220. (Blue Ridge Summit, PA: TAB Professional and Reference Books, 1980).

6. Goerlitz, W and Ito, A, "Substrates for Flexible Magnetic Recording Media: The Role of Base Films for Modern Performance requirements," *J Mag & Mag Mat*, **120** (1993) 76.

7. Cannon, Max R and Seger, Paul J, "Data Storage on Tape," in *Magnetic Recording, Volume II: Computer Data Storage*, C Denis Mee and Eric D Daniel, ed. (New York: McGraw-Hill, 1987).

8. Graves, David, "Capacity and Data Rate Advances in IBM Longitudinal Magnetic Tape Recording Technology," IEEE Symposium on Mass Storage, France (1994).

9. Doyle, William D, "A High Capacity, High Performance, Small Form Factor Magnetic Tape Storage Subsystem," *IEEE Trans on Magnetics*, **26** (1990) 2152.

APPROACHES TO 100 Gbit/in^2 RECORDING DENSITY

Mark H. Kryder
Engineering Research Center for Data Storage Systems
Carnegie Mellon University
Pittsburgh, PA 15213-3890

A recording density of 10 Gbit/sq. in. is being pursued by a number of companies and universities in the National Storage Industry Consortium. It is widely accepted that this goal will be achieved in the laboratory within a few years. In this paper approaches to achieving 100 Gbit/sq. in. storage densities are considered.

A major obstacle to continued scaling of magnetic recording to higher densities is that as the bit size is reduced, the grain size in the magnetic media must be reduced in order that media noise does not become so large that the signal to noise ratio (SNR) degrades sufficiently to make detection impossible (1). At 100 Gbit/sq. in., the bit size is only 0.006 square micrometers, which, in order to achieve 30 dB SNR, requires a grain size of about 2.5 nm. Such small grains are subject to thermal instability, and the recorded information will degrade over time unless the magnetic anisotropy of the materials used is increased significantly, or the media thickness is made much larger than expected on the basis of scaling today's longitudinal media thickness.

Perpendicular recording may enable one to use larger media thicknesses and therefore increase the volume of the grains, making it possible to overcome the thermal stability issues. However to record at such high densities onto perpendicular media will require that contact recording be used. Probe heads such as the Micro Flexhead(TM) components proposed by Censtor may provide a solution to this problem (2).

Another solution may be to use structured media in which the bit cells are defined by lithographic or otherwise created structure in the recording media. If the bit cells are defined, then each bit can be stored on a single particle, and instead of requiring 1000 grains per bit, it is possible that 1 grain per bit would be adequate. In this case recording densities as high as 10 Tbit/sq. in. would theoretically be thermally stable with today's materials.

Alternatively, bits could be recorded in the form of cylindrical domains in perpendicularly oriented, exchange-coupled magnetic media, like those used for magneto-optic recording today. With careful design of the magnetic parameters of such media, it is possible to balance the inward directed force of the domain wall surface tension against the outward directed force of the demagnetizing field. This produces a magnetic domain which is easily stabilized by moderate coercivity. Using near-field magneto-optic recording, domains have already been written and readback at a density of 45 GBit/sq. in. in such media (3). These domains have been shown to be thermally stable for several years in these media.

Whatever form of recording media is utilized, it is likely that some form of near-field magnetic or optical probe head will be required to record and playback the data. In order to achieve the desired resolution, the head will likely have to operate with a head-media spacing of less than 10 nm. Although it is too early to say for sure that such heads could not be "flown" above the media surface on a slider, it currently appears more likely that either the probe head would be run in contact with the media, or that some form of active feedback would need to be used to keep the probe head in close proximity to the media similarly to how feedback is used to control the head- media spacing in atomic force microscopes (AFM) today. If the AFM approach is used, then some means must be found to enable an adequate data rate. Theory and experiment indicate that, if a probe head is used with feedback, the data rate from a single head will be limited to a few megahertz (4).
One approach to achieving higher data rates would be to use an array of probe heads. L. R. Carley, et al. have been micromachining arrays of probe heads, actuators and control electronics for head positioning on a silicon wafer (5). This offers one approach to achieving the high data rates that are required.

In conclusion, storage densities of 10 Gbit/sq. in. are likely to be achieved with longitudinal recording; however, densities of 100 Gbit/sq. in. appear to require some changes in approach. Perpendicular recording, structured media and exchange coupled media all offer possible solutions to the thermal instability which is expected to result from too small a grain size. Because of resolution requirements, some form of probe head spaced less than 10 nm from the media is anticipated to be required.

1. Pu-Ling Lu and Stanley H. Charap, *IEEE Trans. Magnet.*, **30** (1994) 4230.
2. H. Hamilton, R. Anderson and K. Goodson, *IEEE Trans. Magnet.*, **27** (1991) 4921
3. R.E. Betzig, J.K. Trautman, R. Wolfe, E.M. Gyorgy, P.L. Finn, M.H. Kryder and C-H. Chang, *Appl. Phys. Lett.*, **61** (1992) 142
4. H.J. Mamin, L.S. Fan, S. Hoen and D. Rugar, "Micromechanical Data Storage with Ultra Low-Mass Cantilevers", Technical Digest of Solid State Sensor & Actuator Workshop, June 13-16, 1994, p. 17.
5. L.R. Carley, "Data Storage System Based on an Array of MEMS-Activated STM Tips" 1992-1993 DSSC Annual Report for Industry, Data Storage Systems Center, Carnegie Mellon University, 1993.

Reproducible Direct Exposure Environmental Testing of Metal-Based Magnetic Media

Paul J. Sides
Department of Chemical Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
ps7r@andrew.cmu.edu
Tel: 412 268 3846
Fax: 412 268 7139

Abstract

A flow geometry and flow rate for mixed flowing gas testing is proposed. Use of an impinging jet of humid polluted air can provide a uniform and reproducible exposure of coupons of metal-based magnetic media. Numerical analysis of the fluid flow and mass transfer in such as system has shown that samples confined within a distance equal to the nozzle radius on the surface of impingement are uniformly accessible to pollutants in the impinging gas phase. The critical factor is the nozzle height above the surface of impingement. In particular, the uniformity of exposure is less than +/- 2% for a volumetric flow rate of 1600 cm^3/minute total flow with the following specifications: For a one inch nozzle, the height of the nozzle opening above the stage should be 0.177 inches; for a 2 inch nozzle - 0.390 inches; for a 3 inch nozzle - 0.600 inches. The tolerance on these specifications is 0.010 inches. Not only is the distribution uniform, but one can calculate the maximum delivery rate of pollutants to the samples for comparison with the observed deterioration.

Introduction

Recording density and archivability are important characteristics of any data storage medium. Metal-based tapes such as metal particle (MP) and metal evaporated (ME) media perform well in recording but might, however, be vulnerable to corrosion.

Direct exposure to humid polluted air is a basic test of the archivability of a medium. For example, accelerated testing of MP media several years ago indicated some susceptibility to corrosion [1]. Although a tape cartridge or cassette must be considered as a "system" that affords protection of the tape by virtue of the spooling and incorporation into a package, the most basic line of defense against deterioration is the corrosion resistance of the media itself.

We have developed a corrosion test protocol based on the Battelle [2] flowing gas specifications but enhanced by the additional specification of a well-defined flow geometry, known as the "impinging jet", and flow rate. [3] We used it to expose coupons of MP [3] and ME [4] tape to variations of a Battelle Class II environment and found that some commercial MP media is very corrosion resistant while commercial ME media is vulnerable to corrosion on direct exposure. Numerical modeling of the flow has shown that, under certain circumstances to be discussed herein, the system possesses the useful property of uniform accessibility of the pollutants to the surface. In this paper, we report on the results of this modeling and discuss the standardization of direct exposure testing to eliminate the lab-to-lab variations that have been reported in the literature.

The fluid dynamics of the impinging jet

The impinging jet configuration, a variation of the stagnation point flow geometry, appears in Fig. 1. The incoming gas stream flows through a nozzle oriented perpendicularly to a nearby surface. The velocity profile of the jet changes from fully developed parabolic flow to free jet flow in the potential core, beyond which the centerline velocity of the jet decreases at a rate inversely proportional to the distance from the nozzle. The axial fluid velocity approaches zero in the region near the stagnation point. The flow in the body of the jet is axisymmetric, inviscid, irrotational, and the thickness of the boundary layer is insensitive to radial position. The flow is inviscid in the body of the jet because the vorticity is of the order of the inlet velocity divided by the radius of the jet, which is large with respect to the vorticity of the fluid in the boundary layer that grows from the stagnation point outward near the surface of impingement.

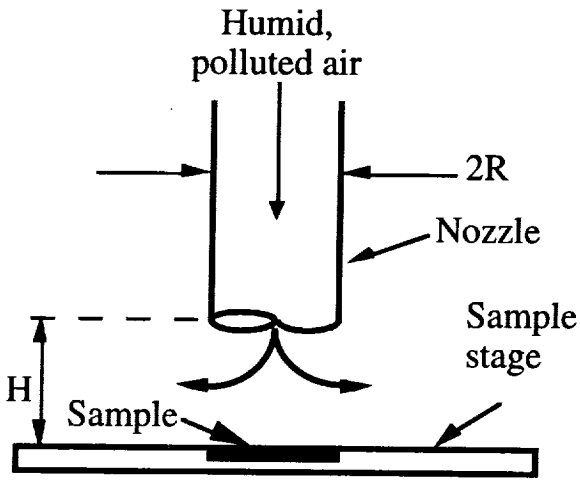


Figure 1. A schematic diagram of the impinging jet

Previous investigators have modeled the transport in the impinging jet reactor and experimented with it. The theoretical investigations can be divided into those that assume a uniform inlet flow and those that assume a well-developed parabolic flow. Homann [5] solved the one dimensional equations under the assumption of uniform accessibility. Chin and Tsang [6] surveyed the literature on the impinging jet and developed a semiempirical solution of an isothermal convective diffusion model in the form of an asymptotic series to give an estimate of the mass transfer rate for Schmidt number (Sc) > 0.7. Scholtz and Trass [7], examining mass transfer in a laminar impinging jet with a parabolic velocity profile at the inlet, measured velocity, pressure distributions, and the evaporation of naphthalene in a jet of air flowing at nozzle Re in the range 375 to 1970. The results of the experiments agreed satisfactorily with their analysis of the fluid flow and mass transfer equations under the nozzle. The rate and radial distribution were insensitive to nozzle height in the range 0.25 to 6 nozzle diameters. The mass transfer was uniform to the surface from the stagnation point to one fifth of the nozzle radius. The dependence of the mass transfer on radius from the axis was a strong function of nozzle height. For nozzle heights greater than half a radius, the mass transfer rate was a maximum at the axis and decreased radially. The mass transfer rate at the nozzle radius was approximately 80% of the rate at the stagnation point for nozzle heights of greater than one half the nozzle radius. For nozzle heights less than a third of the radius, the pattern was inverted; that is, the mass transfer rate was a minimum at the axis and increased radially. Snyder et al. [8] modeled the flow numerically and confirmed this inversion of the radial dependence of the mass transfer.

We used the impinging jet design in accelerated corrosion testing of magnetic media [3,4] because of its simple construction and high mass transfer rates but were concerned about the uniformity of the exposure. The inversion of the mass transfer rate dependence on the nozzle height, noticed by Scholtz and Trass [7] and illustrated in Fig. 2, presented an opportunity to improve the uniformity of the mass transfer to the surface under the jet. We sought to determine the radial profile of the mass transfer flux for various values of H/R between 0.3 and 0.5 in order to explore what happens when the profile changes from the one having a maximum at the axis to the profile having a minimum at the axis. The results of this modeling and their relation to the uniformity and rate of mass transfer of dilute components from a flowing mixed gas to coupons in an impinging jet chamber are presented in this contribution. The objective of the modeling was to determine conditions under which the uniformity of mass transfer to coupons would be optimal.

The Numerical Model of the Impinging Jet

A diagram of the domain appears in Fig. 2. A carrier gas and a reactive minor component enter the domain with a parabolic velocity profile and the flow emanates from a nozzle six to seven radii from the inlet. The gas impinges on a surface located at various distances from the nozzle and it exits along the open end of the domain. For the purposes of this study, the concentration of the minor component vanishes at the impingement surface (i.e. mass transfer control). This specification produces the maximum rate of delivery of the reacting species to the surface. The width of the domain was two radii. Axial symmetry in the problem allowed two dimensional representation and sectioning of the domain. Equations for the convective diffusion of mass and momentum in the impinging jet geometry governed the transport in the domain [7]. The dilute solution approximation was appropriate for this case. The boundary conditions were: (1) zero flux of mass and momentum through the nozzle wall and the reactor wall connected to the nozzle; (2) zero flux of momentum through the impingement surface; (3) finite concentration of the minor component at the inlet and (4) zero concentration at the surface of impingement.

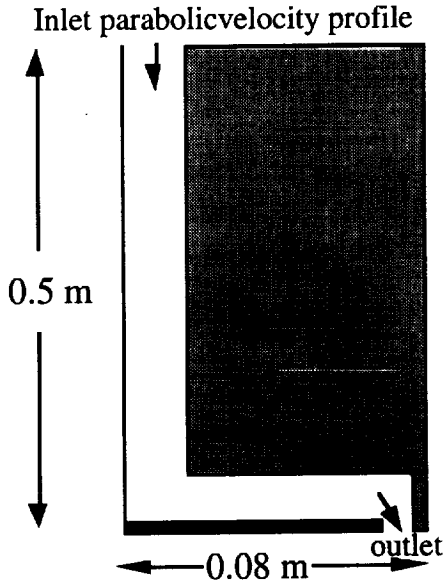


Figure 2. A schematic diagram of the domain of the numerical solution.

A diagram of the chamber used in the experiments [3,4] appears in Fig. 3.

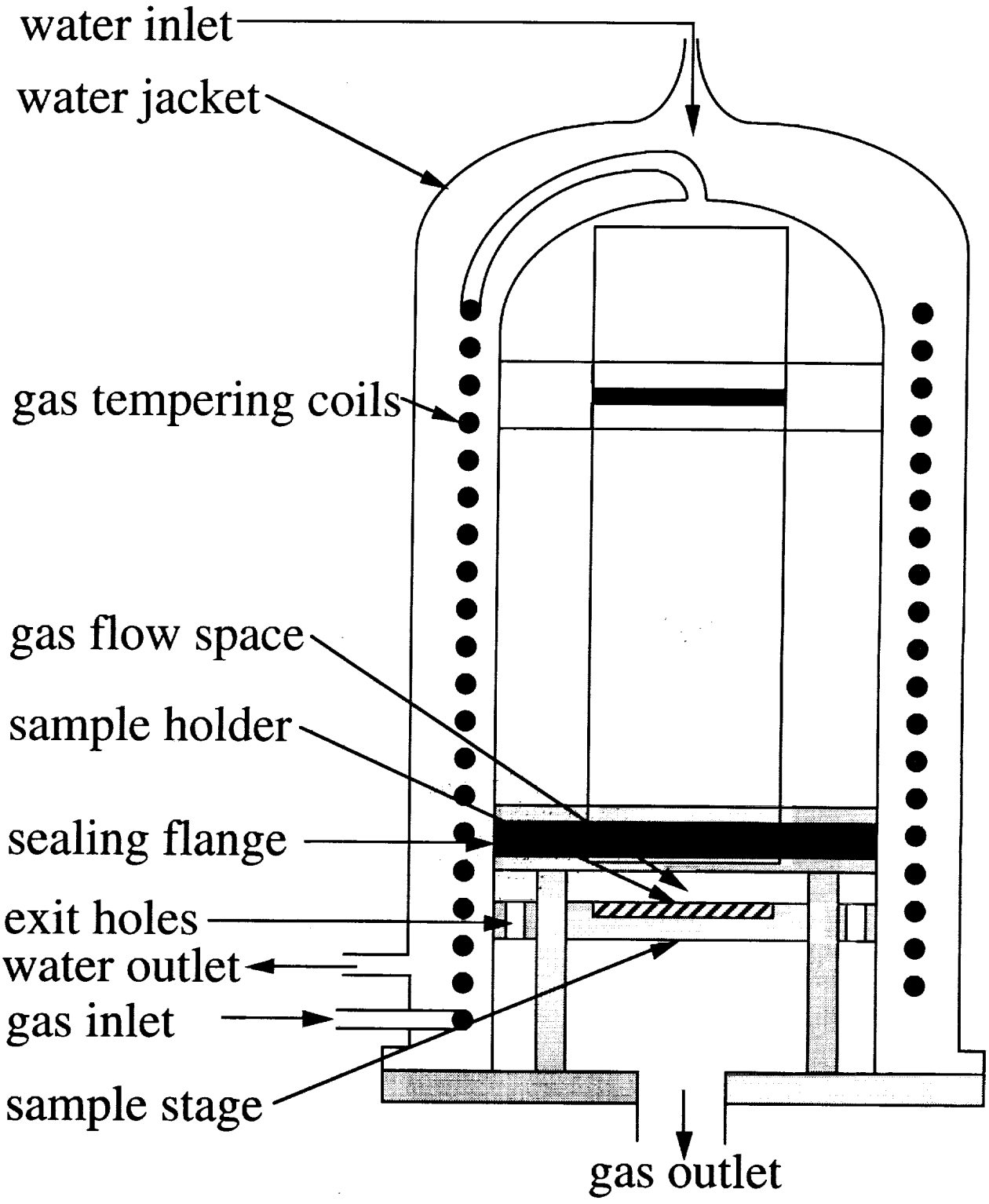


Figure 3. Detailed schematic of the impinging jet chamber

The program for the impinging jet reactor in this study solved the governing equations subject to the boundary conditions using a finite volume method documented by Patanker [9]. The FLUENT software package was used to formulate and solve the appropriate discretization equations for the impinging jet model. Details about the grid and the solution method can be found in Snyder *et al.* [8]. The physical constants and specifications used in the simulation were the following:

Molecular diffusivity of the pollutant:	1.5E-5 m^2/s
Volumetric flow rate of humid polluted air:	2.67E-5 m^3/s
Density:	1.13 kg/m^3
Viscosity:	1.85E-5 kg/ms
Temperature:	303 K

These factors gave a Sc number of 1.09. This volumetric flow rate gave Re numbers of 82, 41, and 27 for the 1 inch, 2 inch and 3 inch tube diameters, respectively.

Results of the Numerical Modeling

The local mass transfer coefficient to the samples below the nozzle, in dimensionless form, appears in Figures 4, 5, and 6 as a function of radial position for nozzle diameters of 1 inch, 2 inches, and 3 inches, respectively. The ordinate, expressed in physical properties, is

$$\frac{Sh}{Re^{1/2}} \equiv \frac{k}{D}\left(\frac{\rho R}{2\mu < v >}\right)^{1/2} \qquad (1)$$

where

$$Sh = \frac{kR}{D}$$

$$Re = \frac{2R < v > \mu}{\rho}$$

$$Sc = \frac{\mu}{\rho D}$$

and k is the mass transfer coefficient; R is the nozzle radius; D is the binary diffusivity of oxidant in the air; <v> is the average velocity in the nozzle; μ is the viscosity; and ρ is the density of the test gas. Thus the ordinate is proportional to the local mass transfer coefficient, k, and is therefore proportional to the ability of the mixed flowing gas to deliver the pollutant to a particular location.

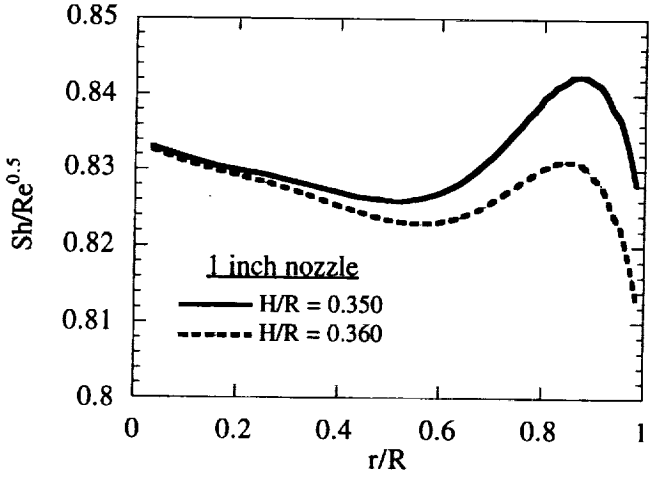


Figure 4. The dimensionless local mass transfer coefficient on the surface of impingement as a function of radius from the axis of the 1 inch nozzle. In the case of H/R = 0.35, the maximum nonuniformity is +/- 0.0085 or about 1% of the average value.

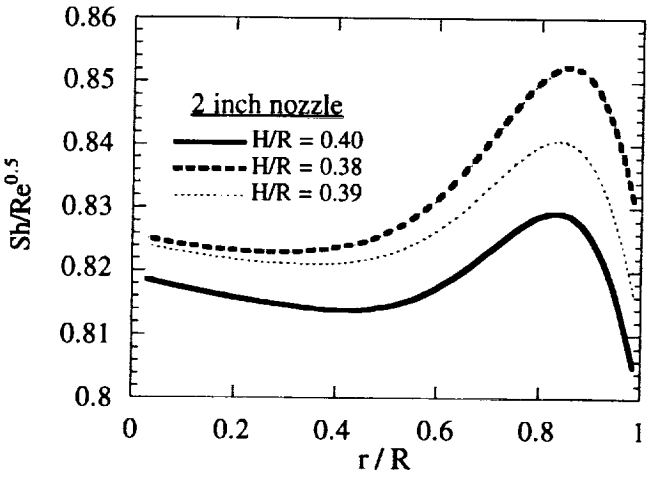


Figure 5. The dimensionless local mass transfer coefficient on the surface of impingement as a function of radius from the axis of the 2 inch nozzle. In the case of H/R = 0.39, the maximum nonuniformity is +/- 0.012 or about 1.5% of the average value.

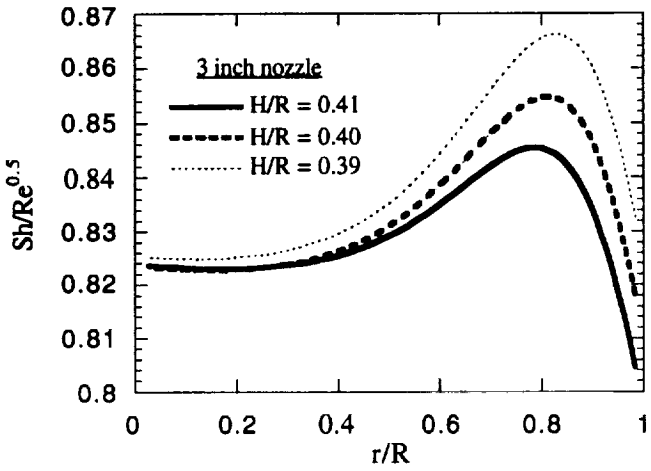


Figure 6. The dimensionless local mass transfer coefficient on the surface of impingement as a function of radius from the axis of the 3 inch nozzle. In the case of H/R = 0.40, the maximum nonuniformity is +/- 0.018 or about 2% of the average value.

Note the resolution of the ordinates in Figures 4, 5, and 6. The maximum variation of the mass transfer coefficient at the surface of impingement over the radius of the nozzle is +/- 1%, 1.5% and 2% , respectively, for the 1 inch, 2 inch, and 3 inch diameter nozzles. The implication is that the delivery of pollutants to the surface of impingement is uniform to within the quoted percentages under the assumption that all of the active species in the gas phase that reach the surface are immediately and irreversibly consumed. The optimum results for the three different nozzle sizes, plotted on a scale that reveals the essential uniformity of the mass transfer coefficient over the radius of the nozzle, appear in Figure 7.

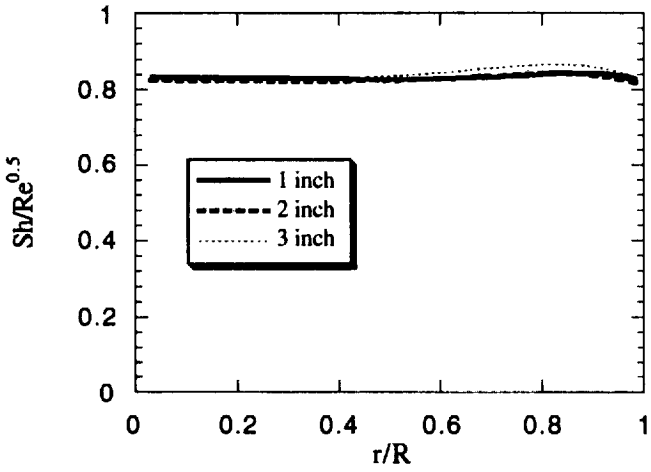


Figure 7. The uniformity of mass transfer to samples. Heights of the nozzles above the sample stage: 1 inch: 0.177"; 2 inch: 0.390"; 3 inch: 0.600". The heights should be within +/- 0.010" of the indicated values.

The domain and boundaries appearing in Figure 2 shows that the entire surface of impingement below the nozzle is active from the axis to the radius of the chamber. In practical situations, one places coupons only in the area of interest below the nozzle, but this can lead to spurious effects for the samples at the outer edge because the pollutants can diffuse upstream and increase the exposure of the outermost samples. This situation is depicted in Fig. 8. The dashed line corresponds to the case where the outer boundary of the sample region ends at the radius; the dark continuous line corresponds to the situation when the entire surface is active or when a set of buffer samples is included outside the radius of the nozzle.

Discussion

Given the above results and the work of Scholtz and Trass [7], one can express the dimensionless mass transfer rate as

$$Sh = 0.82\,Re^{0.5}Sc^{0.36} \quad (2)$$

Rewriting equation (2) and multiplying by the concentration of oxidant, one can calculate directly the mass transfer limited corrosion rate in μm per day.

$$\frac{d\delta}{dt} = N\left[0.82\left(\frac{\mu}{\rho}\right)^{-0.14} D^{0.64}\left(\frac{2\langle v\rangle}{R}\right)^{1/2}\left(\frac{Px_o}{R_{gl}T}\right)\right]\left[\frac{M_m}{\rho_m}\right] \quad (3)$$

where

$\frac{d\delta}{dt}$ = corrosion rate
$\langle v\rangle$ = average flow velocity in the nozzle
R = radius of the nozzle
P = atmospheric pressure, 1 atm
x_o = mole fraction of oxidizer in the system
R_{gl} = gas law constant
T = temperature
N = the number of media atoms corroding per molecule of oxidant
ρ_m = density of magnetic material
M_m = molecular mass of magnetic material

Equation (3) assumes that there is a primary oxidant that is transported by diffusion and convection to the sample surface where it is adsorbed and reacts with unit probability. Note that the density of the magnetic medium must be its effective density. For example, ME films contain voids, formed during the deposition process, that reduce the saturation magnetization relative to the bulk alloy [10].

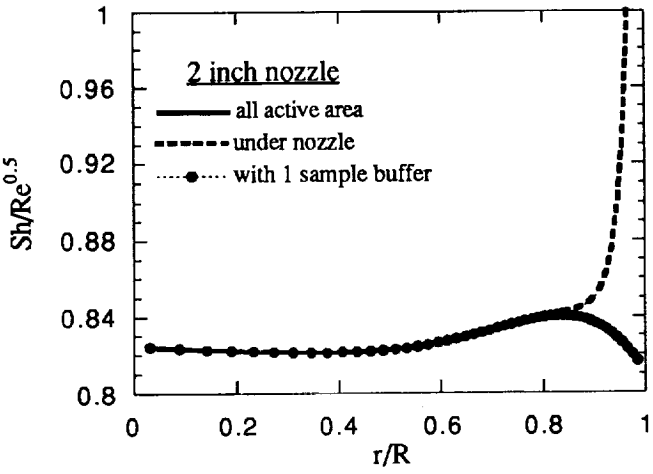


Figure 8. Dimensionless local mass transfer as a function of radial position from the axis on the surface of impingement. When the entire surface outside as well as inside the nozzle radius is active, the dependence is identical to the performance when a buffer layer one coupon wide is included is added to the samples. When no buffer layer is present, the mass transfer rate increases strongly because of the availability of species to diffuse upstream and react.

In general, the result of experiments of this type is not either the mass transfer limited reaction rate or the reaction limited rate, but a mixed rate. The answer one wants from an experiment such as this, for comparison to other media, is not the mixed-mode rate, but the reaction-kinetics-limited rate that is obtained, in principle, when the flow velocity is infinite. One can find this rate by assuming linearity of both the transport and reaction with oxidant concentration, an assumption crucial to accelerated testing in any case, and by expressing the kinetics as a series resistance problem. Analysis of this model reveals the following relationship between the measured corrosion rate, the intrinsic reaction rate, and the radius of the nozzle., *i.e.*

$$\frac{1}{\left(\frac{d\delta}{dt}\right)} = \frac{1}{\left(\frac{d\delta}{dt}\right)_r} + C R^{3/2} \qquad (4)$$

where $\left(\frac{d\delta}{dt}\right)_r$ is the reaction kinetics-limited rate of moment loss and C is a collection of constants related to the mass transfer part of the problem. A plot of the left side of equation (4) versus the 3/2 power of the radius should be a straight line having the reciprocal of the desired quantity as the intercept at the ordinate.

Conclusion: A Proposed Flow Geometry for Direct Exposure Testing

Using the above results and analysis, we propose the following prescription as a means of assuring the reproducibility of direct exposure testing. Construct an isothermal chamber having the capacity to hold a glass nozzle 1 inch, 2 inches, or 3 inches in diameter and 30 inches long. The nozzle should face a sample stage capable of holding samples at right angles to the nozzle. The opening of the nozzle facing the samples should be flush with a plane that extends to a radius of at least 3 inches. Provide a supply of humid polluted air to the chamber at a total volumetric flow rate of 1600 sccm. All the other Battelle Class II specifications remain the same. The nozzles should be suspended above the sample stage by 0.177", 0.390", and 0.600" +/- 0.010" to assure that the delivery of pollutants to the surface does not vary from axis to nozzle radius by more than approximately 2%. The samples should be coplanar with the surface of impingement. We have found that coupons 5 mm square and attached by double stick tape to glass inserts in a plastic chip carrier matrix work very well. Coupons this size generally have sufficient magnetic moment to be measured and a number of them can be exposed at once, particularly if the 3" nozzle is used. The thickness of the glass inserts and tape can be matched to permit the sample coupons to be flush with the impingement surface.

There are two main advantages and one disadvantage of this specification. First, the proposed configuration eliminates ambiguity in the flow over the samples and therefore promotes reproducibility. Second, the laminar flow pattern can be calculated and the mass transfer coefficients deduced to determine the relation between the amount of pollutants delivered and the rate of corrosion of the samples. For example, one draws very different conclusions if the measured rate is much less than-, equal to-, or much greater than the rate of supply of pollutants to the samples. The disadvantage of the proposed approach is that the close proximity of different samples in the sample tray under the nozzle can affect the results if the samples have very different susceptibilities to corrosion. For example, if one sample is very reactive and it adjoins a relatively unreactive sample, the reactive samples will "steal" reactant from the more noble sample and the result would be that the difference between the two samples would be accentuated. If the precise corrosion rate of a particular sample is desired, the best solution is to do preliminary comparison testing of mixed sample types and then do final testing of important sample types separately. This problem could also be alleviated by a higher volumetric flow rate of gas so that the mass transfer boundary layer thickness over the samples was much less than the dimension of the samples.

References

1. A. Djalali, D. Seng, W. Glatfelter, H. Lambropoulos, J. S. Judge, and D. E. Speliotis, *J. Electrochem. Soc.*, **138**, 2504 (1991).
2. W. Abbott, *Br. Corros. J.*, **24**, 153 (1989).
3. P. J. Sides, G. Spratt, and J. P. Kampf, *IEEE Trans. Magn.* (1994).
4. Kampf, J. P., P. Sides, G. Spratt, J. Electrochem. Soc. (submitted, 1994).
5. Schlichting, H. *Boundary Layer Theory.* 1955 Pergamon Press. New York, NY.
6. Chin, D.-T. and C.-H. Tsang *J. Electrochem. Soc.* **125**: 1461, 1978.
7. Scholtz, M. T. and O. Trass, *AIChE J.* **16**: 82, 1970.
8. D. Snyder, P. Sides, E. Ko, *J. Crystal Growth* **123** 163 (1992).
9. Patanker, S. V. *Numerical Heat Transfer and Fluid Flow* 1980 Hemisphere Publishing Corporation. New York.
10. S. L. Zeder, J.-F. Silvain, M. E. Re, M. H. Kryder, and C. L. Bauer, *J. Appl. Phys.*, **61**, 3804 (1987).

Optical Storage Media Data Integrity Studies

N95-24130

Fernando L. Podio
National Institute of Standards and Technology
Building 225, Room A61
Gaithersburg, MD 20899
fernando@pegasus.ncsl.nist.gov
301-975-2947
301-216-1369 (fax)

43466
p. 12

Abstract

Optical disk-based information systems are being used in private industry and many Federal Government agencies for on-line and long-term storage of large quantities of data. The storage devices that are part of these systems are designed with powerful, but not unlimited, media error correction capabilities. The integrity of data stored on optical disks does not only depend on the life expectancy specification for the medium. Different factors, including handling and storage conditions, may result in an increase of medium errors in size and frequency. Monitoring the potential data degradation is crucial, especially for long term applications. Efforts are being made by the Association for Information and Image Management Technical Committee C21, Storage Devices and Applications, to specify methods for monitoring and reporting to the user medium errors detected by the storage device while writing, reading or verifying the data stored in that medium. The Computer Systems laboratory (CSL) of the National Institute of Standards and Technology (NIST) has a leadership role in the development of these standard techniques. In addition, CSL is researching other data integrity issues, including the investigation of error-resilient compression algorithms. NIST has conducted care and handling experiments on optical disk media with the objective of identifying possible causes of degradation. NIST work in data integrity and related standards activities is described.

Introduction

Many organizations are using optical disk-based systems for the storage and retrieval of large sets of valuable information. One general indicator for long term storage of data is the optical disk media life expectancy. For this indicator to be of value, a standard method to determine life expectancy is essential. Extrapolated life expectancy values may vary greatly because they depend on the test method used for calculating the quality parameter (e.g. the byte error rate), the measurement approach (including areas on the disk tested, data patterns written, and amount of data tested), the mathematical model used, and the criteria for data analysis (including the statistical analysis used and the confidence levels), Podio [1].

If a standardized test were employed by all media manufacturers, media life expectancy could

be a good parameter to select media for long term applications. However, to determine a life expectancy specification, tests are run with a small sample of disks from a population of manufactured disks. In addition, media technological changes would require running life expectancy tests almost continuously to test newcomers on the market, since the old data obtained on previous life expectancy tests may not apply to new technology. In conclusion, a life expectancy specification is useful only as a general indicator for media selection. Individual disks will still fail at different times.

All storage devices are designed with powerful, but not unlimited, error correction capabilities. Because of different factors which include handling and storage conditions, errors may increase in size and frequency. If the level of errors increases beyond the maximum capacity of the ECC (error correcting codes) in the device, data will be uncorrectable. By not being informed of the level of error correction that is taking place in the optical disk device, users learn about critical error events only when the data is already irretrievable. If these types of critical errors occur on a specific unlinked data structure, this data structure may no longer be recoverable but data degradation may not have caused extensive damage outside the unlinked data structure. However, if data degradation at these critical levels of unrecoverable errors occurs in any linked structure or with a compressed data entity, substantial data losses may result.

Several approaches can be followed for improving data integrity. One approach is to monitor data errors with time. Users can gather information to highlight trends in particular selected disks or their entire data sets. This monitoring capability allows users to make decisions on transferring data to new media in a timely and economic manner before data loss occurs. Another method to increase data integrity is to use layered ECC. Although layered error correction decreases the user data capacity, it adds error resilience.

Compressed data in the presence of errors is especially vulnerable to catastrophic data failure. Woolley [2] emphasizes the importance of robust error control in data compression applications. For compressed data, in addition to using media error monitoring and layered ECC, there are other techniques to improve data integrity in the presence of errors: error correction integrated with data compression Kobler [3], entity reduction and error-resilient compression. NIST is currently investigating the error-resilience of these techniques.

Efforts to develop standard media error monitoring tools and techniques for optical disk drives and NIST's involvement in the development of this standard are described. NIST investigations on media error monitoring tools, data analysis statistical models for error distribution, and media error visualization are also described.

Another aspect of data integrity research at NIST includes an experimental program for the care and handling of optical disks. A series of experiments were performed using different types of optical disks. The experiments included exposure to liquids and vapors, cleaning agents, solvents, fire smoke, food substitutes, paint fumes and paint, temperature and humidity cycles, heat and cold shocks, uniform pressure, static electricity, gamma rays, etc.

A brief description of this work is also included.

Standards for Media Error Monitoring and Reporting Techniques

In 1991, the Computer Systems Laboratory of NIST sponsored a workshop to identify the state of the art on media error monitoring approaches for optical disks and to identify the user's needs, Podio [4]. As a result of the workshop, a working group was formed to identify media error monitoring techniques. The working group developed a set of procedures for monitoring and reporting media error correction levels on optical disk devices. The results of this activity are being used as a basis for formal standard work.

With NIST leadership, the Association for Information and Image Management (AIIM) Committee C21 (Storage Devices and Applications) is developing the American National Standard ANSI/AIIM MS59. ANSI/AIIM MS59 specifies media error monitoring and reporting techniques for the verification of information stored on optical digital data disks[1], AIIM C21 [5].

Parallel efforts are taking place in developing an international standard under the auspices of the International Standards Organization (ISO) Technical Committee TC 171, Micrographics and Optical Memories for Document and Image Recording, Storage and Use. The current content of the proposed ISO standard is based on ANSI/AIIM MS59.

ANSI/AIIM MS59 provides a toolkit of media error monitoring and reporting techniques, any combination of which may be employed. The standard provides two levels of media error monitoring and reporting techniques, a functional approach and an implementation of a selected set of Small Computer Interface (SCSI-2) commands.

The high level approach (a set of functional commands) is independent of the host operating system (e.g. DOS, Unix, OS/2, etc) and the interface that connects the optical disk device with the host (e.g. SCSI-2, IPI, LAN, etc). This high level interface approach is media type and size independent. That is, it can be used with systems that use WORM (write-once read many), rewritable or partially read-only media and optical disk drives for different media sizes from 90 mm to 356 mm media. The implementation of a selected set of SCSI-2 commands enables media error monitoring and reporting techniques at the device level providing direct communication with an optical disk drive that uses the SCSI-2 interface.

1

The U. S. National Archives and Records Administration (NARA) has recently published a Technical Information paper NARA [6]. NARA's publication provides recommendations on long-term access strategies for Federal Agencies using digital-imaging and optical digital disk storage systems. One of NARA's recommendations on data integrity states that users should "require that equipment conform to the proposed national standard ANSI/AIIM MS59".

The media error information that can be obtained using the tools specified in the standard include:

- A list of reallocated sectors.
- Corrections that exceed some specified media error levels.
- Warning on specified verify media error levels.
- Total number of bytes in error, number of bytes in error per sector and maximum number of bytes in error in any sector codeword.
- The uncorrected or corrected sector content.
- Errors encountered reading header information such as the sector address, sector marks, and synchronization signals.
- The maximum length of contiguous defective bytes.

From the user's perspective, the purpose of ANSI/AIIM MS59 is to allow users of the standard:

- To have a better understanding of the status of their data stored on optical disks.
- To obtain media error information as directed by the system administrator.
- To enable data recovery with tools of the desired level of sophistication.
- To provide media error information allowing the user to make decisions about the media at the present time, and also provide error information which will highlight trends in particular selected disks or in their entire data sets.
- To make decisions about how long the media can be used without an unacceptable risk of data loss.
- To develop more cost effective backup, recopying and data transfer policies.

The user or implementor of ANSI/AIIM MS 59 will be able to:

- Format the optical digital data disks with or without certification.
- Reallocate sectors when specified media error levels are exceeded.
- Obtain information about all the reallocated sectors and/or a defect list of initial media defects.
- Set media error level values to obtain early warning information about the status of the data and/or interrogate the drive to obtain the values of those set media error levels.

The **media error levels** are what the optical disk drive will use for error recovery. If the ECC level of correction exceeds one or more of the set levels and reallocation is enabled, the sector that exceeded the media error level(s) is reallocated to a spare sector. Whether reallocation is enabled or not, the optical disk drive reports to the host that a set level was exceeded, indicating which one was exceeded, and whether or not the data was recovered.

The following are the media error levels specified in ANSI/AIIM MS59:

- Number of bytes in error per codeword
- Number of bytes in error per sector
- Number of bad sector IDs
- Number of missing resync bytes (when the drive uses resync bytes)

AIIM C21 is also developing an accompanying ANSI Technical Report, AIIM C21 [7] describing guidelines for the use of the media error monitoring techniques documented in ANSI/AIIM MS59. The current outline for the guidelines includes:

- A description and use of the media error monitoring and reporting techniques documented in ANSI/AIIM MS59.
- Discussion of error management strategies.
- Methods of visualization of media error information using the techniques specified in MS59.
- Methods to estimate the data integrity including:
  - use of sampling methods
  - use of baseline media error parameters and distributions
  - use of statistical models

NIST Investigation of Media Error Monitoring Tools for Optical Disks and Media Error Visualization

Concurrently with the standardization efforts, NIST has also been conducting laboratory work in investigating media error monitoring and reporting (MEMR) techniques, statistical models for error distribution, and methods for error data visualization.

The **MEMR techniques** are used in optical disk drives for the verification of information stored on the optical disks. These techniques allow users to obtain timely information about the status of their data. NIST has investigated some of the MEMR techniques available in commercial drives, and has researched possible new implementations. All of this work has contributed to the content of the proposed ANSI/AIIM MS59 standard and the parallel proposed ISO standard.

NIST has also developed guidelines for the use of the MEMR tools. The guidelines include procedures that end users or system integrators can use to monitor the status of data stored on optical disks. These MEMR techniques for optical disk drives may be the basis of similar MEMR techniques for other types of high density/high capacity mass storage technologies, such as magnetic media disk/tape drives, optical tape drives and devices based on new page-oriented memories.

NIST has looked at **statistical models for media error distributions** on optical disks. One model is the modified Gilbert model, Marchant [8]. This model is based on two different classes of defects and has been found to give an excellent fit to defect statistics on media that

uses the magneto-optical recording Takeda, Saito, and Itao [9]. This model takes into account modeling non random errors (long burst defects). The Gilbert model requires two byte error rate (BER) values and two average burst lengths. One BER is derived from microscopic defects, the other from larger media substrate damage. The model output is the burst-length probability distribution.

A simpler statistical model has also been developed at NIST. This model is the basis for predicting the maximum number of errors in a sector codeword. One version of this model assumes a uniformly random binomial distribution of errors, and uses only one byte error rate (BER) and the number of sector interleaves as input. It produces a baseline for comparison at different times on the status of an optical disk every time the disk is tested. The output of this model can provide a basis for comparison with reported disk error statistics so that the user can identify abnormal changes in the media error distribution. Figure 1 shows the model output, a distribution of bytes in error per codeword.

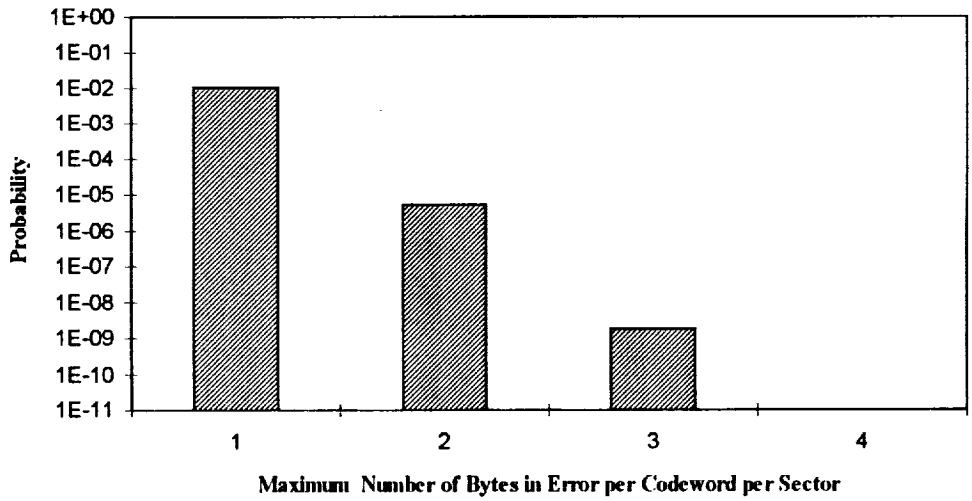


Figure 1. *For this particular disk type, the sector data field, which includes the user data bytes, the ECC bytes, and the CRC bytes, is divided into five codewords. The maximum number of bytes in error of the five codewords per sector is determined, and the probability of occurrence of one, two or more bytes in error is plotted. The maximum number of errors per codeword that can normally be corrected in drives that use this type of media is eight.*

Depending on the different values of the expected or empirical BER, the model can also indicate that the drive's maximum error correcting capabilities are being approached or exceeded. The modified Gilbert model describes only error length distributions, but does not make the connection to error correcting capabilities. This simpler model makes this link. It uses the real disk data structure (sectors and interleaves) and assumes only a simple uniformly random distribution of errors, such as binomial or Poisson.

Given a measured BER, the model can be used to generate a predicted distribution of the

number of bytes in error per codeword. If the predicted distribution is not acceptable (number of bytes in error in any codeword exceeds certain user established number), the user may consider retiring the media at this point. If the predicted distribution is acceptable, the user should determine the real distribution of the number of bytes in error per codeword (using a ANSI/AIIM MS59 compliant drive or any other drive that provides this type of media error information). If the measured distribution shows an excess in the number of bytes in error, the user may consider retiring the disk. Because the model assumes a random distribution of errors, if the predicted and measured distributions are significantly different, a non random error distribution might be supected and the user may use other MEMR tools to investigate the level and the distribution of these errors further.

NIST has also developed **media error visualization tools** for the byte-error statistics retrieved via the media error reporting tools documented in the ANSI/AIIM MS59 standard. The media error visualization tools include:

a. Line graphs depicting:

- Sector reallocations over time.
- Bad sector ID's over time.
- Byte error rate over time.
- Maximum bytes in error per codeword and per sector over time for a given disk as shown in Figure 2.

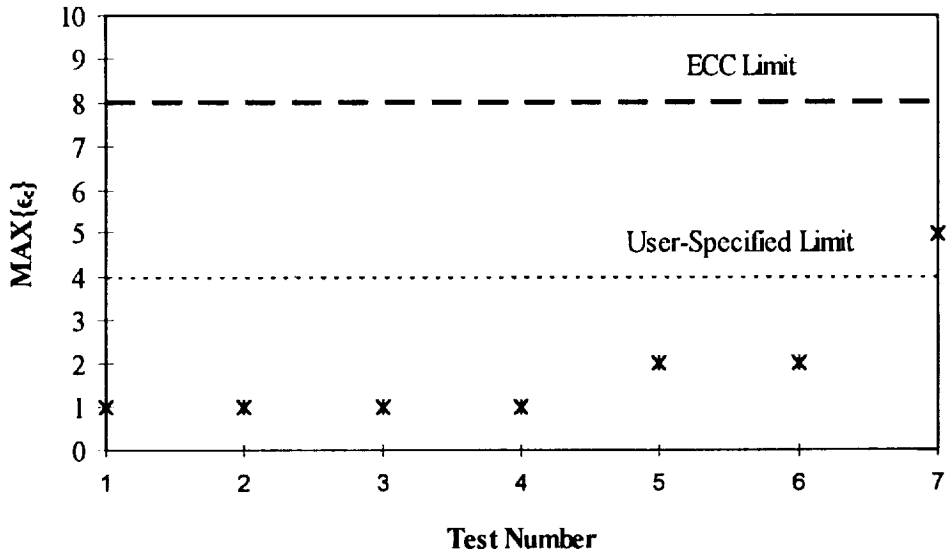


Figure 2. *This line graph shows how the maximum, or worst-case, number of bytes in error per codeword may change over time for a given disk. The normal ECC correction limit capability is eight bytes per codeword.*

In Figure 2, the user-specified error level is set at a maximum of four bytes in error per codeword.

Using ANSI/AIIM MS59 compliant drives the user can check the default level of bytes in error per codeword that, when exceeded, would enable the reallocation of the sector. Using this type of drive, the user can also change this error level.

b. Bar charts depicting:

- Relative frequency of maximum bytes in error per codeword and per sector.
- Maximum bytes in error per codeword per radial area of a disk.
- Maximum bytes in error per codeword per band of tracks.

The bar chart in Figure 3 shows tha maximum number of bytes in error per codeword per band of tracks for a given disk.

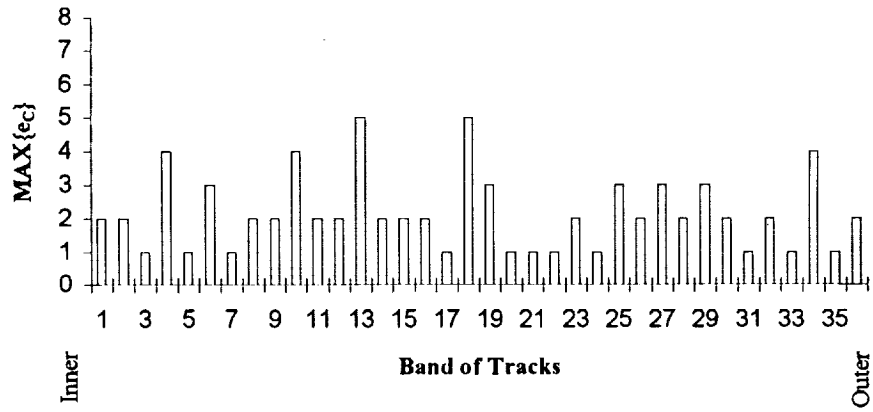


Figure 3. *The maximum number of bytes in error per codeword is shown in this chart in a different way than in Figure 2. The disk has been divided into bands of n tracks and the maximum number of bytes in error per code word in shown for all the sectors within the tracks specified in a particular band. Dividing the disk into bands of n tracks enables visualization of the entire disk from the inner area to the outer area.*

c. Three-dimensional histograms depicting:

- Maximum bytes in error per sector over the disk.
- Maximum bytes in error per codeword over the disk as shown in Figure 4 in a three-dimensional histogram.

The media error information conveyed in this Figure may be, in many cases, sufficient for most users. As in Figure 3, the disk has been divided into bands of n number of tracks (31 bands in this Figure).

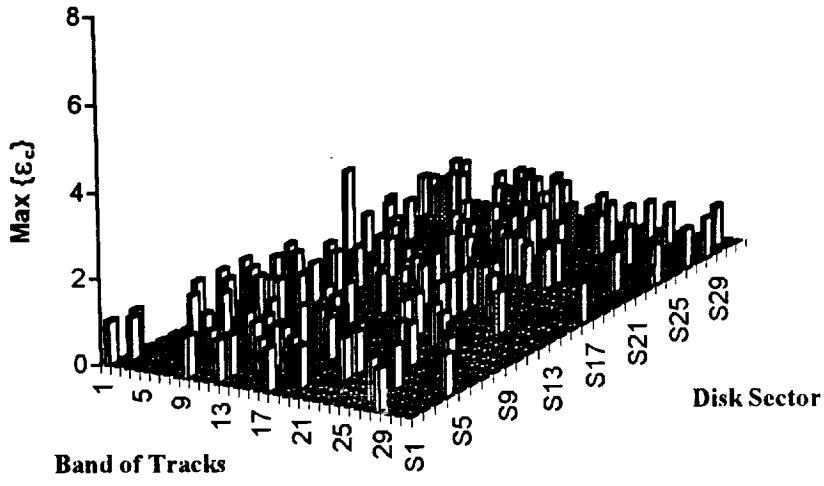


Figure 4. *The maximum number of bytes in error per codeword over the disk is shown in this Figure using a three-dimensional representation. The maximum number of bytes in error per codeword is shown for all the sectors within the tracks specified in a particular band. Only one band shows sectors that have codewords with more than one byte in error. The ECC correction capability for drives compatible with this type of disk is eight bytes in error per codeword. The Figure shows a fairly healthy disk.*

By using ANSI/AIIM MS59 compliant drives or other drives that provide similar media error information, the user can typically obtain from the drive information on media errors with the desired level of detail and sophistication.

When the user wants to analyze media errors in specific disks, media error visualization charts can be used. However, when users want to apply media error monitoring tools to a large number of disks, plotting error distributions or other statistics might be impractical. In this case, the user should access the required information numerically and take the appropriate action. For example, if the users set the media error levels, they may decide to transfer data to another disk when these levels are exceeded. More information about these procedures or how to use the visualization tools is provided in AIIM C21 [5] and [7].

Care and Handling Experiments for Optical Digital Data Disks

NIST has conducted care and handling experiments with the objective of identifying possible causes of data degradation in optical disk media. In order to conduct these experiments NIST developed optical disk media measurement systems capable of determining data degradation parameters such as the byte error rate, and error distributions. Figure 4 was based on one of these error distributions. A measurement system for mechanical characteristics of optical disk media was developed by Nevenzel and Voogel [10].

Optical disks may deteriorate if subjected to some unusual conditions such as extreme temperature and humidity, temperature and humidity cycles, and high energy radiation. Some office cleaning substances and other components like tobacco smoke, liquids and food may also produce data degradation. Such degradation effects were investigated through care and handling experiments. The approach that was followed consisted of writing a selected number of sectors of the disks and reading them back checking the bytes in error and the error distribution. The information that is derived is the byte error rate (BER), which gives an average measure of the number of bytes in error, the defect distributions, and the location of burst errors. Some mechanical measurements were also performed. For CD media testing, which included CD-ROMs and CD-R's (CD recordable media), a commercially available tester was used. NIST is currently analyzing the test results.

The experiments included:

- Cleaning agents immersion tests and vapor and gas exposure.
- Fire smoke exposure and exposure to chemicals used in fire extinguishers.
- Exposure to food substitutes.
- Exposure to paint and wax fumes.
- Temperature and humidity cycles; cold and heat shocks.
- Mechanical experiments such as impact and uniform pressure.
- Human interaction such as scratches, permanent inks, hand creams and bending experiments.
- Electromagnetic exposure such as magnetic fields, gamma-rays, X-rays and electrostatic discharge and sun light.
- Exposure to possible harmful liquids such as gasoline and diesel.
- Read, write and erase cycles.

A complete description of the measurement procedures, the experiments conducted and the test results are to be included in a NIST report that is currently being prepared for publication.

Current and Future Plans

NIST will continue to investigate techniques to improve data integrity in the presence of

media errors. Work will continue in the development and analysis of statistical models for error distribution and visualization tools. The work will also include the investigation of data integrity of storage media using layered ECC, and the investigation of emerging techniques for data compression including entity reduction and error-resilient compression. In addition, there is interest from both users and industry in extending the work done on data integrity for optical disks to other emerging storage device technologies.

Conclusions

A life expectancy specification for optical disks (and other types of storage media) is useful only as a general indicator for media selection. But it cannot be the only indicator for assuring data integrity in optical disks. Individual disks may still fail at different periods of time. Work on standardizing media error monitoring and reporting techniques for the verification of information stored on optical disks systems is ongoing. These techniques provide users with a better understanding of the status of the data stored on optical disks. Users can then make decisions about the media at the present time, identify trends and develop more cost-effective backup, recopying and data transfer policies. Without access to media error information, the level of media errors may increase beyond the maximum capacity of the error correcting codes in the device. In this case, data will be uncorrectable.

The level of errors may increase because of factors such as improper handling or storage conditions. Understanding the data integrity of data structures on a disks is important. Compressed data, in the presence of uncorrectable errors, is especially vulnerable to catastrophic data failure. Error-resilient algorithms and other techniques such as layered ECC may increase the chances of data recovery in the presence of uncorrectable media errors. The need to investigate media error monitoring and reporting techniques to verify data integrity on optical disks and other emerging storage media technologies is apparent. Investigation of error-resilient compression techniques is also needed.

Acknowledgements

The work described in this paper is based on the efforts of several people at NIST. I want to acknowledge my colleagues for their input, support and work in this area. In particular, I want to acknowledge the following individuals: Sandra Woolley and Thierry Gouverneur for their work in media error monitoring, data visualization and sampling plans, Sam Silberstein for the development of a statistical model for error distributions, and Stefan Leigh, James Filliben, and other personnel from the Statistical Engineering Division of the Computing and Applied Mathematics Laboratory of NIST for their useful advise and cooperation. Many individuals participated in the care and handling experiments and I want to acknowledge their participation and support in that work. I also want to acknowledge the members of AIIM C21 for their contribution to the ANSI/AIIM MS59 standard and the guidelines. Special thanks are due to Charles Obermeyer from the U.S. National Archives and Records Administration (NARA) and David Patton from IBM, Tucson, AZ. The work described in this paper is

carried out at NIST, under Interagency agreements with several U.S. Federal Government agencies including the U.S. National Archives and Records Administration, the Social Security Administration and the Federal Bureau of Investigation.

References

[1] Fernando L. Podio, "Development of a Testing Methodology to Predict Optical Disk Life Expectancy Values", NIST Special Publication 500-200, December 1991.

[2] Sandra I. Woolley, "The Importance of Robust Error Control in Data Compression Applications", Third NASA Goddard Conference on Mass Storage Systems and Technologies, 1993.

[3] Ben Kobler, "Techniques for Containing Error Propagation in Compression/Decompression Schemes, Space and Earth Science Data Compression Workshop, 1991.

[4] Fernando L. Podio, "Monitoring and Reporting Techniques for Error Rate and Error Distribution in Optical Disk Systems", Results of a Workshop, NIST Special Publication 500-198, August 1991.

[5] AIIM Technical Committee C21, Storage Devices and Applications; Proposed ANSI Standard ANSI/AIIM MS59: "Media Error Monitoring and Reporting techniques for Verification of the Information Stored on Optical Digital Data Disks", Fifth Draft, November 1994.

[6] The National Archives and Records Administration, "Digital-Imaging and Optical Digital Data Disk Storage Systems, Long-Term Access Strategies for Federal Agencies", Technical Information Paper No. 12, July 1994.

[7] AIIM Technical Committee C21, Storage Devices and Applications; Proposed ANSI Technical Report ANSI/AIIM TR39: Guidelines for the Use of Media Error Monitoring and Reporting techniques for Verification of the Information Stored on Optical Digital Data Disks", Second Draft, November 1994.

[8] Alan B. Marchant, "Optical Recording, A Technical Overview", Addison-Wesley Publishing Company, 1990.

[9] Takeda, T., M. Saito, and K.Itao,"System Design of 90 mm Optical Microdisk Subsystem", SPIE Proceddings, 899, 16, 1988.

[10] Gerhard Nevenzel, Martin Voogel; "Mechanical Deformation Measurements for Optical Data Disks", NIST IR 5208, February 1993.

Optimizing Tertiary Storage Organization and Access for Spatio-Temporal Datasets

Ling Tony Chen, Doron Rotem, Arie Shoshani
Mail Stop 50B/3238
Lawrence Berkeley Laboratory
Berkeley, CA 94720
Tel: (510) 486-7160; 486-5830; 486-5171
Fax: (510) 486-4004
Email: LTChen@lbl.gov; D_Rotem@lbl.gov; shoshani@lbl.gov

Bob Drach, Steve Louis
L-264; L-561
Lawrence Livermore National Laboratory
Livermore, CA 94550
Tel: (510)422-6512; 422-1550
Fax: (510)422-7675; 422-0435
Email: drach@cricket.llnl.gov; louisst@nersc.gov

Meridith Keating
Lawrence Livermore National Laboratory
Livermore, CA 94550
Email: mkeating@llnl.gov

Abstract

We address in this paper data management techniques for efficiently retrieving requested subsets of large datasets stored on mass storage devices. This problem represents a major bottleneck that can negate the benefits of fast networks, because the time to access a subset from a large dataset stored on a mass storage system is much greater that the time to transmit that subset over a network. This paper focuses on very large spatial and temporal datasets generated by simulation programs in the area of climate modeling, but the techniques developed can be applied to other applications that deal with large multidimensional datasets. The main requirement we have addressed is the efficient access of subsets of information contained within much larger datasets, for the purpose of analysis and interactive visualization. We have developed data partitioning techniques that partition datasets into "clusters" based on analysis of data access patterns and storage device characteristics. The goal is to minimize the number of clusters read from mass storage systems when subsets are requested. We emphasize in this paper proposed enhancements to current storage server protocols to permit control over physical placement of data on storage devices. We also discuss in some detail the aspects of the interface between the application programs and the mass storage system, as well as a workbench to help scientists to design the best re-organization of a dataset for anticipated access patterns.

1. Introduction

Large-scale scientific simulations, experiments, and observational projects, generate large multidimensional datasets and then store them temporarily or permanently in an archival mass storage system (MSS) until it is required to retrieve them for analysis or visualization. For example, a single dataset (usually a collection of time-history output) from a climate model simulation may produce from one to twenty gigabytes of data. Typically, this dataset is stored on up to one hundred magnetic tapes, cartridges, or optical disks. These kinds of tertiary devices (i.e., one level below magnetic disk), even if robotically controlled, are relatively slow. Taking into account the time it takes to load, search, read, rewind, and unload a large number oof cartridges, it can take many hours to retrieve a subset of interest from a large dataset.

An important aspect of a scientific investigation is to efficiently access the relevant subsets of information contained within much larger datasets for analysis and interactive visualization. Naturally, the data access depends on the method used for the initial storage of this dataset. Because a dataset is typically stored on tertiary storage systems in the order it is produced and not by the order in which it will be retrieved, a large portion of the dataset needs to be read in order to extract the desired subset. This leads to long delays (30 minutes to several hours is common) depending on the size of the dataset, the speed of the device used, and the usage load on the mass storage system.

The main concept we pursue here is that datasets should be organized on tertiary storage reflecting the way they are going to be accessed (i.e. anticipated queries) rather than the way they were generated or collected. We show that in order to have an effective use of the tertiary storage we need to enhance current storage server protocols to permit control over physical placement of data on the storage devices. In addition, these protocols need to be enhanced to support multiple file reads in a single request. We emphasize in this paper the aspects of the storage server interfaces and protocols, as well as simulation and experimental results of the effects of dataset organization for anticipated access patterns.

In order to have a practical and realistic environment, we choose to focus on developing efficient storage and retrieval of climate modeling data generated by the Program for Climate Model Diagnosis and Intercomparison (PCMDI). PCMDI was established at Lawrence Livermore National Laboratory (LLNL) to mount a sustained program of analysis and experimentation with climate models, in cooperation with the international climate modeling community [1]. To date, PCMDI has generated over one terabyte of data, mainly consisting of very large, spatio-temporal, multidimensional data arrays.

A similar situation exists with many scientific application areas. For example, the Earth Observing System (EOS) currently being developed by NASA [2], is expected to produce very large datasets (100s of gigabytes each). The total amount of data that will be generated is expected to reach several petabytes, and thus will reside mostly on tertiary storage devices. Such datasets are usually abstracted into so called "browse sets" that are small enough to be stored on disk (using coarser granularity and/or summarization, such as monthly averages). Users typically explore the browse sets at first, and eventually focus on a subset of the dataset they are interested in. We address here this last step of extracting the desired subsets from datasets that are large enough to be typically stored on tape.

Future hardware technology developments will certainly help the situation. Data transfer rates are likely to increase by as much as an order of magnitude as will tape and cartridge capacities. However, new supercomputers and massively parallel processor technologies will outstrip this capacity by allowing scientists to calculate ever finer resolutions and more time steps, and thus generating much more data. Because most of the data generated by models and experiments will still be required to reside on tertiary devices, and because it will usually be the case that only a subset of that data is of immediate interest, effective management of very large scientific datasets will be an ongoing concern. However, there is an additional benefit to our approach. Even if we accept the premise that users will be

tolerant of long delays (i.e. placing orders that take several hours or overnight to fill), it is still in the best interest of mass storage facilities to be able to process requests more efficiently, by avoiding to read data not needed. This translates into savings on the hardware needed to support an average access load.

It is not realistic to expect commercial database systems to add efficient support for various types of tertiary storage soon. But even if such capabilities existed, we advocate an approach that the mass storage service should be outside the data management system, and that various software systems (including future data management systems) will interface to this service through a standardized protocol. The IEEE is actively pursuing such standard protocols [3] and many commercially available storage system vendors have stated that they will help develop and support this standards effort for a variety of tertiary devices. Another advantage to our approach is that existing software applications, such as analysis and visualization software, can interface directly to the mass storage service. For efficiency reasons, many applications use specialized internal data formats and often prefer to interface to files directly rather than use a data management system.

In section 2, we describe in some detail our approach and the components modules necessary to support it. Section 3 describes the storage system interface design for both the write and the read processes. Section 4 contains simulation and experimental results, and section 5 describes a workbench that was designed to help scientists in selecting the organization of datasets that best suits their anticipated access patterns.

2. Technical Approach

The goal is to read as little data as possible from the MSS in order to satisfy the subset request. For example, for geological fault studies the most likely access pattern is regional (in terms of spatial coordinates) over extended time periods. For this application, the dataset should be partitioned and stored as regional "bins" or "clusters" over time, as opposed to the traditional way of storing data globally for one time slice. In general, the portions of a dataset that satisfy a query may be scattered over different parts of the dataset, or even on multiple volumes. For example, typical climate simulation programs generate multiple files, each for a period of 5 days for all variables of the dataset. Thus, for a query that requests a single variable (say "precipitation") for a specific month at ground level, the relevant parts of the dataset reside on 6 files (each for a 5 day period). These files may be stored on multiple volumes. Further, only a subset of each file is needed since we are only interested in a single variable and only at ground level. If we collected all the parts relevant to a query and put them into a single file, then we would have the ideal cluster for that query. Of course, the problem is one of striking a balance between the requirements of all queries, and designing clusters that will be as close as possible to the ideal cluster of each query. This idea is a common methodology used in data management systems (called "physical database design"). However, such methods have not been applied or investigated much in the context of mass storage systems.

In the past two years we have investigated and developed partitioning algorithms for a specific application: simulation data generated by climate models. In that context, we

have identified specific access patterns which we characterized as query types. In general, the problem is one of finding the best compromise in how to store the data for conflicting access patterns. We have shown that although the general problem is NP-complete, it is possible to develop effective approximate solutions using dynamic programming techniques [4]. We only discussed below the methodology of our approach and the software modules needed to support our approach. The details of the algorithms used are discussed in [4].

2.1 Functional description of the components

We need to address the components necessary for both writing the reorganized dataset and reading the desired subset. The first component, which we call the "data allocation and storage management" is responsible for determining how to reorganize a dataset into multiple "clusters", and for writing the clusters into the mass storage system in the desired order. The parts of the dataset that go into a single cluster may be originally stored in a single file or in multiple files. The second component, which we call "data assembly and access management" is responsible for accessing the clusters that contain relevant data for the requested subset, and for assembling the desired subset from these clusters. One consequence of this component is that analysis and visualization programs are handed the desired subset, and no longer need to perform the extraction of the subset from the file. The details of the two components are shown in Figures 1 and 2.

On the left of Figure 1, the Data Allocation Analyzer is shown. It accepts specifications of access patterns for analysis and visualization programs, and parameters describing the archival storage device characteristics. This module selects an optimal solution for a given dataset and produces an Allocation Directory that describes how the multidimensional dataset should be partitioned and stored.

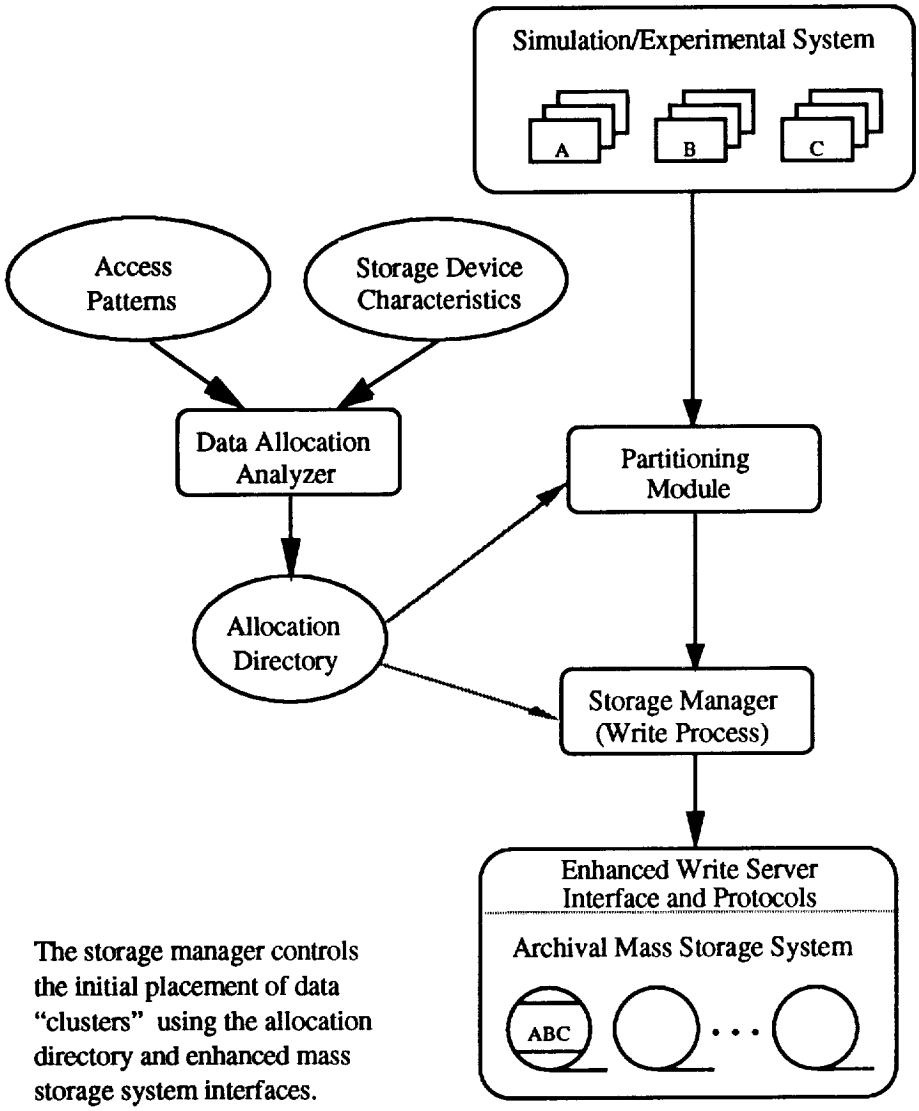


Figure 1: Data allocation and storage management details

The Allocation Directory is used by the File Partitioning Module. This module accepts a multidimensional dataset, and reorganizes it into "clusters" that may be stored in consecutive archival storage allocation spaces by the mass storage system. Each cluster is then stored as a single file, which in most tertiary storage devices today is the basic unit of retrieval (that is, partial file reads are not possible). The resulting clusters are passed on to the Storage Manager Write Process. In order for the Storage Manager Write Process to have control over the physical placement of clusters on the mass storage system, enhancements to the protocol that defines the interface to the archival mass storage system were developed. Unlike most current implementations that do not permit control over the direct physical placement of data on archival storage, the enhanced protocol permits forcing of "clusters" to be placed adjacent to each other so that reading adjacent "clusters" can be handled more efficiently. Accordingly, the software implementing the mass

storage system's bitfile server and storage servers, needs to be enhanced as well. More details on the modified protocols are given Section 3.

In Figure 2, we show the details of reading subsets from the mass storage system.

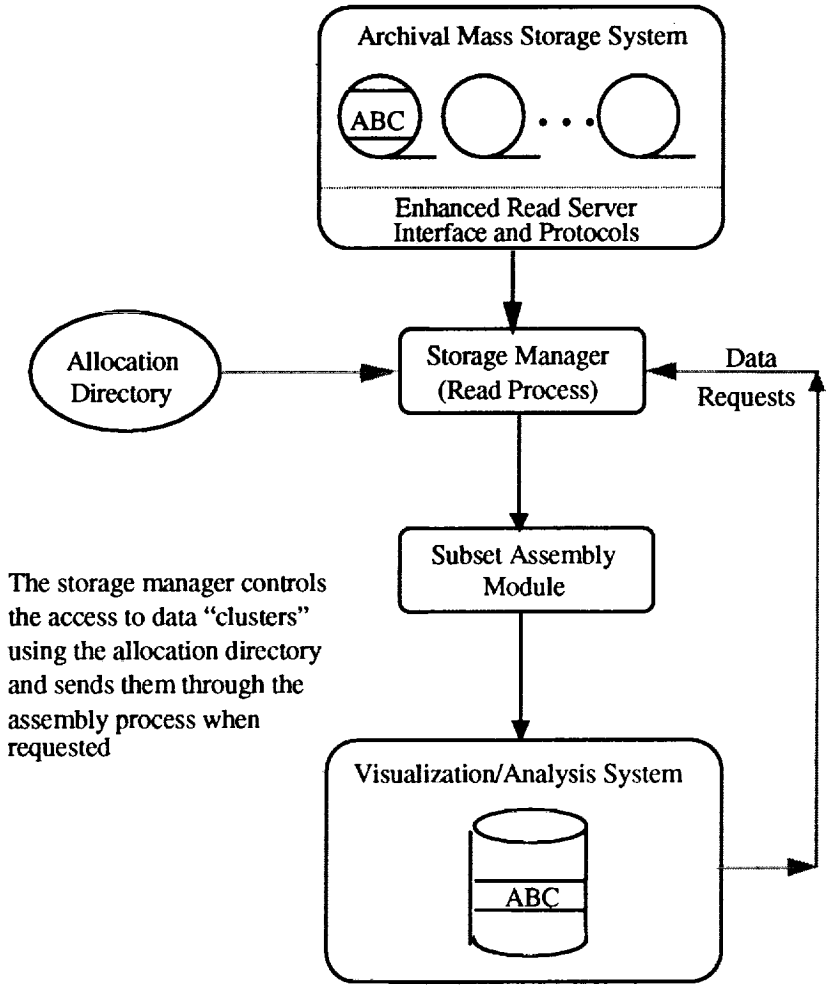


Figure 2: Data assembly and access management details

Upon request for a data subset, the Storage Manager Read Process uses the Allocation Directory to determine the "clusters" that need to be retrieved. Thus, reading of large files for each subset can be avoided. Here again, the bitfile server and storage server of the mass storage system needs to be extended to support enhanced read protocols (see Section 3 for details). Once the clusters are read from the mass storage system, they are passed on to the Subset Assembly Module. Ideally, the requested data subset resides in a single cluster (especially for queries that have been favored by the partitioning algorithm). But, in general, multiple clusters will have to be retrieved to satisfy a subset request, where only part of each cluster may be needed. Still, the total amount of data read will typically be much smaller than the entire dataset. The Subset Assembly Module is responsible for

accepting multiple clusters, selecting the appropriate parts from each, assembling the parts selected into a single multidimensional subset, and passing the result to the analysis and visualization programs.

2.3 Characterization of datasets ,queries, and hardware

The typical dataset in climate modeling applications is not composed of just a single multidimensional file for several variables, but rather a collection of multidimensional files, each for a subset of the variables. The granularity of the spatial and temporal dimensions are common to all variables, but some variables may contain only a subset of these dimensions. For example, a typical dataset may have 192 points on the X dimension, 96 points on the Y dimension, 19 points on the Z dimension (i.e. 19 elevations), and 1488 points on the T (time) dimension covering one year ((12 months) x (31 days/month) x (4 samples/day)). This dataset may contain a "temperature" variable for all X,Y,Z,T and a "precipitation" variable for X,Y,T only. For the "precipitation" variable, the Z dimension is implicitly defined at the ground level. A typical dataset may have close to a hundred of such variables, each using a different subset of all the dimensions. Thus, the characterization of a dataset involves a description of each the above dimensions and for each variable the dimensions that apply to it.

The characterization of queries required extensive interaction with the scientists using the data. After studying the information provided by scientists, we have chosen to characterize "query types", rather than single queries. A query type is a description for a collection of queries that can be described jointly. For example, a typical query type might be "all queries that request all X,Y (spatial) points, for a particular Z (height) one month at a time over some fixed subset of the variables". Thus, assuming that the dataset covers 2 years and 20 height levels, the above query type represents a set of 480 queries (24 months X 20 heights). It was determined that providing query types is more natural for these applications. Further, the query type captures a large number of example queries, and thus permits better analysis of usage patterns.

Each query type is defined as a request for a multidimensional subset of a set of variables, where the multidimensional subset must be the same for all variables of the query type. A query type is defined by selecting one of the following 4 parameters for each dimension:

1) *All*: if the entire dimension is requested by the query type.

2) *One*(coordinate): if exactly one point (the coordinate element) of the dimension is requested.

3) *Any:* if one value along the dimension is requested for this query type. Note that it is assumed that any one of the values within this dimension is equally likely to occur.

4) *Range*(low,high): if a contiguous range that starts at low and ends at high of the dimension is requested.

All variables in our application are defined over a subset of the following seven dimensions: X(longitude), Y(latitude), Z(height), Sample, Day, Month, Year. Note that the Time dimension has been split into 4 dimensions that specify the sample within a day, the day within a month, the month within a year, and the year. The splitting of the time dimension makes it possible to specify "strides" in the time domain, such as "summer months of each year", the "first day of each month", etc. Some variables may not have all dimensions defined. For example, "precipitation" is defined at ground level only, and has no height (Z) dimension.

It has been determined that for our application this query type definition encompasses almost all possible queries that users would want in this application area. It was observed (and verified with climatologists) that the One and Range parameters are not used as often as the All and Any parameters. An example of a query type specification is given below:

Temperature, Pressure: All X, All Y, One(Z,0), All S, All D, Range(M,6-8), Any Y

This query type specifies that temperatures and pressures are requested over all X,Y positions, for Height 0 (ground level), but only for sample points and days in the summer months for a single year. A query belonging to this query type can be specified for any year. Thus, if the dataset is over 20 years, this query type represents 20 possible queries, each being a subset of the multidimensional space.

The characterization of the tertiary storage devices should accommodate various types of devices. We identified the following 5 parameters that are needed to characterize any tertiary storage device for the purpose of determining the optimal partition:

1) M (MegaBytes): the capacity of each tape.

2) R (MB/second): sustained transfer rate, excluding any overhead for starting and stopping.

3) Ts(x) (seconds): fast forward seek function. A mapping function between the distance of the forward seek and the time it takes. For example, if it takes 10 seconds to initialize a seek, and 20MB/s thereafter, the seek function is: $Ts(x) = 10 + (x/20)$. In cases where it is difficult to determine the constant value, the seek function is simply x divided by the seek speed.

4) Tm (seconds): mount time. The time it takes to change a cartridge up to the point where we can read the first byte out of the new cartridge. This time includes: unload previous tape, eject previous tape, robot time to place previous tape back on shelf, robot time to retrieve new tape from shelf, mount new tape, setup tape to be ready to read the first byte.

5) FO (bytes): extra File Overhead. This is the overhead (in bytes) involved in breaking one long file into two shorter files. If retrieving the long file takes T2 seconds, and

retrieving the two consecutive shorter files requires T1 seconds, then the file overhead FO = (T1-T2)*R, where R is the transfer rate defined in point 2 above.

This five parameter model has proven to be sufficient to describe most removable media systems such as robotic tape libraries or optical disk juke boxes. In the latter case, we adjust the seek time component of our cost function to zero as it is negligible compared to the time it takes to dismount and mount a new platter. Measurements of these parameters for two robotic tape systems are given in the next section.

3. The Storage System Interface Design

The developmental and operational site for our work is the National Storage Laboratory (NSL), an industry-led collaborative project [5] housed in the National Energy Research Supercomputer Center (NERSC) at LLNL. The system integrator for the National Storage Laboratory is the IBM Federal Sector Division in Houston. Many aspects of our work complement the goals of the National Storage Laboratory.

The NSL provides two important functions: a site where experiments can be performed with a variety of storage devices, and facilities necessary to support storage and access of partitioned datasets. The first mass storage software developed at the NSL was an enhanced version of UniTree, a system originally written in the 1980s at LLNL and later commercially marketed by DISCOS, OpenVision, and T-mass. The enhanced NSL system, called NSL-UniTree, features network-attached storage, dynamic storage hierarchies, layered access to storage-system services, and new storage-system management capabilities [6]. A commercial version of NSL-UniTree was announced late in 1992 by IBM U.S. Federal. Work on a new storage software system, called the High Performance Storage System (HPSS) [7], is also in progress at the NSL. A central technical goal of the HPSS effort is to move large data files at high speed, using parallel I/O between storage devices and massively parallel computers. HPSS also seeks to increase the efficiency of scientific and commercial data management applications by providing an extensible set of service quality attributes that can be applied to storage resources and devices.

NSL-UniTree manages data using a typical hierarchical file system approach compatible with widely used operating systems such as UNIX. Access to the file system is provided via standard FTP and NFS file transfer protocols, or via a file-oriented client application programming interface (API). The initial interfaces developed for HPSS also support FTP, NFS, and a Client API. However, large scientific datasets are more efficiently viewed as a collection of related objects, rather than as a single large file or multiple independent files. To provide better access to such datasets, we have developed a specification for a more suitable interface between mass storage systems and application software to provide better control over data storage organization and placement for data management clients, such as the data partitioner and subset assembler discussed here. Though modifications to existing interfaces will also be required for HPSS, these will be much smaller in scope because of the system's better ability to classify data by service quality.

3.1 The "write process"

Enhancements necessary to support this work were designed for an experimental version of NSL-UniTree. For the Storage Manager Write Module to provide new attributes related to physical placement of clusters on the mass storage system, modifications to the NSL-UniTree file transfer protocols were developed. Unlike most current implementations that do not permit control over the direct physical placement of data on archival storage, the modified protocol provides a space allocation scheme for storing related data "clusters" and the piece-wise writing and reading of these "clusters".

We designed a functional interface between the data partitioning engine and the mass storage system that provides the ability to control allocation of space and physical placement of data. The approach taken is to define several "Class of Service" (COS) attributes associated with clusters and cluster sets and provide them to the storage system via a modified FTP interface. These cluster-related COS attributes consist of:

1) a cluster set ID.
2) a cluster sequence number.
3) a frequency of use parameter.
4) a boundary break efficiency.

The cluster sequence number identifies the linear relationship between the clusters, and in effect tells the storage system the desired order for storing the clusters. The frequency of use parameter indicates the desirability of storing a cluster close to the beginning of a tape (or to a suitable dismount area) to avoid seek overhead. The boundary break efficiency is a measure of how desirable it is that a cluster stays adjacent to its predecessor. This attribute is used to determine whether separation of two clusters across tape volumes should be avoided if possible.

The COS attributes for a set of clusters are provided to the storage system prior to delivery of the individual data clusters. In reality, they are treated as the initial cluster in a set. This permits the storage system to assign the the rest of the clusters to tape volumes for the desired tertiary storage device. We call an ordered collection of clusters assigned to a single physical volume a "bundle". Thus, the last step of the partitioning process, (i.e. the bundling of clusters such that a bundle can fit on one physical volume), is done by the storage system. This was considered necessary for situations where precise storage system parameters may not be known to the partitioning engine. In addition, this provides the storage system management with an ability to override the partitioning engine's decisions in order to prevent storage system overload or wasted space on physical volumes.

The interface design for NSL-UniTree is shown in Figure 3. As can be seen, this mechanism allows the partitioning engine to determine what it perceives to be optimal data layout for a given device destination. However, it gives final control to the storage system. Data is transferred to the storage system in cluster sets with the COS attributes sent as the first cluster. There is no strict requirement that all clusters be contained within

a cluster set, but a containment relationship is necessary if the benefits of data association are to be realized. Clusters that are provided to the storage system independently will be stored individually (i.e., as normal files).

So as not to constrain the partitioning engine unnecessarily, no limits on cluster set size have been established. This, however, means that a cluster set might be larger than the available storage system disk cache. To prevent cache overflow, the clusters are organized into sets whose sizes are manageable by the storage system.

Once the cluster set attributes are available in the storage system, the bundling function is called to assign clusters into bundles for internal use by the storage system's migration process. The migration server builds a bundle table and iteratively examines that table to confirm existence of complete bundles. To provide the necessary inter-cluster cohesion, bundles are migrated to tertiary storage levels only when complete. Note that bundles may not necessarily represent an entire cluster set. This depends on the storage characteristics of the targeted tertiary volume.

To ensure clusters are stored without unnecessary volume breaks (which would result in an additional media mount penalty for the reading process), the migration server checks destination volume space availability prior to actual bundle migration. If there is insufficient space to store the entire bundle on the destination volume, the bundle migration will be deferred for a finite period that can be set by storage system management policy. The idea behind this deferral is to allow non-bundled files to be migrated first in the hope that this results in the mounting of a new (i.e., empty) volume. Storage system management can override deferral in situations where the system's cache space is in danger of exhaustion. In some storage systems, a full disk cache is a fatal condition.

3.2 The "read process"

We wish to support an object view for application programs where the objects are multidimensional datasets and requests can be made for subsets of these datasets for a single variable at a time. The function of the "subset assembler" is to take such a request from the application, figure out what clusters to request from the mass storage system (if more than one is needed to satisfy the request), and assemble the relevant parts of each cluster into a single multidimensional file to be returned to the application. Consequently, the application programs do not need tp deal with the data assembling details.

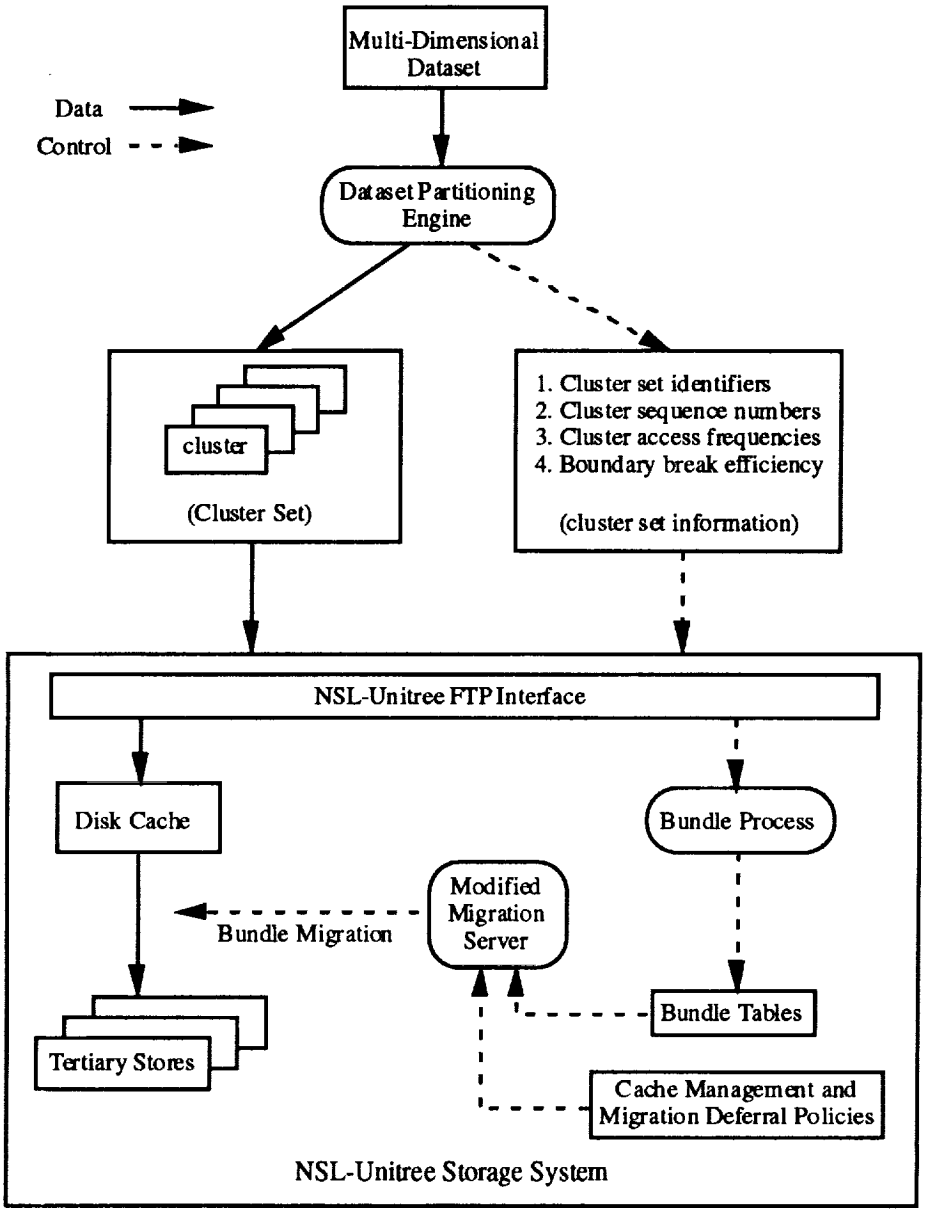


Figure 3: NSL-UniTree enhancement implementation

One of the more difficult tasks that must be accomplished to achieve the subset assembly is the selection of relevant parts from each cluster and the transposition of the dimensions for those parts into the order desired by the application. We decided to take advantage of existing software to perform this function. Currently, the climate modeling files are stored in DRS format [9]. Each file is linearized on the dimensions, and there is a description file associated with each data file. A DRS library exists that performs selection of a part of a multidimensional file, transposing it as desired. The DRS library tries to allocate adequate memory to perform this function in memory. However, when the file is too large to fit in memory, a buffer management algorithm is used to optimize the use of available buffers for files residing on disk. One of the advantages of dealing with clusters, which are

relatively small files, is that it is more likely that enough memory can be allocated for these files, and thus the selection and transposition operations can run efficiently.

The Storage Manager Read Module supports efficient reading of clusters. The desired interface to the mass storage system for the read process is one which supports a single request for multiple files corresponding to the set of clusters that the subset assembler needs. When only a single cluster is needed to satisfy the subset request, the read module needs to mount the proper volume, position to the cluster and read only that cluster. When multiple clusters are needed to satisfy a subset request, the read module needs to ensure reading of clusters in such a way that no unnecessary rewind of the volume take place. Thus, the storage system management should treat this request as a request for an unordered set of files; that is, disregard the specific order that the files were mentioned in the request. It should read the files in the order that is internally most efficient, depending on what volumes are mounted at the time of the request and the order of files on the volume. Abilities to support this type of optimization are supported in most modern storage systems, including NSL-UniTree. This capability was recently tested at the NSL with an Ampex DST robotic tape system. We make use of the DRS library and the NSL-UniTree storage management systems as shown in Figure 4.

As can be seen from Figure 4, the subset assembler accepts a request from the application program using a specially designed API language. This request typically consists of the name of a dataset, a variable, and range specifications for each of the dimensions that the variable is defined on. The subset assembler consults the Allocation Directory, and identifies which clusters are needed to satisfy the request. It then issues a request to NSL-Unitree to stage the files representing these clusters into the cache.

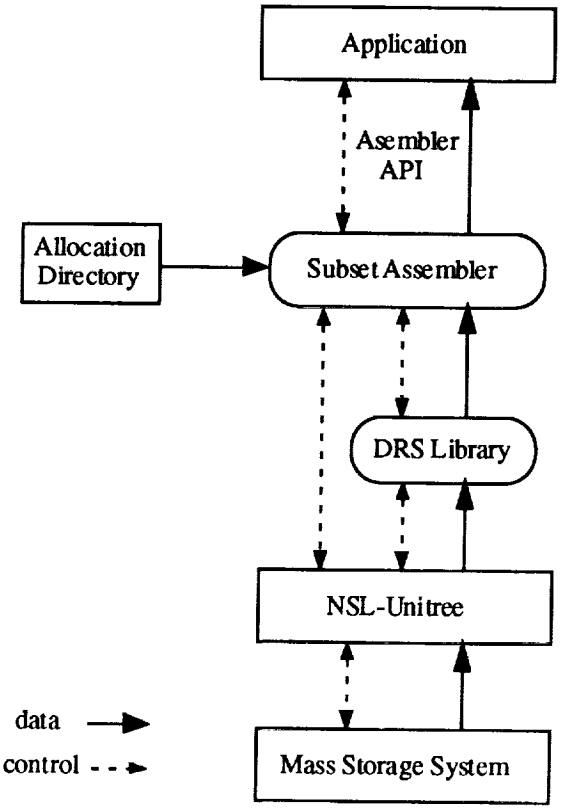


Figure 4: Subset assembler and its interface to NSL-Unitree and DRS library

At this point the assembler does not know in which order files (clusters) will be staged. NSL-Unitree supports a "check status" function where the requesting program can check if a single file was already staged to disk. Therefore, the assembler needs to issue a series of checks to see which of its requested files has been staged. Once a positive response is received for a specific file, the assembler issues a request to the DRS library to read the relevant portion of the file and to transpose the data into the desired order. The information for composing the DRS request is derived from the original application request as well as information from the allocation directory on how each cluster is structured.

The subset assembler places the data which are returned as a result of the DRS request into a pre-allocated buffer space that is large enough to hold the entire subset requested by the application. It then repeats the above process, one cluster at a time, until all clusters have been read. For most visualization applications this subset is small enough to fit in memory. Otherwise, it will be stored on the user's workstation disk. There is the potential of performing these reads in parallel, but we have not concentrated on parallel disk reads, since the major bottleneck for the application is reading files off robotic tape storage.

We plan to partition a commonly used dataset and use this subset assembly process for visualization applications in the summer of 1995. We have performed several experiments and simulations to verify the expected benefits of partitioning a dataset into clusters. These were performed directly on the device, and do not involve the read process described above. We expect the read process to add negligible overhead to the measurements made, since the main component of the response time is reading data from tertiary storage. This needs to be verified through direct use of the subset assembler. The results of the experiments and simulations that were performed so far are given in Section 4.

3.3 HPSS interface implications

We plan to extend our work to the NSL's HPSS storage system as well. There are significant differences between NSL-UniTree and HPSS that may improve the effectiveness of the current mass storage system interface design.

HPSS already implements a COS capability [8] that defines a set of performance and service quality attributes assigned to files and affects their underlying storage resources. COS definitions are designed to help provide appropriate service levels as requested (or demanded) by HPSS clients. In the current HPSS implementation, COS is a data structure associated with a "bitfile", a logical abstraction of a file as managed by an HPSS Bitfile Server. The storage resources used to store the bitfile's data are provided by HPSS Storage Server objects. These Storage Server objects (virtual volumes, storage maps, and storage segments) are associated with a COS-related data structure called a "storage class" that identifies device-dependent characteristics of the type of storage provided by the objects.

Using an HPSS Client API library that currently mirrors POSIX.1 specifications, applications such as the read and write processes described here can specify an existing COS identifier for a file, or fill in COS "hint" and "priority" structures to describe desired (or required) service quality attributes for the file. User-specified priorities which may be NONE, LOW, DESIRABLE, HIGHLY_DESIRABLE, or REQUIRED, affect how the hints will be treated by the Bitfile Server. Using these capabilities, it should be a straightforward process to provide to HPSS with the COS attributes generated by the dataset partitioning engine, and to have these attributes interpreted properly by the servers.

Creating a new file through the Client API is currently performed through an "hpss_Open" call whose input parameters can include pointers to the hint and priority structures. The Bitfile Server will be required to locate an appropriate type of storage if these structures are provided by an application. On the other hand, if null pointers are passed, the Bitfile Server will be free to use a default COS definition for the new bitfile. The hpss_Open call returns a pointer to the COS definition actually used by the Bitfile Server so that applications may monitor the level of service they receive.

The Storage Server's storage segment object is the conventional method for obtaining and accessing "internal" storage resources. Clients of the Storage Server (this would normally

be the Bitfile Server, but could be a data management application authorized to use Storage Server APIs) are presented with storage segments of specific storage class, with address spaces from 0 to N-1 (where N is the byte length of the segment). The Bitfile Server, or other client, provides a storage class identifier and an allocation length during creation of new storage segments. During the Storage Server's allocation of new physical space, only storage maps that have proper storage class are searched. To ensure locating free space of the appropriate type, the storage class should represent a service conforming to the client-specified COS hints and priorities passed to the Bitfile Server when creating new files.

The COS capability in HPSS was designed to be extensible, and additional attributes can be implemented to exert greater influence over server actions for data cluster placement and migration operations. A stated goal for future releases of HPSS is better integration with large data management systems and COS attributes for controlling placement or collocation of related data on physical media in HPSS. This will allow HPSS to be easily used by the read and write processes described here.

Unlike NSL-UniTree, the separation of the Bitfile Server, Storage Server, and Migration/Purge Server in HPSS is clearly defined. Modular APIs exist at each server interface, providing different implementation choices using HPSS. One approach is to expand the Bitfile Server's present COS definition to match the requirements of the data partitioning engine and assembly process. The current bundling algorithm could be implemented as added code in the Bitfile Server to ensure inter-cluster cohesion is maintainable. Another approach is to write a new application that could entirely replace the Bitfile Server. This application, if appropriately authorized, could access the internal storage class metadata structures of the Storage Server, and would also be able to provide application-specific COS attributes to interested storage consumers. As a Bitfile Server replacement, it would be able to create its own mappings of COS to storage class, thus explicitly controlling the actions of the Storage Server, and therefore able to enforce external partitioning decisions at the internal device level.

4. Simulation and Experimental Results

4.1 Measurements on hardware characteristics

We have performed detailed timing measurements on an Exabyte Carousel Tape System as well as an Ampex D2 Tape Library System, to validate our hardware model and also collect the appropriate parameters for the model. The results of our experiments are shown in Table1.

Note that the file overhead for the Ampex system is quite large (11 seconds, which is the time needed to transfer 141 MB) due to our lack of control over the behavior of the robotic system. In contrast, the Exabyte has a relatively small file overhead. In order to remain device independent, we treat each device as a black box and measure its performance. Our first experimental results from the Ampex were both larger than expected, and less consistent from file to file. We speculated that the speed of the device

might be high enough that a degree of inertia might be present in hardware, or that it might be presumed in software, such that consecutive file reads at high speeds were unattainable in our environment.

To test this hypothesis, we inserted a small pad file (1 KB) between each of the larger test files. This resulted in much faster, and much more consistent read times for the test files. Consequently, the average file overhead decreased from 11 to 3.25 seconds.

The effects of file overhead on query response times are discussed in the Section 4.3 below.

	Capacity (MB)	Transfer Rate (MB/s)	Seek Speed (MB/s)	Mounting (Seconds)	File Overhead (seconds)	File Overhead (MB)
Exabyte	4500	0.265	31.25	315	0.24	0.064
Ampex (high FO)	25000	12.864	503.32	39	11.0	141.5
Ampex (low FO)	25000	12.864	503.32	39	3.25	41.8

Table 1: Measurements of hardware characteristics

4.2 Response Time Results

Based on these hardware measurements, we were able to apply our partitioning algorithms to an actual PCMDI dataset. We could then compare the response times of queries before and after we apply the partitioning method. The partitioning method puts query types into groups, such that all query types in each group have at least one variable in common. Obviously, there is no need to consider the effect of a query type on another if they access no variables in common. This simplifies the presentation of possible solutions to the designer. The query types defined by the designer for the actual dataset are shown in Table 2 along with the amount of data that each needs to access.

query type	variable and dimensions specification	Megabytes requested
query types for group 1		
1	U,V,W for any month at ground level	76.2
2	U,V,W for any month at all height levels	1447.5
3	U,V,W for any day at any height level	1.6
4	U,V,W for any month at any level for any Y	3.0
5	U,V,W for any year at all levels for range of Y	2171.3
query types for group 2		
6	T for any month for all for all X,Y,Z	482.5
7	T for a range of 3 months for any height level	76.2
8	T for any sample on any height for a range of Y	0.07
query types for group 3		
9	A cloud variable for any month for all X,Y	50.8

Table 2: Query types and amount of data requested

Note that groups 1 and 2 have 5 and 3 query types, respectively, and group 3 has only one query type. It turns out that groups with only one query type are quite common, since for some variables there is only one predominant type of access. It is always possible to find an optimal solution for a query type belonging to such a group, since there are no potential conflicting query types. Therefore, we show here only one such representative group.

Note that the amount of data requested varies from less than a megabyte to over 2 gigabytes. Queries requesting small amounts of data are typically for visualization purposes, while those requesting large amounts of data are typically for summarization and post processing. Queries for intermediate amounts of data are typically needed for analysis, such as Fourier transformation. The post processing type queries are less important to optimize since they take so long that users might as well wait for an overnight processing.

Tables 3 and 4 show the response time for the 9 query types of Table 2. The response times are expressed in minutes, where "original" and "new" refer to the times before and after partitioning, respectively. The new timings on the Ampex were actual measured times on the real system, while all other times are calculated times based on the measured hardware model. It was impractical to run the queries before partitioning because their response time takes several hours. We also show the optimal times, which are calculated assuming that all the information to answer the query is in a single file, and that the file is positioned at the beginning of a tape. Thus, the optimal time is equal to the time to mount a single tape, plus the time to read the first file.

Query type	Megabytes requested	Optimal	Original	New	Ratio
1	76.2	6.45	174.97	6.45	27.13

2	1447.5	92.85	174.97	92.85	1.88
3	1.6	1.72	30.55	4.08	7.48
4	3.0	7.35	174.97	92.85	1.88
5	2171.3	274.27	2142.60	1118.72	1.92
6	482.5	32.01	174.97	40.83	4.28
7	76.2	6.45	535.65	6.45	83.05
8	.07	3.25	30.55	3.25	9.40
9	50.8	8.44	237.30	8.44	28.12

Table 3: Exabyte carousel response times (in minutes)

Query type	Megabytes requested	Optimal	Original	New	Ratio
1	76.2	0.75	9.80	2.73	3.59
2	1447.5	2.52	9.80	2.73	3.59
3	1.6	0.65	1.60	2.73	0.59
4	3.0	0.65	9.80	2.73	3.59
5	2171.3	3.47	117.00	29.07	4.03
6	482.5	1.27	9.80	10.67	0.92
7	76.2	0.75	29.30	1.90	15.42
8	.07	0.65	1.60	1.90	0.84
9	50.8	0.72	9.80	0.72	13.61

Table 4: Ampex D2 tape system response times (in minutes)

The dataset we experimented with, contained 57 variables (each defined over all or a subset of the seven dimensions X,Y,Z,S,D,M,Y) and 62 query types. These query types were derived after extensive interviews with scientists interested in this dataset. The query types were partitioned into groups as explained above, and each group was analyzed separately. The tables only show the query types that access the wind velocity vector U,V,W, the temperature T, and one cloud variable that had only a single query type associated with it. However, they are representative of the response times for other query types as well. In practice, most of the variables are accessed by a single query type, and only a few variables are accessed by 2-5 query types.

The original layout of the dataset was one where all the variables for all X,Y, and Z for a period of 5 days was stored in a single file. Files were stored one after another according to time, until the next file would not fit on the same tape. At this point a new tape was used and the process continued until all the data was stored. This storage method represents the natural order that data was generated by the climate simulation program, which, in general, is a poor organization for typical access patterns. The original response times were calculated on the basis of this actual storage of the dataset.

We note that all the new response times on the Exabyte are better than the original times. For the Ampex tape system, the improvements in response time are not as dramatic as in the case of the Exabyte tape system. The main reason for this is that the Ampex has a much larger file overhead than the Exabyte tape system. The large file overhead resulted in larger files being created by the partitioning algorithm. In the case of variables U,V,W, each file corresponds to one month of data for all X,Y, and Z, which comes to roughly 1.5 GB of data per file. And in the case of variable T, each file corresponds to roughly two Z levels of data for all X,Y, and T, which comes to roughly 700 MB of data per file. The consequence is that queries that ask for a small amount of data end up reading a single large file to answer the query. This is especially obvious for query types 3 and 8, where the result is that the response time is even slower than with the original data organization. As will be seen in the next section, the improvement of the file overhead from 11 seconds to 3.25 seconds made a significant difference in the results. In addition, our algorithm can take advantage of partial file reads (no file overhead) to further improve performance.

As expected, the partitioning for the cloud variable achieved optimal time as it was tuned for the single query type accessing it.

4.3 Effects of the file overhead

The experiments with the Ampex robotic system were performed after it was connected recently to NSL-UniTree. As was mentioned above, the file overhead was found to be quite large, about 11 seconds. Consequently, the size of each cluster was relatively large (about 200 MB), and the total number of clusters for this dataset was only 245 for a one year dataset. For the lower file overhead (about 3.3 seconds), which was obtained later, the size of each cluster was about 100 MB and the number of clusters about doubled. As discussed below, the lower file overhead improved the solution results significantly. In general, when the file overhead is small, and the number of clusters is larger, the response times tend to be shorter, because less unnecessary data is read for a given requested subset of the dataset.

To understand the effect of the file overhead better, we performed a simulation for the same set of query types, assuming the lower file overhead of 3.25 seconds and no file overhead at all. The effect of no file overhead can be achieved if the tape system permits partial file reads; that is, the system can seek to a position on the tape and read precisely the number of bytes requested. Thus, we can take the set of files (clusters) assigned to a tape and store them consecutively as a single physical file. This is indeed a feature that the Ampex system is capable of, and it will be exploited in the NSL-HPSS implementation (mentioned in Section 3.3).

The simulation results (in minutes) for the Ampex system are shown in Table 5, along with the original and measured response time that were shown in Table 4 for comparison purposes.

Query Type	Original	High FO	Low FO	No FO	High FO ratio	Low FO ratio	No FO ratio
1	9.80	2.73	1.58	0.75	3.59	6.20	13.07
2	9.80	2.73	2.68	2.53	3.59	3.66	3.87
3	1.60	2.73	1.58	0.75	0.59	1.01	13.07
4	9.80	2.73	2.68	2.53	3.59	3.66	3.87
5	117.00	29.07	25.70	24.60	4.02	4.55	4.76
6	9.80	10.67	8.88	8.19	0.92	1.10	1.20
7	29.30	1.90	0.86	0.75	15.42	34.07	39.07
8	1.60	1.90	0.86	0.69	0.84	1.86	2.32
9	9.80	0.72	0.72	0.72	13.61	13.61	13.61

Table 5: Effects of various file overheads on response time for the Ampex D2

As can be seen, the new ratios improved for all query types, some by a factor of 4, and the response times for all query types are better than the original times in both the case of the low file overhead (low FO) and the case of no file overhead (no FO). These simulation results show that systems that permit partial file reads perform better than systems that do not support that. But, even if partial file reads are not available, the gains that can be obtained by the partitioning algorithms are still very significant, especially in cases that the file overhead is low.

Our experiments confirmed, as expected that the lower the file overhead the better the results. However, we have learned from the experiments and simulations that even with a large file overhead the overall improvement of the partitioning algorithm is very significant. As was shown in Table 4, 6 of the 9 queries improved by a factor of 3.5 to 15, at the cost of 3 queries degrading by a small factor of less than 2. We also note that the lower file overhead of 3.25 seconds is still large compared to the Exabyte system. The file overhead is a function of the software that controls the tape system, and that it should be reduced to the extent possible to accommodate scientific applications. Of course, the best devices for this purpose are those that support partial file reads.

5. The Dataset Partitioning Workbench

Naturally, the scientist who will be using the dataset (or a database designer acting on the behalf of multiple scientists) is in the best position to know what are the typical queries that will be used and the frequency of their use. The dataset partitioner should optimize the reorganization of the dataset according to this information. Initially, we thought that letting the scientist specify a weight for each query will be a reasonable way of representing the relative importance and/or the frequency of use of each query. However, we found out that assigning weights is not a meaningful task for the scientists, and was confusing in practice.

Depending on the weights assigned to queries, different partitioning solutions can be found. The choice of a partitioning solution is very important since the process of

partitioning and reorganizing a large dataset is a costly one. Thus, it was important from a practical point of view that we develop methods for facilitating the process for selecting a solution based on intuitive measures rather than query weights. The result is an interactive "partitioning workbench". The goal of the workbench is to help the scientists see the trade-offs between possible solutions based on actual times that each query will take under a given solution.

In presenting the solutions to the scientist, we assume that all queries that belong to the same query type will have the same response time. In reality the response times to queries that belong to the same query type may vary somewhat depending on where the data relevant to the query is located on tape. This approximation is reasonable since each query that belongs to the same query type needs to access the same amount of data.

We observed that in order to make the estimates on actual times meaningful it is necessary to present them relative to the best possible time for each query. The best possible time (optimal time), is calculated assuming that all the information to answer the query is in a single file, and that the file is positioned at the beginning of a tape. Thus, the optimal time is equal to the time to mount a single tape, plus the time to read the first file. For example, if the best possible time for a query is 30 minutes, and the solution chosen results in 33 minutes, there is little to gain by trying to further optimize for that query.

We produce the multiple solutions presented to the designer as follows. We start by assigning all query types the same weight, and generate multiple possible solutions based on the permutation of the dimensions. Each solution consists of the estimates on actual response times for all query types, as well as the optimal response times. The solutions are ordered according to the overall response time relative to best possible times. The best solution is presented to the user as "option 1" as shown in the left part of Figure 5 (which is actually in color, but shown in black and white here). It shows an actual partitioning of a dataset for the Ampex robotic tape system. Six query types are considered for this group of query types (group 2) where the narrow bars shows the optimal times, and the thick bars the actual times for each of the six query types. As can be seen this option achieves optimal times for the first four query types, at the expense of the last two. Note that the user can control the scale of the display for better viewing of details.

At this point, the user can select any query type to be viewed on the right part of the screen for other possible options. For example, the scientist may want to optimize query type 5, and see what the effect will be on the other query types. The scientist selects query type 5 (by clicking on it) and all possible solution options are then displayed, as shown on the right part of the screen of Figure 5. As can be seen, there are three options that achieve optimal time for query type 5: options 2, 4, and 5. The scientist can now select any of these options (by clicking on the desired option) to see the effect on other query types. Suppose that option 2 was selected. The result is shown in Figure 6. As can be seen in the left part of the figure, the effect of selecting option 2 is that query type 5 achieves optimal time at the expense of slowing down query types 1, 2, and 4. This may be a preferred solution if indeed query type 5 is particularly important to optimize for.

The scientist can continue to explore other options and settle on the most desirable option for his/her needs. This form of interaction is much more meaningful to scientists than assigning weights. Seeing the trade-offs in terms of actual times makes the choice of a solution a more straightforward process. In some cases there may be a serious conflict between query types, in that selecting a solution that favors one query type may disfavor another, and vice versa. When such conflicts arise the scientist needs to make a difficult choice or resort to some duplication of data. We have not addressed so far the possibility of data duplication. It is the subject to future research.

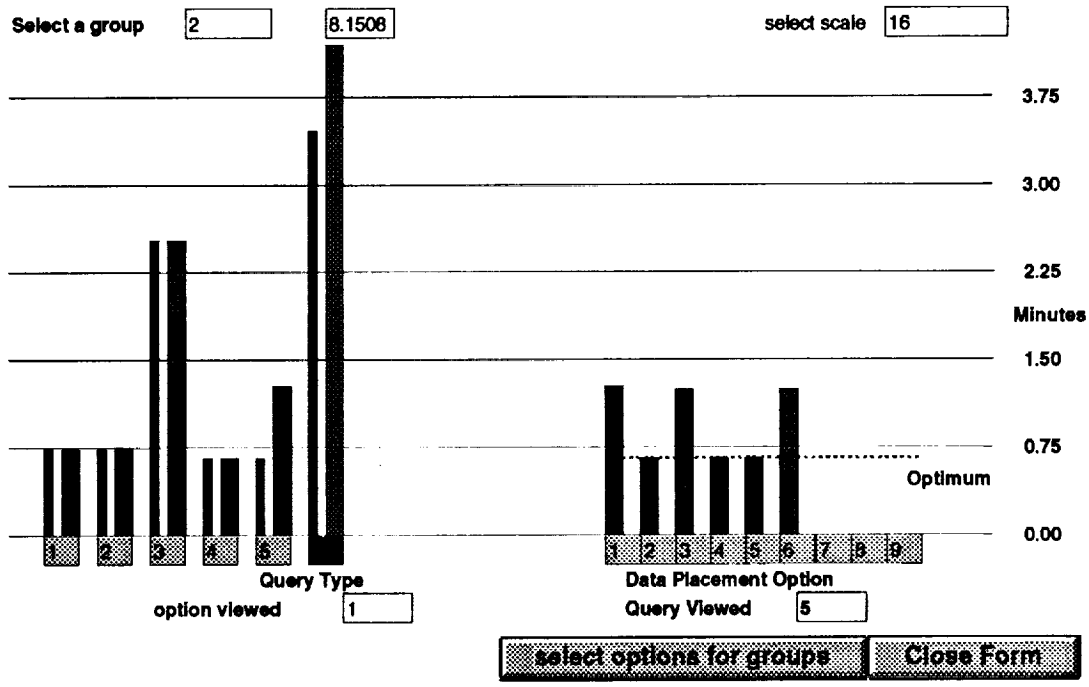


Figure 5: Display of a solution option for a group of query types

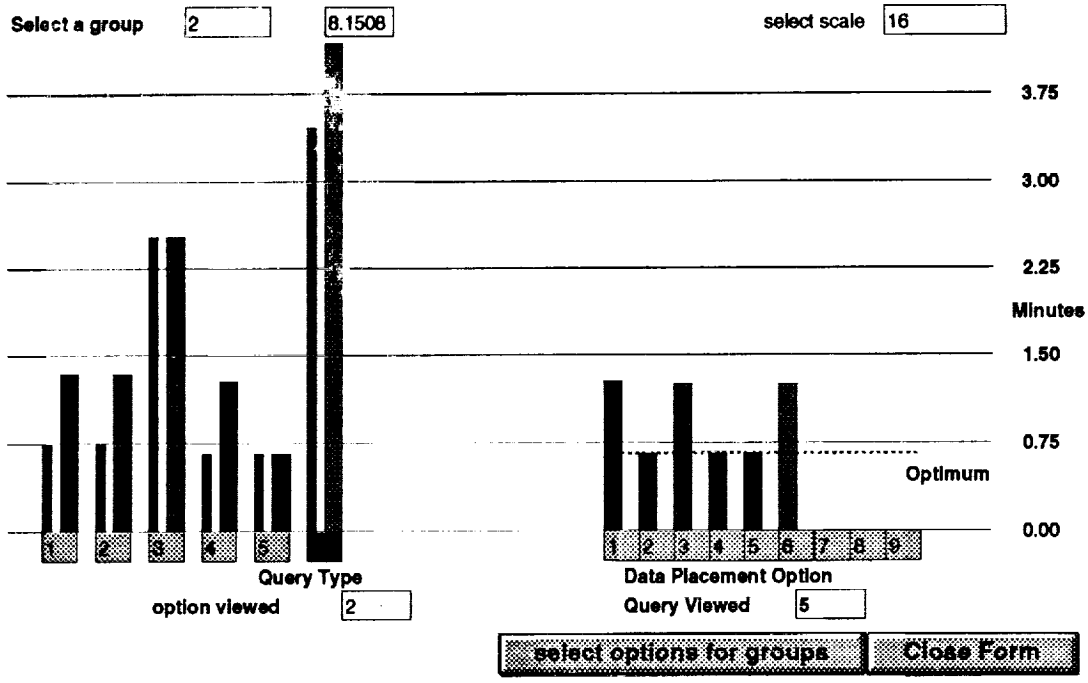


Figure 6: Display of a solution that optimizes query type 5

Once a desired solution is selected, the restructuring of the dataset into multiple clusters and their layout on tape volumes is generated by the workbench. This information is used by the write process to partition the dataset accordingly and to store the clusters on tapes.

The front end of the workbench was implemented using a relational database system (ACCESS) on a PC Windows platform that presents the user with a GUI. It is used to store up to 25 tables that contain the dataset information, the hardware characteristics, query types, and the various partitioning solutions calculated. The backend of the workbench is a collection of C++ programs that calculate the partitioning solutions, and the estimated response times. In addition to screens for selecting a solution discussed above, there are various screens developed for the scientists to enter or modify the information on datasets, hardware characteristics and query types.

6. Concluding remarks

Scientific application that access very large datasets face a major bottleneck when they need to access subsets of very large datasets from tertiary storage. This state of affairs has been the main reason that scientific analysis is not currently performed on an ad-hoc basis. Analysis cannot be spontaneous if a request for an interesting subset takes hours. Further, if the mass storage system is sharable by many users, the access of unnecessary data when subsets are requested, reduces the efficiency of such systems. As a result, supporting a certain user load requires additional physical devices such as read channels or larger robotic systems.

The approach we have taken is common in data management systems. To achieve higher efficiency of data access from disks, data is often clustered according to its expected use. We have chosen to work closely with a specific application area (climate modeling) where this problem is affecting the productivity and quality of the analysis. This gave us a realistic framework to understand the nature of spatio-temporal datasets and typical access to such datasets in modeling applications.

The results so far confirm the benefits of this approach. In realistic examples, it was possible to pinpoint typical access patterns and restructure the datasets to fit the intended use of the data. We found it useful to provide users with estimates of response times for making partitioning choices. Once a partitioning choice is made, users can estimate the response time to ad-hoc queries, and decide whether they want to wait for a response. Because many of the requests are for on-line visualization, the size of the subsets requested are small, and thus only a few clusters need to be accessed. In such cases, response time improved by a factor of 10-100.

There are many directions that one can take at this point. One is to consider the benefits of duplicating some of the data. In some applications, a small percentage of duplication can dramatically improve global response time when query types have inherent data partitioning conflicts. Another area is to consider more general access patterns and query types.

There is also the question of how generic such algorithms can be. We think that it will be necessary to specialize on application domains. However, selecting broad categories of data types, such as spatio-temporal data, or sequence data (e.g., time series), can make such techniques generally useful. The reason that we chose to concentrate on the spatio-temporal domain is that many disciplines are in this domain (geology, earth science, environmental sciences, etc.) and that spatio-temporal datasets tend to be very large (simulation data, satellite data, etc.).

Finally, it is worth mentioning that tape striping techniques are being investigated to mitigate the slow response time of accessing data from tape systems (see, for example [10]). In this approach no knowledge of access patterns is used; rather it is intended to take advantage of multiple tapes that are synchronized to be read in parallel. Striped tape systems will complement our partitioning techniques, in that clusters could now be spread over multiple tapes for parallel reads. The main gains provided by partitioning will continue to be important when we use such systems because they reduce the data that needs to be read for a given request.

Acknowledgment

This work was performed by the Lawrence Livermore National Laboratory under contract W-7405-Eng-48 and by Lawrence Berkeley Laboratory under contract DE-AC-76SF00098, and was supported by the Office of Scientific Computing of the Office of Energy Research, U.S. Department of Energy.

References

[1] Gates, W., Potter, G., Phillips, T., Cess, R., An Overview of Ongoing Studies in Climate Model Diagnosis and Intercomparison, Energy Sciences Supercomputing 1990, UCRL-53916, pages 14-18.

[2] EOS Reference Handbook, NASA document NP-202, March 1993.

[3] Coleman, S., Miller, S., Eds., Mass Storage System Reference Model, Version 4, IEEE Technical Committee on Mass Storage Systems and Technology, May 1990.

[4] L.T. Chen, et al, "Efficient Organization and Access of Multi-Dimensional Datasets on Tertiary Storage Systems", To appear in a special issue on Scientific Databases, Information Systems Journal, Pergammon Press, Spring 1995.

[5] R. Coyne, and R. Watson, The National Storage Laboratory: Overview and Status, *Proc. Thirteenth IEEE Symposium on Mass Storage Systems*, Annecy, France, June 13-16, 1994.

[6] S. Coleman, R. Watson, and R. Coyne, The Emerging Storage Management Paradigm, *Proc. Twelfth IEEE Symposium on Mass Storage Systems*, Monterey, CA, April 26-29, 1993.

[7] D. Teaff, R. Coyne, and R. Watson, The Architecture of the High Performance Storage System, *Fourth NASA GSFC Conference on Mass Storage Systems and Technologies*, College Park, MD, March 28-30, 1995.

[8] S. Louis, and D. Teaff, Class of Service in the High Performance Storage System, *Second International Conference on Open Distributed Processing*, Brisbane, Australia, February 21-24, 1995.

[9] R. Drach, and R. Mobley, *DRS User's Guide*, UCRL-MA-110369, LLNL, March, 1994.

[10] Drapeau, A., Katz, R., Striped Tape Arrays, Twelfth IEEE Symposium on Mass Storage Systems, 1993, pages 257- 265.

Building a COTS Archive for Satellite Data

Ken Singer, Dave Terril
Federal Data Corporation
4800 Hampden Lane
Bethesda, Maryland 20814
+1-301-457-5505
{ksinger,dterrill}@saa.noaa.gov

Jack Kelly
L.A. Systems
9027 Greylock Street
Alexandria, Virginia 22308
+1-703-360-1000
jkelly@nesdis.noaa.gov

Cathy Nichols
NOAA/NESDIS/IPD
Federal Building 4
Suitland, MD 20746
+1-301-457-5243
cnichols@nesdis.noaa.gov

Abstract

The goal of the NOAA/NESDIS Active Archive was to provide a method of access to an online archive of satellite data. The archive had to manage and store the data, let users interrogate the archive, and allow users to retrieve data from the archive. Practical issues of the system design such as implementation time, cost and operational support were examined in addition to the technical issues. There was a fixed window of opportunity to create an operational system, along with budget and staffing constraints. Therefore, the technical solution had to be designed and implemented subject to constraint imposed by the practical issues. The NOAA/NESDIS Active Archive came online in July of 1994, meeting all of its original objectives.

Introduction

The functional requirements of the NOAA/NESDIS Active Archive were quite similar to most other archives. The NOAA/NESDIS Active Archive had to perform the following functions: 1) provide a means to manage and store a great number of large datasets 2) give users access to interrogate the archive 3) give users the ability to retrieve data from the archive. In addition, the following technical features were also desired: scalability so new and future datasets could be included, a modular architecture to allow enhancements, and security since the archive was intended to be accessed across the Internet. All of these requirements and features could be implemented in a straightforward manner using hardware, software and a support staff focused entirely on creating the archive. The challenge faced in designing and implementing the NOAA/NESDIS Active Archive was to successfully accomplish the same task by using existing hardware to minimize cost, commercial off the shelf (COTS) software to minimize software development, and existing support personnel to reduce new staffing requirements.

The implementation of these functions and features had to be tempered by the fact that there was a window of opportunity to implement the archive, a limited budget, and other everyday work still to be accomplished. The greatest potential enemy was the goal itself, the design and implementation of the archive. If the design was too complex, it might take too long to implement and may never actually happen. If the design did not utilize existing hardware and software, cost might prohibit the project from moving forward. If "leading edge" became the buzzword for too many components, then the project would be throttled by the effort needed to bring these components into an operational state. If the skills needed by programmers and the support staff were not available, time for the training would extend the time needed for development. If too many changes to operational procedures and the "usual way of doing business" were needed, management might not agree to the proposed changes. By considering these issues ahead of time and understanding their implications, the solution had to avoid these "potholes" as much as possible.

Characteristics of Archive-Stored Data

The initial purpose of the NOAA/NESDIS Active Archive was to store (level 1B) datasets from the Advanced Very High Resolution Radiometer (AVHRR) instrument flown on NOAA's current series of polar orbiting satellites. In the future it is very likely that additional datasets from the current satellites and new datasets from future satellites will become candidates for inclusion. The AVHRR datasets vary in size from 50 to70 Megabytes (MB), so 60MB is used as an average for calculations. Each operational satellite transmits approximately 45 datasets per day. Today there are 2 satellites, NOAA 12 and NOAA 14, downloading AVHRR data. So daily data volume is 5.4 Gigabytes (estimated).

Issues Considered

The design of the NOAA/NESDIS Active Archive had to achieve a balance of the following issues: implementation time and complexity, overall system cost, commercial availability of hardware and software, reliability, future growth and scalability, and the migration path from existing systems

Solution Approach and Architecture

The solution selected takes advantage of the strengths of two different families of computers: the IBM mainframe and UNIX workstations. The mainframe offered high reliability, strong I/O capabilities, established connectivity to mass storage devices and time-proven Hierarchical Storage Management (HSM) software. UNIX workstations were chosen for their reasonable price for performance, the availability of tools for developing user interface programs, and strong TCP/IP performance for Internet user access and data delivery. So the function of data management and storage would be done by the mainframe, and the functions of interrogating the archive and retrieving the data would be handled by the UNIX workstations.

The function of interrogating the archive is done totally by the UNIX workstations, with no assistance from the mainframe. This was done for the following reasons. First, the performance of interrogations of the archive would be more consistent by maintaining, on the UNIX workstation, the database of metadata that describes each dataset. The mainframe is heavily used by many other batch-oriented jobs and thus has many periods of peak utilization. This could affect the response the user sees. Second, storage and generation of the browse images is done on UNIX. To assist the user in narrowing down the list of desired datasets, a browse image is provided on request. The underlying goal of

interrogating the archive is to help the user narrow down and limit the number of datasets that are of interest. This saves the user time because only the data truly desired is obtained. And, it helps to minimize the load on the data storage and management function because fewer datasets are requested. Finally, reliability of the system should be higher because each function is independent. If the archive interrogation function is not running, the data management and storage function can continue to accept incoming data. Or if the data management and storage function is temporarily unavailable, the user can still interrogate the database and submit requests to retrieve data (although the request may take longer to fulfill).

The approach described above provides scalability. One of the true strengths of the mainframe is the attachment and throughput to storage devices. This strength will probably continue well into the future, allowing for growth. In addition it is possible to have the archive interrogation programs query multiple data servers, which would be another technique to increase capacity. Modularity is also emphasized with this approach. The user interrogation function is separate from the management and storage of data. Similarly, retrieving the data is accomplished without the user seeing or having to understand the storage and management of the data. These abstractions allow changes to be made to any components of the solution without changes to the other components. For example, new tools for interrogating the archive can be added without affecting the storage and management of the data. Or, new data storage devices can be utilized without the user having to worry about how the data is retrieved. Security is also provided because no user can directly access the data storage and management function. Users only interact with application menus on UNIX, which then cause other events to occur elsewhere in the system.

An additional benefit of this approach is that the NOAA CEMSCS (Central Environmental Satellite Computer System) mainframe was already there, handling the ingest of these datasets, connected to a mass storage device, and with a knowledgeable support staff. So, part of the needed solution was in place and functioning. Plus, the processing that already existed on CEMSCS, such as creating other NOAA satellite data products, could utilize the data in the active archive as well.

While the hybrid approach offers many positive features, there are some tradeoffs. There is administrative overhead for coordinating the metadata database with the real archive. There could be missing entries or incorrect entries, each of which would cause different problems. Also, operational support issues are more complex maintaining a system that spans across two different computing platforms.

Solution Overview

The description of the solution is based on the functional requirements stated earlier: the ability to interrogate the archive, the ability to retrieve data from the archive, and dataset management and storage. The description of interrogating the archive and retrieving data will be in the section below labeled "Interaction with the Archive", while the dataset management and storage description will be in a section of the same name. Each section will cover how the function is implemented, as well as a description of the hardware and software used.

Interaction With the Archive

User Interrogation of the Archive

A user wants to see what is in the archive based on a set of criteria. For example, "Is there any data in the archive from the month of May, 1994 over Greenland?" The metadata provides these answers. The NOAA/NESDIS Active Archive provides access to the metadata through an application called the Satellite Active Archive (SAA). SAA is responsible for collecting the metadata after the dataset is available on CEMSCS. In addition, SAA also provides a series of menus to allow the user to query the metadata and view the results. The results are a list of identifiers that point to specific datasets in the archive. As a final aid to the user, SAA provides a browse image for each AVHRR dataset in the archive. The browse image is at a lower resolution than the actual data, but should be of great value to the user. For example, by viewing the amount of cloud coverage in a scene, the user can further reduce the number of datasets that are of interest.

Retrieving Data From the Archive

After the user has selected some datasets of interest, now the user wants to go and get the data. Once again, the user interacts with the SAA application. SAA presents a series of menus to allow the user to order the datasets. The user can order the data for electronic delivery or delivery on tape media. In addition, the user can order the complete dataset or an extracted portion of the dataset. Presently SAA imposes a restriction on the amount of data that can be delivered electronically to insure reasonable network performance. The extract capability makes this possible because the user can limit the size of the delivered data by reducing the geographic area desired or by requesting fewer channels of data.

After the user is done interacting with the SAA application, the dataset management and storage function is finally called into action. A job is submitted to the dataset management and storage function to retrieve the requested data. First, the hierarchical storage management software gets the raw data wherever it may reside. Then the extract function is performed to cut out a piece of the data and add proper headers. Finally, the SAA application sees that the dataset management and storage function has the data ready and waiting. Then SAA picks up the data and delivers it to the customer in the requested manner.

UNIX Work Station Functions

The IBM RS/6000 UNIX workstation running AIX was chosen to implement the UNIX based functions. Although most UNIX workstations could have performed the necessary functions, the RS/6000 was chosen for three main reasons: first, an existing contractor had strong knowledge and skills to support the RS/6000. Second, the RS/6000 offered ESCON channel connectivity to the CEMSCS mainframe. This could be used as a highly reliable, high performance point-to-point communication link using standard TCP/IP applications like FTP and NFS. Finally, the original basis for the front end application was the Global Land Information System (GLIS) from the US Geological Survey EROS Data Center (EDC) in Sioux Falls, SD. GLIS had been ported to the RS/6000, so the local software developers had a strong head start.

The front end application (SAA) that the user interacts with utilizes both ASCII and X-windows based screens. The SAA screens and menus were based on GLIS. Strong similarities can be seen between the systems today and may continue due to future cooperation. The cooperation and help provided by the EDC staff was of invaluable assistance in getting the SAA prototype off the ground so quickly. Similarly, the SAA metadata database uses INFORMIX, following the recommendations of EDC.

The electronic data delivery capabilities of SAA were a brand new function that had to be added since GLIS did not support electronic data delivery. The delivery functions are designed to provide reliability and flexibility for growth. Reliability was a necessity, since use of the NOAA/NESDIS Active Archive would probably be minimal if users could not be confident of receiving the data they had ordered. Flexibility for growth was important so the system could be up and running in a short time, but permit the easy addition of new functions such as FTP push and subscriber data delivery. Also, as usage grew, modifications may be needed to insure balanced network performance.

Work Station to Mainframe Communications

The solution architecture necessitates communications between the CEMSCS mainframe and the UNIX workstations on two occasions. The first is to update the metadata database and generate the browse images on the UNIX workstation. The second is to retrieve data from the physical archive for delivery to customers.

The update of the metadata database is accomplished in the following way. The UNIX workstation wakes up at regular intervals (presently every hour) and does a directory listing (using NFS) of high level qualifiers where the AVHRR data resides. This list is compared to a list of datasets already in the metadata database. The MVS naming convention uses the Julian date, so only one day's worth of data is examined at a time. If any new datasets are found, they are either copied to UNIX or processed across the NFS-mounted directory. Some datasets can be processed across the NFS-mounted directory because only the header needs to be read. A record of the metadata information is created and then entered into the database. At the same time, the browse image is created and stored.

The process for retrieving data is more complex. First, the SAA application on the workstation submits a request to the mainframe. This request is actually a JCL job that is submitted using FTP. The recall and availability of the dataset needed is handled transparently by the SMS (system managed storage)/HSM software on the mainframe. The JCL job runs the extract to subset the dataset if needed. The result is placed in a particular directory and named by the SAA order number. A suffix is added to the name to indicate whether the job is in progress, successfully completed, or failed. Next, the SAA application on the workstation does a directory listing from the mainframe using NFS. If a file named with the SAA order number and the proper suffix is found, that file is copied to the UNIX workstation from the NFS mounted directory.

At this point, the SAA Delivery Server manages the delivery of the data to the user. The Delivery Server is a program that is modeled after a "state diagram". Each order corresponds to a unique entry in the order database. The Delivery Server wakes up at a regular interval and performs a specific action on each order based on what "state" the order is in. For instance, a job may be submitted to CEMSCS, a check is made if the extract job is complete, the dataset is copied to UNIX, or the data is FTP'ed to the user. The Delivery Server has the ability to retry any state a specified number of times. If a failure is detected, a message is sent to the SAA system administrator.

Data Management and Storage

The Storage Server

The storage server used was the NOAA CEMSCS (Central Environmental Satellite Computer System) mainframe. CEMSCS is a multisystem complex of two IBM ES/9000

mainframes and associated peripherals. Workload is scheduled and resources are allocated by the IBM MVS Job Entry Subsystem (JES3). CEMSCS is used create level 1B datasets as well as many level 2 and level 3 products based on the raw data.

Software Utilized on the Storage Server

The storage server software selected on CEMSCS is based on IBM's mainframe Systems Managed Storage (SMS). The data archiving component of SMS is HSM, which has matured over twenty years of intense DP growth as a COTS solution. CEMSCS had previously elected to utilize HSM as a viable alternative to postpone and minimize expensive DASD acquisition. Since this product had proven successful in managing data for the CEMSCS environment, HSM was reviewed to see if it would satisfy the needs of the NOAA/NESDIS Active Archive. The concern was not the amount of data, which is measured in multiple terabytes, but rather the number of files retained. The number of files is in the area of many hundred thousands. The management of this large number of distinct files approached the limitations of HSM but recent changes addressed this concern. SMS has allowed CEMSCS to minimize personnel requirement, standardize storage retrieval and archiving methodologies, and isolate the installation from the ever changing hardware enhancements.

SMS attempts to optimize placement of data, according to installation directives. This process maximizes automation and minimizes the staffing requirements, but initially requires a higher level of expertise. Complex issues, such as data reference patterns, locality of reference, read to write ratios, etc., are minimized but not eliminated. Once the directives are established, the day-to-day process requires less expertise. SMS allows minute to minute evaluations about data to be made without requiring manual intervention while minimizing data access time.

HSM manages the retention and migration of data, again according to installation directives. HSM attempts to ensure that frequently referenced data is maintained on accessible storage while less frequently referenced data is maintained on alternate, less expensive storage media. Data may be migrated to less expensive DASD or tape media, depending upon installation criteria, e.g. data, size, importance, or residency time. Data compression can be optionally performed on the migrated data at either the software or hardware level.

The retrieval of data, *i.e.* moving data from a lower form of the hierarchy to a higher one, is performed with no user intervention. If the data has been migrated and it is referenced, then HSM automatically moves it to an accessible media. If the data has to be brought back from a non-DASD device, then a user can be notified that the retrieval may require an "extended" amount of time. With the inclusion of tape robotics, this extended time is less than ninety seconds..

Archive Storage Devices

Access to data must be accomplished from DASD. Once SMS has placed a file on an appropriate DASD device, the file remains on this device until it is migrated by HSM or deleted. Several different DASD media have been utilized at CEMSCS. DASD caching maintains response time, especially for the larger capacity devices. SMS allowed CEMSCS to easily create "pools" of DASD to satisfy the different requirements of the archive. In this context, a pool is simply a grouping of DASD for a specific purpose. As the archive development evolved and moved into operations, so did its requirements. To date, these changes have been easily addressed via SMS mechanisms.

CEMSCS created two pools of DASD for the archive. Initially the satellite data is placed in a pool of four IBM 3390-II and three IBM 3380-III where it resides for one to three days, depending on access. If the data is migrated from this initial pool and subsequently accessed then it is recalled to a separate pool of five IBM 3390-II. This smaller pool has a different migration and residency criteria than the initial pool. The archive concept is that current data is more likely to be accessed: therefore, keep it accessible and do not waste resources migrating it. After time, data is less likely to be accessed; therefore, the data can be migrated. If data is recalled, then the data is placed on the second pool. This method allows recalled data not to impact the management objectives of the current data.

Cost effectiveness has convinced CEMSCS to migrate to robotic tape subsystems. Two different vendors' robotics have been utilized, as well as two different tape media, IBM and STK, 3480 and IBM 3490E. The 3490E media is the media of choice for HSM's migration function. The 3490E has the capacity required for the archive and the 3490E has the ability to locate records on tape directly. HSM "understands" and takes complete advantage of these hardware enhancements of the 3490E device. Due to the nature of the satellite data, the hardware compaction (IDRC) available with the 3490E does not provide much benefit, less than three percent.

Currently SAA has captured a terabyte of data comprising 19K files. A subset of this data resides on the two DASD pools of 16 GB. The remainder resides on 1,100 3490E volumes.

ASF Archive Issues: Current Status, Past History, and Questions for the Future

Crystal A. Goula and Carl Wales
Alaska SAR Facility
University of Alaska
P.O. Box 757320
Fairbanks, Alaska 99775-7320
email 1: crystal@dino.gi.alaska.edu
email 2: cwales@iias.images.alaska.edu
Phone: (907) 474-7926
Fax: (907) 474-5567

Abstract:

The Alaska SAR Facility collects, processes, archives, and distributes data from synthetic aperture radar (SAR) satellites in support of scientific research. ASF has been in operation since 1991 and presently has an archive of over 100 terabytes of data. ASF is performing an analysis of its magnetic tape storage system to ensure long-term preservation of this archive. Future satellite missions have the possibility of doubling to tripling the amount of data that ASF acquires. ASF is examining the current data systems and the high volume storage, and exploring future concerns and solutions.

Introduction:

Synthetic Aperture Radar (SAR) is an imaging radar technique involving the use of an aircraft or satellite-borne antenna to obtain an artificial radar aperture effect by utilizing the forward motion of the vehicle. Using the movement of the aircraft or satellite, the antenna emulates a larger sized aperture antenna. The technique produces the results of a larger aperture antenna, and is especially important when size limitations would prevent using the physically larger antenna.

The Alaska SAR Facility (ASF) was established at the University of Alaska Fairbanks in 1986. Funded by NASA, ASF is dedicated to the collecting, archiving, processing and distribution of SAR data. The major data-handling systems in use today at ASF were developed by the Jet Propulsion Laboratory, and installed in Fairbanks in 1990. ASF first began collecting data from the European Space Agency's ERS-1 satellite in August of 1991. More satellites were scheduled for later dates. ASF receives SAR data in real-time and tape recorded transmissions from satellites, processes the data into other usable forms, archives and distributes the data, in accordance with NASA's international agreements. Because of these agreements, ASF must maintain the archive of raw data for ten to fifteen years after the end of the satellite's mission.

ASF can receive SAR data covering Alaska, eastern Siberia, the Arctic Ocean, the north Pacific, and northwestern Canada. ASF collects large volumes of raw SAR data and processes the data into images that scientific researchers use to study sea ice, oceanography, geology, glaciology and botany. The processed images add to the large data store at ASF. Data are archived at ASF for long term storage on high density magnetic tape.

Satellites:

ASF is currently collecting data from two satellites: ERS-1 (European Remote-Sensing Satellite-1) and JERS-1 (Japanese Earth Resources Satellite). Incoming data from ERS-1 and JERS-1 is approximately 1.2 terabytes per month, or over 14.4 terabytes per year.

SATELLITE MISSION INFORMATION

	ERS-1	JERS-1	ERS-2	RADARSAT	ADEOS
Launch Date	July 17, 1991	Feb. 11, 1992	Spring, 1995	Spring, 1995	Feb. 1996
Mission life	3 years	2 years	3 years	5.25 years	3 years
Number of Links*	2 **	2	2 **	2	3
Data rate (mbps)	105	60/60	105	105/85	60/60/6
Orbital Period	100.47 min	95.87 min	100.47 min	98.594 min	98.59 min
On-Board HDDRs	no	yes	no	yes	yes
Archive Data Beyond Mission	10 years	10 years	10 years	15 years	n/a

* X band only -- does not include S band.
** ASF only collects data from the high bit rate signal. ASF does not use the low bit rate signal.

The ERS-1 satellite is the European Space Agency's (ESA) first remote sensing satellite. ESA launched ERS-1 in July of 1991. ERS-1 transmits data at 105 megabits/sec (or approximately 13 megabytes/sec). ERS-1 data passes last up to fifteen minutes and ASF records multiple datatakes per day. The average number of data-collecting passes per day is nine, which yields 41 minutes per day of data. ASF collects approximately 32 gigabytes per day from ERS-1. The anticipated mission life of ERS-1 was three years. At present, ASF is still collecting data from ERS-1. ERS-1 will be decommissioned after ERS-2 is operational.

The JERS-1 satellite is one of the Japanese space agency's (NASDA) Earth Resources Satellite. JERS-1 was launched in February of 1992. The JERS-1 satellite has an on-board tape recorder and can transfer two streams of data simultaneously. The JERS-1 satellite also has two sensors on-board, one SAR and one optical. The optical data are recorded and sent on to NASDA for processing. The SAR data are archived at ASF and sent to NASDA. Data are transferred at 60 megabits/sec (or 7.5 megabytes/sec), regardless of whether the data are real-time or recorded. ASF collects on an average of four passes per day, which averages about 19 minutes per day. ASF collects approximately 8.5 gigabytes from JERS-1 daily. The anticipated prime mission life was two years, however, JERS-1 is in extended mission phase, which could last for seven more years.

The ASF Facility:

All the ASF components are important to the successful operation of the facility. Focusing on how the data get to the archive and how the data are retrieved narrows the number of departments down to those two directly involved in the physical archiving processes, which are the Receiving Ground Station (RGS) and the SAR Processing System (SPS).

Functional Diagram - ASF Today

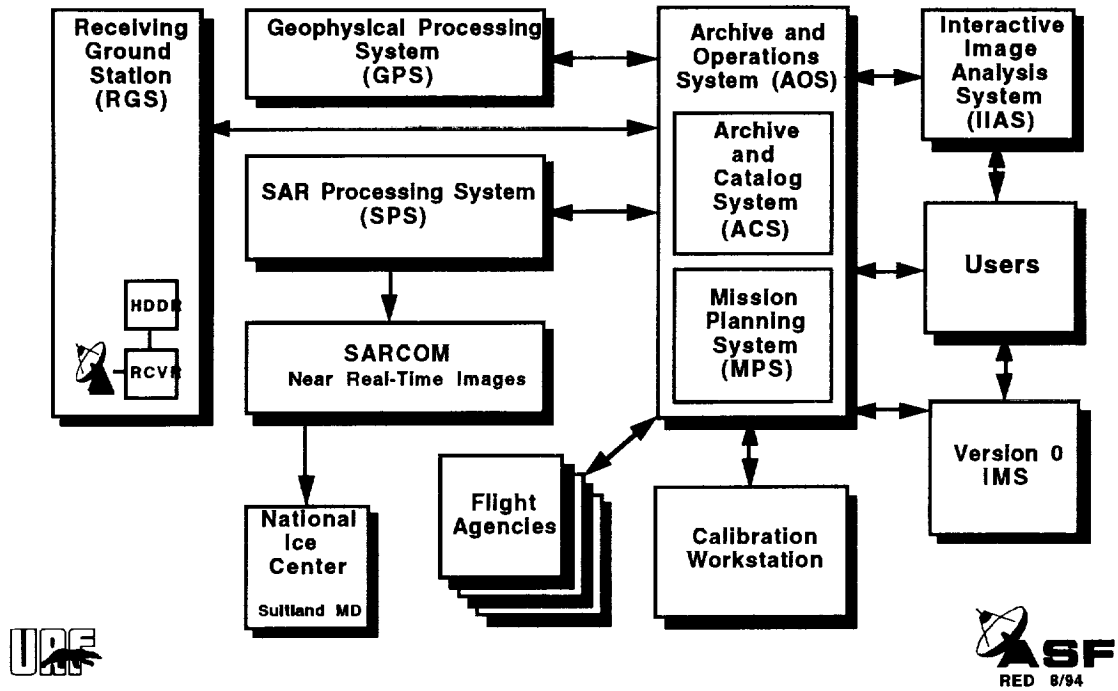


FIGURE 1 -- CURRENT ASF FUNCTIONAL DIAGRAM

Receiving Ground Station:

The RGS (Receiving Ground Station) tracks satellites with a 10 meter tracking antenna. Using high speed, high density recorders, the RGS then receives and stores the data from SAR satellites for later use by the processing system in support of researchers. The raw signal data recorded and stored during this process are considered level 0. ASF records two tape copies of the raw signal data. One tape is designated as the Archive Signal tape and put into storage as a backup to the second tape designated as the Working Signal tape. The Working Signal tape is used for data retrieval and processing operations.

Currently, ASF uses Honeywell HD96 and AMPEX DCRSi tape recorders to record incoming satellite data. In the case of the ERS-1 data, the DCRSi recorders record both the Archive Signal tape and Working Signal tape. With JERS-1 data, the situation is slightly different because of the on-board tape recorder. Data intended for NASDA are recorded on HD96 recorders, while SAR data to be used at ASF are recorded on AMPEX DCRSi tapes. For both the HD96 and DCRSi, data are recorded to the recorders in the serial mode.

The HD-96 reel tapes will hold about 15 minutes or approximately 12 gigabytes of raw signal data. The DCRSi tapes will hold 40 to 50 minutes or 47 gigabytes of raw signal data.

SAR Processing System:

The SAR Processing System (SPS) reads and decodes the raw data into image products. Data that have been processed by the SAR processor are considered level 1 data.

The only tape recorders connected to the SPS are the AMPEX DCRSi recorders. The data are accessed in parallel mode. ASF cannot retrieve data from the HD96 tapes unless the data were recorded or copied on to DCRSi tapes.

Raw data can be processed into full resolution images and low resolution images. It takes 195 megabytes of raw data (approximately 15 seconds of data transfer) to make one ERS-1 full or one low resolution image. The ERS-1 full resolution image is 8k x 8k pixels and covers an approximate area of 100 km x 100 km. JERS-1 raw data size is slightly smaller. The full resolution image is slightly smaller also, covering an approximate area of 100 km x 75 km. Processing the raw data into a full resolution image generates a file approximately 64 megabytes in size. By taking an 8 x 8 average of the full resolution image, the full resolution image can be processed into a 1k x 1k pixel low resolution image that takes up approximately one megabyte file space. To date, ASF has processed over 100,000 full resolution images.

It takes on the average 20 minutes to process one minute of raw data. With a datatake lasting anywhere from six to fifteen minutes, the processing of one datatake can run 120 to 300 minutes. One DCRSi tape can hold ten to twelve passes. The access time from end to end of an AMPEX DCRSi tape is five minutes. To process one DCRSi tape would take over twenty hours.

Archive:

ASF is collecting approximately 1.2 terabytes per month. ASF's archive consists of DCRSi tapes only. Currently, there are 980 Archive signal tapes and 1162 Working signal tapes on compact shelves at ASF. ASF also stored full resolution images on DCRSi tape. There are 237 of these image tapes in the archive. ASF currently has approximately 96 terabytes in the archive on Archive and Working Signal tapes. With another approximate 10 terabytes of full resolution images, ASF's current data storage totals 106 terabytes.

Along with the DCRSi main archive, ASF also has low resolution images archived on 12" optical platters in a jukebox. Currently, ASF has over 146,000 low resolution images stored on the optical platters, totaling about 146 gigabytes.

Use of AMPEX DCRSi Recorders at ASF:

ASF presently has six AMPEX DCRSi recorders on site. Three of the recorders are dedicated to the RGS and three are dedicated to the SPS. A recorder can be switched between subsystems, if needed to keep ASF operational. Two of the recorders were delivered to ASF in January of 1989. Three other recorders were delivered around November of 1990. The sixth recorder arrived in May of 1992. The sixth recorder is slightly different from the other five recorders. AMPEX had made some design modifications on the recorders by 1992. One of the new features on the sixth recorder is a low tension tape transport. This feature lowers the tension on the tape in the recorder and reduces the amount of stretching and fatigue on the DCRSi tape. Another one of the new features is a wide tip scanner. This feature improves the ability of the recorder to read from and write to the tape. The sixth recorder is currently installed on the RGS.

Figure 2 shows a simplified layout of how data are recorded on the DCRSi tapes. Two tracks are recorded longitudinal. The control track is used by the AMPEX recorder, and ASF does not do anything with this data. The user log and coarse address track is used by the AMPEX recorder when searching for data to get the tape roughly to a requested address. The user log contains information such as satellite identification, type of data, and the satellite revolution number. The recorder uses the coarse address information to get the tape near the specific address. The recorder will then read the transverse data to locate the specific address. The RGS computer will record the beginning and ending scan addresses from the DCRSi, so that the data can be retrieved later using the SPS. An example of a coarse address is 100100, while an example of a scan address is 100112. The DCRSi recorder has a bit error rate (BER) of 1×10^{-7} or better. DCRSi tapes are rated for 500 passes in low tension tape recorders, and 200 passes for high tension tape recorders.

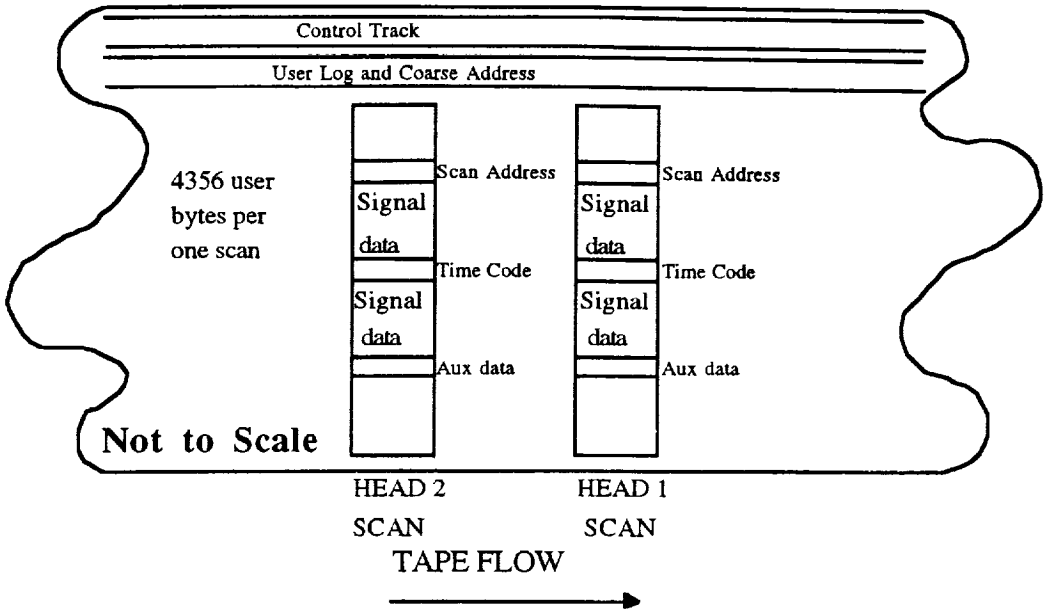


Figure 2 -- Simplified Data Scan

Figure 3 shows a simplified diagram of the scanner head and tape assembly. The data are written on the DCRSi tape from the scanner while both scanner and tape are moving. Although the tape is moving, the tracks of data are written in transverse mode. The scanner is moving fast enough that the tracks are only .1 slanted from the horizontal, and so is considered transverse. The scanner is rotating at a speed of 512 rps. The linear tape speed is 5.28 ips. There are six heads on the scanner. Each head will scan a track of 4374 bytes of data onto the tape, consisting of 4356 bytes of user data and 18 bytes of addressing/time code data. As each head will write one track of data per revolution, the recorder writes approximately 3073 tracks per second, with an average of 582 tracks per inch of tape.

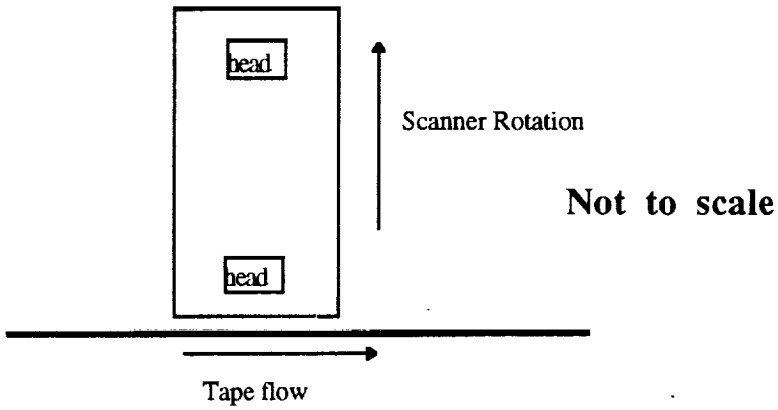


Figure 3 -- Simplified AMPEX Head/Scanner Assembly

ASF invests considerable effort to support this suite of recorders. ASF has two staff members, trained by AMPEX, to perform standard maintenance and repairs on the DCRSi

recorders. Every attempt is made to standardize all measurements on the recorders. ASF performs weekly maintenance on the DCRSi recorders, including a crossplay test. Using a single tape, a crossplay test pattern is recorded on each recorder. The tape is then tested on each recorder. A computer records the differences and errors in each of the test patterns. This process shows the staff which machines require any type of head phase adjustment. The standard crossplay testing is done in the serial mode.

ASF maintains a supply of spare parts and boards on site. There is a backup tape transport assembly for on site replacement. If the recorder cannot be repaired because ASF's spare is in use by another recorder, ASF must contact AMPEX to see if AMPEX has a spare part in their supply. If AMPEX does, they will ship the part to ASF. ASF will replace the part and send the damaged part back to AMPEX. ASF usually gets the working part in a couple of days, so down time is minimal. If AMPEX does not have a spare part in their supply, ASF's part must be sent to AMPEX for repair, which can take at least a month, but typically more like two to three months. Of the six AMPEX DCRSi recorders on site, on the average five recorders are functioning at any given time.

Since 1991, ASF has accumulated over 106 terabytes of data and has supported the processing of over 100,000 image products for science and operations users nationally and internationally. ASF is investigating several operational issues regarding the combination of tape recording systems and data-handling systems, which may be helpful to other users.

Retrieval of Signal Data:

ASF is experiencing problems retrieving raw signal data from the AMPEX DCRSi tapes after only three years of data collection. When recording satellite data, any number of reasons during transmission could cause a data dropout on the tape. Generally, an operator watches the RGS equipment during a download of data. In particular, the operator is watching a spectrum analyzer to make sure a good X-band signal is being received. If the X-band signal drops during recording, this is considered a data dropout, and is hand-noted in the RGS log for reference later. The signal loss indicates that there could be more than one segment of data to process. If there is a problem during processing, the RGS log is consulted to determine if an operator noted a problem during the satellite datatake. Under normal operations, the processor reads a Working Signal tape, finds a code indicating the beginning of a datatake, and accesses the data until a code indicating the end of the datatake is found.

When the SPS tries to read the signal data, and reports back more than one or two segments of data, the RGS log is checked to see if there was a problem when the data was recorded. If not, then there is a problem with retrieving the signal data. A processing data gap occurs when the SPS loses synchronization with the scan address codes in the raw signal datatake, meaning that the next scan address is not what the SPS was expecting. The SPS assumes that this is the end of the segment and searches for an ending code. Finding none, the SPS will then search for a new beginning code. Since this problem is occurring in the middle of a data segment, the SPS will find neither codes. The SPS will then assume that that was the end of the data, and begin processing the next section of the data segment that the SPS can read, using the same beginning code. An operator will manually cancel the request to retrieve this datatake with the number of segments goes above six. Figure 4 shows what happens when the SPS loses sync with the data on the tape. Approximately 5% of the datatakes processed to date have shown problems during retrieval. There was no indication of physical damage to the tape, nor does any data suggest that ASF has accessed this tape more than 200 times.

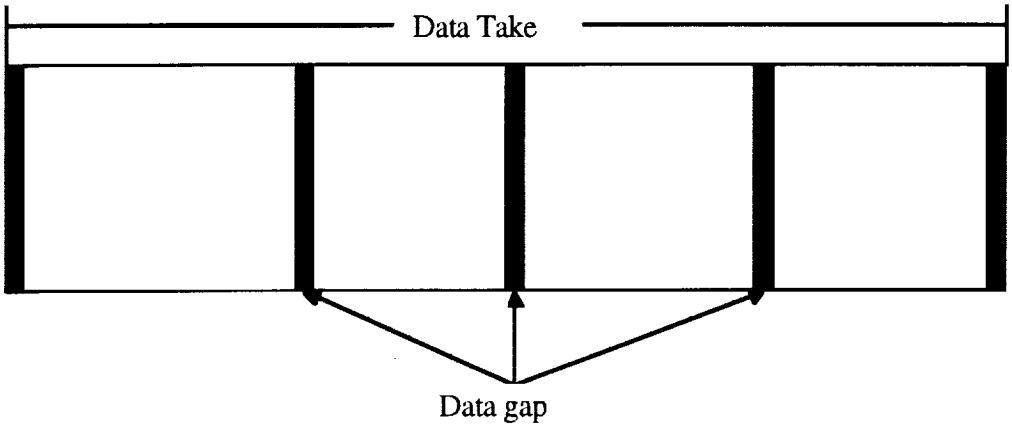


Figure 4 -- Data gaps on a single segment datatake

Normally ERS and JERS data will have no more than three planned segments per datatake, with the typical number of segments being one or two. The sync loss problem can turn a one or two segment datatake into a twelve segment datatake. A data gap is where the SPS loses sync with the scan addresses. There are two ways of trying to recover the data in the data gaps. One method involves moving the tape to another recorder and trying to access the data there. Different tape machines will read the same tape differently. There is no guarantee that changing drives will solve the problem, and switching drives to reprocess the data is a time consuming operation. At one point, ASF scanned the Archive Signal tape first to see if errors would come up in the same spot on the Archive signal tape as the Working Signal tape. If a scan of the Archive Signal tape experiences the same errors in the same spots, ASF assumes that a dropout in data did occur during the original recording of the raw signal data. The most expedient and most successful method of fixing the problem is to dub that datatake from the Archive Signal tape to another tape. Because the process of dubbing the Archive Signal tape to another tape was so successful, ASF has eliminated the scanning step. To date, ASF has dubbed approximately 225 datatakes. These dubs have added an additional 22 DCRSi tapes to the archive. By using one or both of these solutions, ASF resolves the data problem about 95% of the time. ASF has been unable to retrieve less than .3% of the archived raw signal data.

After consulting with AMPEX about the symptoms, the problem was diagnosed as a head phase and channel gain/equalization problem. This problem is the result of crossplay: recording the tape on one machine and trying to play the same tape on another recorder. One of the factors that affect the playback of a recorder is channel gain and equalization. The channel gain and equalization settings affect the reading of the data. The channel gain refers to the amplitude of the data signal. The channel equalization minimizes the errors in a data signal. By adjusting these settings, the number of bit errors can be reduced. Adjusting the channel gain and equalization settings is a way of optimizing the recorder's performance, but it is a time-consuming process. Another part of the problem with crossplay is head phase. The head phase on one machine is going to be slightly different from the head phase of another machine. This means that the timing control for a specific head will turn the head on either before the head has reached the tape data or while the head is in the middle of the tape data. Trying to read the tape when the head is not where it should be results in the dropout-like error. The head phase and channel gain/equalization are set when a scanner or transport assembly is replaced. It is not part of the weekly maintenance to check these settings. If ASF experiences multiple data retrieval problems with a recorder, ASF will check the gain/equalization and adjust if necessary.

AMPEX informed ASF that the new AMPEX DCRSi 107 should solve this problem. The new AMPEX 107 auto adjusts play alignment to the tape. Playback alignment may improve a high bit error rate.[6] ASF is in the process of acquiring a 107 model from AMPEX to verify if this would indeed solve the problem. The play alignment feature makes internal adjustments to the following: gain, equalization, clock phase, and tracking. The playback command is issued to the 107, but the adjusted settings are not permanent. If the save command is not issued, a reset or power off will clear these settings out of the memory and return the recorder to the original operating range.

Retrieval of Image Data:

ASF also archived the full resolution images output by the SAR processor. ASF was having problems completing approximately 14% of the full resolution image requests. The SPS would encounter problems when retrieving image data from an Image Archive tape. This problem was almost exclusively an addressing problem. There are two variations of the full resolution images addressing problem.

The JPL designed system software treats the tape drive as a disk drive, by preaddressing the tape. The preaddressing of a tape simply involved writing sequential addresses on the DCRSi tape. After an image was recorded on the DCRSi tape, there was the chance that the DCRSi would not write over the preaddressed address, causing a discontinuity between the legitimate addresses of the images. When trying to retrieve the images, the recorder would read one of the preaddressed addresses, which was not contiguous with the addresses for the images. The software was not designed to handle this problem, and after a few tries to retrieve the data, the process would stop. The result was that an image could not be retrieved because the correct address could not be found.

The second part of the addressing problem involved potentially corrupt scan addresses. AMPEX told ASF that there was a possibility that the scan address in the transverse data could be corrupted. If the scan address was corrupted, the software would not be able to find the exact address where it should be and the image retrieval process would be stopped. This problem had the same results: the image could not be retrieved because the correct address could not be found.

An additional problem with retrieving the image data was the bit error rate (BER). For raw data the acceptable BER is $3x10^{-5}$. For the full image data the acceptable BER is $1x10^{-9}$. The rated BER of the DCRSi is $1x10^{-7}$ or better. The two orders of magnitude between the raw data and rated BER allows a margin for error in the data. Because the full image demanded a much lower BER, the recorders would have to operate above the rated BER all the time, which is not a reasonable expectation. Even without the preaddressing problem, ASF believes that the full image data would have been more difficult to retrieve because of the low tolerance for errors in the data.

Originally ASF attempted to solve the addressing problem. Changes were made in the software to bypass the preaddressing issue, but the secondary corrupt scan address problem persisted. A fix to the corrupt scan address problem was discussed. It would have been possible to modify the software to allow an operator to back the tape up to a readable address, and then skip forward the expected number of bytes between this readable address and the requested address, however this fix would not have solved the BER issue. Because of these problems and other constraints, ASF abandoned the archive in August 1994. ASF decided that it would be more efficient and more successful to

process an image when a user requested it, rather than archive the image, use up limited archive space, and try to retrieve the data when and if a user requested the image. ASF is changing to a process-on-demand strategy, where signal data will be processed to image data, delivered to the user and no longer archived. This change is being made for several reasons, including budget constraints, space limitations, and evolution of the entire data system. While this change in strategy "solved" the problems with the full resolution image archive, it may introduce additional problems in the length of the archive life by increasing the frequency of access to signal data.

Archive life:

The estimated shelf life of the AMPEX DCRSi tapes is at least fifteen years.[1,7] During shelf life, degradation of the magnetic coating will eventually lead to unreadable tapes. To date, no deterioration of the archive as a function of age has been detected. Reading an archive tape will also cause degradation of the magnetic coating. As the DCRSi tapes are rated for 500 passes in low tension machines and 200 passes in high tension machines before suffering loss of data, increased access to the archive tapes will hasten their decline. This is especially true since all but one of ASF's recorders are high tension machines. ASF is migrating to a process-on-demand strategy where each time an image is requested by a user, the Working Signal tape will be accessed to process the image and satisfy the request. This will put increased wear on the tapes, which could shorten their lives. In turn, the increased wear on the Working Signal tape would also increase the frequency of duplicating from the Archive Signal tape. Also, for long term storage, tapes should be rewound every one to five years to relieve stresses in the pack.[1] Every access to Archive Signal tape increases the risk of damage.

Because of the high speed access of the DCRSi recorders, catastrophic damage to the tapes could result in loss of all data on the tape. Catastrophic damage includes broken tape, tape stretch, or severe crinkle in tape that could catch as the tape passes the scanner. Even the act of dropping a tape cartridge could damage the data on the tape.[1] Minor damage, such as minimal tape edge crinkle, could result in the loss of the information where the damage is, but the rest of the tape should be readable. This makes the archive "fragile" in the respect that any physical damage to the Archive Signal tape could result in loss of irreplaceable data. Because the Archive and Working Signal tapes are stored in the same room, any damage to the current storage area, such as fire or water leakage, could lead to loss of data as well.

Future Focus:

The other satellites in the Satellite Mission Information chart are future data sources. The addition of ERS-2 and RADARSAT in 1995 will at least double, if not triple, the data volume ASF is currently handling. Because RADARSAT also carries an on-board recorder like JERS-1, multiple data streams could also be possible, which would also affect the incoming data volume. With the new satellites, incoming data will increase from approximately 1.2 terabytes per month to between 2.3 to 3.3 terabytes per month.

Along with the new satellites, ASF will be archiving and processing data collected at the McMurdo station in Antarctica. ASF anticipates that McMurdo will send approximately 800 DCRSi tapes every six months. This equals approximately 36 terabytes per shipment. Because these tapes are ASF's only copies, ASF will have duplicate the tapes when they arrive to produce a Working Signal tape. The originals from McMurdo would become ASF's Archive Signal tape.

Soon ASF will add two Sony ID1 recorders to the RGS. In approximately one year, ASF will add six more ID1 recorders and a second 11 meter tracking antenna to the RGS. The six ID1 recorders will be part of the ADEOS satellite recording process. The two Sony ID1 recorders will replace the DCRSi recorders for ERS-1 SAR data collection. The ID1 recorders will record both the Archive Signal and Working Signal tapes. There is no present plan to convert the existing DCRSi Archive and Working Signal tapes to ID1 tapes. The ID1 uses a DD-1 medium tape, which holds about 40 minutes of raw signal data, approximately 41.2 gigabytes and has an end to end access time of less than 90 seconds.

For long term planning, ASF is considering the following factors: large volumes of data, long term archive responsibility, high download data rate, ease of operation, maintenance, and the access and retrieval necessary to support production and distribution of data.

Solid state memory and disks have faster access times than tape, however, they are generally not economically feasible. Although research is making progress in the high capacity disk storage to reduce costs, beyond a certain point disk storage is still not economically sound. Tape, either magnetic or optical, are still the most likely storage methods for ASF.

Optical tapes tend to have a higher storage capacity than magnetic. Both tapes have similar access speeds. Improvements in magnetic tapes have made some tapes capable of lasting more than twenty years, but the standard for magnetic tapes still seems to be ten to fourteen years. Durability during data reads is another factor to consider. Optical tapes seem to be more durable. Tests on the ICI 1012 optical tape reel, have shown that the tape can withstand 250,000 passes with no degradation of data, while magnetic tapes are typically 2,000 to 40,000 passes.[2,3] The optical tape systems have slower write speeds than magnetic tape systems. Existing laser and media technologies achieve a write speed of approximately 3 megabytes/second (24 megabits/second).

Robotic silos would reduce labor costs of some of ASF's archive and data retrieval process. The AMPEX DCRSi recorders cannot be used in a robotic silo. Robotic silos have a range of capabilities to assist in archive and retrieval of data. The drives for receiving and play back of satellite data could be attached to the same silo, however for better archive protection, two separate silos, one for Working Signal tapes and one for Archive Signal Tapes, will be investigated. The tapes could be passed between silos, and used on different drives, so that if all the recorders in one silo are occupied, the data tape can be passed to another silo with an available recorder. The access time in a silo involves accessing and mounting the tape. In some cases, manufactures also include drive preparation time. Access times range from four to eighteen seconds. The number of tapes that a silo can store varies with manufacture. Some silos can hold 6000 tapes (such as StorageTek Powderhorn), while others hold only 200 tapes. The type of tapes used in silos varies as well, from VHS to IBM 3480.

For future data storage improvements, ASF will be looking at archival aspects, such as media life and durability, volume of media, and robotic possibilities. Other factors such as write speed and cost of the equipment to purchase, maintain, and operate will also be important.

Summary:

As the volume of data at ASF continues to grow, the current data handling systems at ASF will be stretched to the maximum. With the data volume more than doubling in the next

few years, ASF is examining the current data handling systems. From operational experience, ASF has a new understanding of the AMPEX DCRSi recorders and how they function in the current data handling systems. This understanding has led to some operational and program changes at ASF, but these changes may not be enough to accommodate future data handling and storage requirements. Increases in data volume and frequency of data access will affect ASF's data handling and storage systems. Machine cost, machine capabilities, media life expectancy and durability, archive safety, and robotic capabilities are some of the factors that ASF will consider when planning equipment improvements to the data storage and handling system. With careful planning, ASF will insure the protection of the irreplaceable data collection for future scientific research.

References:

1. John Berbert, Ben Kobler, P.C. Hariharan, Jean-Jacques Bedet, and Alan M. Dwyer, "Magnetic Media", Greenbelt, MD, August 1993.

2. John M. Howard and Mark Hewish, "Data, Data, Everywhere: Competing Approaches to Mass Storage", International Defense Review, vol. 25, no.6, PP 75-79, June 1, 1992.

3. Storage Technology Contingency Plan for the ECS Project, EOSDIS Core System Project, August 1993.

4. David Cuddy, Eugene Chu and Tim Bicknell, "Alaska SAR Facility Mass Storage, Current System", Third NASA Goddard Conference on Mass Storage Systems and Technologies, Benjamin Kobler and P.C. Hariharan, eds., College Park, MD, October 1993.

5. Gunter Weller and W.F. Weeks, "The Alaska SAR Facility: An Update", Arctic Research of the United States, vol. 2, no.1, PP 27-31, Spring, 1988.

6. DCRSi 107 lab instruction manual, AMPEX Corporation, B5-B10, 1994.

7. DCRSi Media Archival Properties, AMPEX Corporation, 1994.

8. AMPEX DCRSi 2471/2473 Specification Summary, AMPEX Corporation, 1994.

Acknowledgments:

We would like to acknowledge Ron Faust, Thom Reimers, Brett Delana, and Tom George for their time and generous assistance with this paper.

Architecture & Evolution of Goddard Space Flight Center Distributed Active Archive Center

Jean-Jacques Bedet, Lee Bodden, Wayne Rosen, Mark Sherman
Hughes STX
7701 Greenbelt Road, suite 400
Greenbelt, MD 20770
301-441-4285 Fax (301) 441-2392
{bedet, bodden, rosen, sherman}@daac.gsfc.nasa.gov

Phil Pease
NASA/GSFC
Greenbelt Road
Greenbelt, MD 20771
301-286-4418
pease@daac.gsfc.nasa.gov

Abstract

The Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC) has been developed to enhance Earth Science research by improved access to remote sensor earth science data. Building and operating an archive, even one of a moderate size (a few Terabytes), is a challenging task. One of the critical components of this system is Unitree, the Hierarchical File Storage Management System. Unitree, selected two years ago as the best available solution, requires constant system administrative support. It is not always suitable as an archive and distribution data center, and has moderate performance. The Data Archive and Distribution System (DADS) software developed to monitor, manage, and automate the ingestion, archive, and distribution functions turned out to be more challenging than anticipated. Having the software and tools is not sufficient to succeed. Human interaction within the system must be fully understood to improve efficiency and ensure that the right tools are developed. One of the lessons learned is that the operability, reliability, and performance aspects should be thoroughly addressed in the initial design. However, the GSFC DAAC has demonstrated that it is capable of distributing over 40 GB per day. A backup system to archive a second copy of all data ingested is under development. This backup system will be used not only for disaster recovery but will also replace the main archive when it is unavailable during maintenance or hardware replacement. The GSFC DAAC has put a strong emphasis on quality at all level of its organization. A Quality team has also been formed to identify quality issues and to propose improvements. The DAAC has conducted numerous tests to benchmark the performance of the system. These tests proved to be extremely useful in identifying bottlenecks and deficiencies in operational procedures.

Introduction

The GSFC DAAC is being developed in several phases with Version 0 (V0) being developed to support existing and pre-Earth Observing System (EOS) Earth science data sets, facilitate the scientific research, and test EOS Data and Information System (EOSDIS) concepts. This paper presents the GSFC DAAC V0 missions and requirements, and describes its architecture at the software and hardware level. The ingest, archive, and distribution processes are explained and a walk-through of these functions is presented. Numerous tests have also been conducted to benchmark the performance of storage devices, specific functions (e.g., ingestion), and the overall system. The tests which helped identified deficiencies in operational procedures and software are described. The Hierarchical File Storage Management System, Unitree, is a critical component of the DAAC. Some major issues were discovered during the integration of Unitree with the GSFC DAAC hardware and software, and a list of lessons learned has been compiled. There are some issues that were identified during the development, integration, and operational support of this system which are also discussed. Another topic presented in this paper is the focus and pursuit of quality by the GSFC DAAC.

GSFC DAAC V0 Mission

The initial version of NASA's EOSDIS is Version 0 (V0). This system consists of eight DAACs disseminated across the United States. Each DAAC is generally specialized in Scientific disciplines. The DAAC role is to enhance and improve scientific research and productivity by consolidating access and distribution of Earth science data. The evolutionary approach of building a Version 0 system is intended to demonstrate the concept of an interoperable set of distributed archive centers and to prototype various aspects of the system prior to the first EOS satellite launch.

The Goddard DAAC has defined its mission "to maximize the investment benefit of the Mission to Planet Earth by providing data and services to enable the realization of the potential of global climate data by the science and education communities".

GSFC DAAC V0 Requirements

The GSFC DAAC is being developed in response to EOSDIS functional requirements as well as requirements generated from Science projects such as Sea-viewing Wide Field-of-view Sensor (SeaWiFS), Coastal Zone Color Scanner (CZCS), Total Ozone Mapping Spectrometer (TOMS), Advanced Very High Resolution Radiometer (AVHRR), Tiros Operational Vertical Sounder (TOVS), and Upper Atmospheric Research Satellite (UARS).

The GSFC DAAC has currently 731 GB of data archived (Table 1). This number is expected to increase to about 18 Terabytes by FY97 [1]. In 1995 the daily ingestion workload is estimated to be 26.4 GB/day (Table 2). All ingested data (except AVHRR) are compressed to reduce storage needs. This results in 18.9 GB/day of data being archived on the Metrum RSS-600 ATL (95%) and the Cygnet Jukebox (5%). The volume of data distributed is anticipated to be 40 GB/day of SeaWiFS data and 20 GB/day of non-SeaWiFS data, for a total of 60 GB/day. Two types of distribution orders have been identified: standing orders and random orders. The standing orders, by definition, are requests by users for some or all of the data as it is being received at the DAAC. The

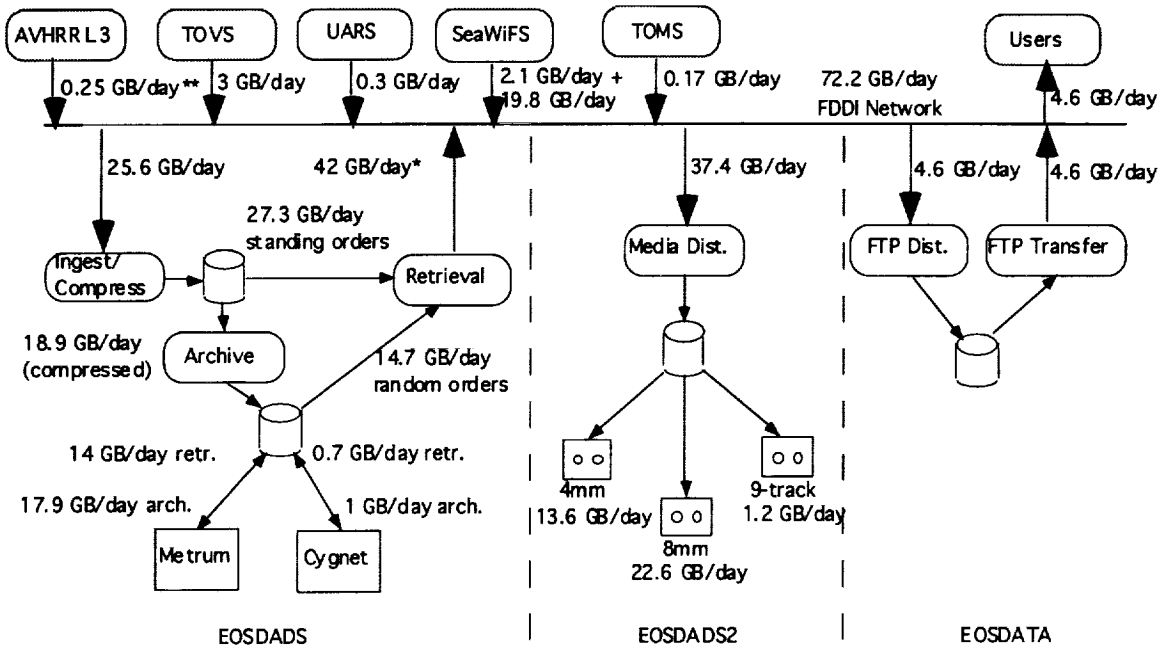
random orders are interactive requests by users for data that has been previously archived and is available for order. A significant proportion of orders are expected to be standing orders (65%) and most of the data ordered (89%) are assumed to be distributed on physical media (e.g., 8 mm) with the remaining being sent over the network (ftp orders). The distribution media supported currently at the GSFC DAAC are 8mm, 4mm, 9 track- 6250 bpi. The estimated V0 DAAC workload is illustrated in Figure 1.

Product	Volume archived on Metrum (GB)	Volume archived on Cygnet (GB)	Total volume archived (GB)
SeaWiFS L1 A (test)		1	1
SeaWiFS L2 (test)	1		1
AVHRR L3	111		111
UARS L3	35		35
TOMS	97		97
CZCS Level 1		345	345
4D assimilation	141		141
Total	385	346	731

Table 1 Total Volume of Data Archived as of 10-31-94

Product	Volume before compression (GB)	compression ratio	Volume after compression (GB)
SeaWiFS (regular)	2.10	0.72	1.51
SeaWiFS (reprocessing)	19.80	0.72	14.26
AVHRR	1.00	0.25	0.25
TOVS	3.00	0.80	2.40
UARS	0.30	1.00	0.30
TOMS	0.17	1.00	0.17
Total	26.37		18.89

Table 2 Estimated 1995 Daily Ingestion Workload



* Data compressed (equivalent to 60 GB if uncompressed)

** Data compressed by AVHRR Pathfinder PGS (equivalent to 1 GB if uncompressed)

Figure 1 Estimated DAAC Workload (Volume/day)

GSFC DAAC V0 Hardware Architecture

GSFC DAAC consists of three components, a Product Generation System (PGS), an Information Management System (IMS), and a Data Archive and Distribution System (DADS). The PGS receives low level data products (raw data requiring processing) and generates higher level data products. The IMS serves as a catalog of the data holdings which can be searched and browsed by researchers to help them identify and order data of interest. All data are archived within the DADS where they are available for on-line retrieval to fill researchers' orders for data.

A strategy was initially developed [1] to identify the best cost effective hardware and software configuration, and to measure the performance of the selected system [2]. Based upon the latest requirements, and projected workloads, the GSFC DAAC Fiscal year 1995 hardware configuration for the IMS and DADS is illustrated in Figure 2. The following are the points of the strategy.

- An SGI 4D/440 S (DADS) runs Unitree and the DADS software. To reduce the load, the DADS software is planned to be moved to a SGI Challenge L. The Unitree cache has 40 GB of disk space.
- Near-line data are archived on either a Cygnet 1803 jukebox (1179 MB) with 2 ATG WORM optical drives or an RSS-600 Metrum Automated Tape Library (ATL) (8700 MB) with 4 RSP 2150 VHS drives.
- A secondary archive is planned with a Challenge S (Backup) to keep a backup copy of all data ingested at the DAAC. The primary copy is archived by Unitree on an SGI 4D/440 S.
- The SGI Challenge L (DADS2) which has a larger number of I/O ports and fast internal bus, has all the distribution tape drives attached to it. The GSFC DAAC has nine 8 mm

drives, four 4 mm drives, and two 9 track drives. Additional drives may be added to satisfy future needs. To receive ingested data and copy data to tapes (e.g. 8mm) 40 GB and 72 GB respectively of disk space is available. Requests for FTP transfers are kept on-line on 40 GB of disks.

• An SGI 4D/440 VGX (DATA) computer runs the IMS software and Oracle. This machine has also the client which provides interoperability with other DAACs through a high-level Information Management System.

• The DAAC's distributed environment includes several ethernet Local Area Networks and an FDDI network connected to the EOSWAN.

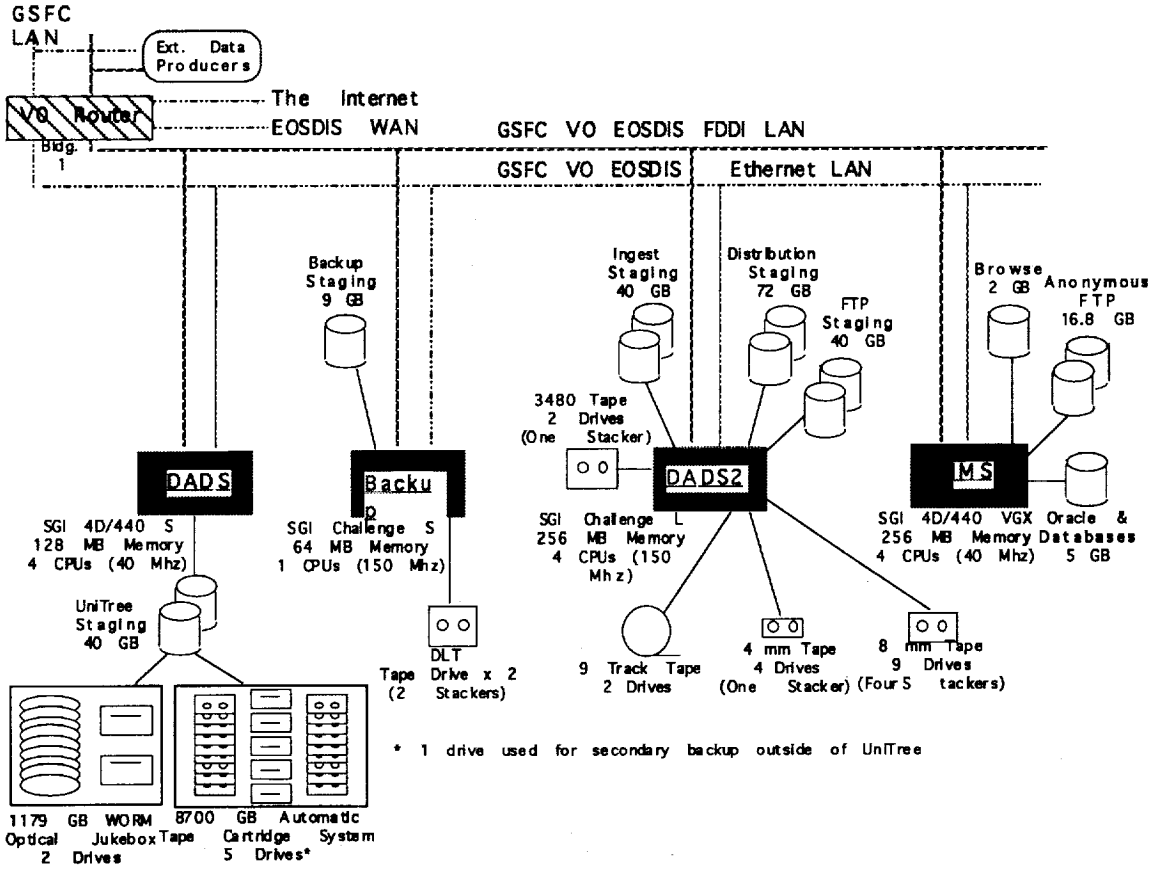


Figure 2 GSFC DAAC FY 95 Configuration

GSFC DADS Functional Design

This paper will now focus on the DADS and the mass storage issues. The GSFC DADS has three main functions: Ingest & Archive, Distribution, and Management. The ingest & archive function consists of accepting data products from outside the system, extracting or creating metadata, validating files, storing the files in the primary and backup archives, and updating the database. The distribution function retrieves files from archives, stages them to a distribution staging area, reformats the data if necessary (e.g. tar is the normal format for orders), and then writes the data to tapes or to an FTP staging disk. The DADS management software handles the scheduling, tracks DADS activities, and controls

allocation and deallocation of resources. The DADS functional design is illustrated in Figure 3.

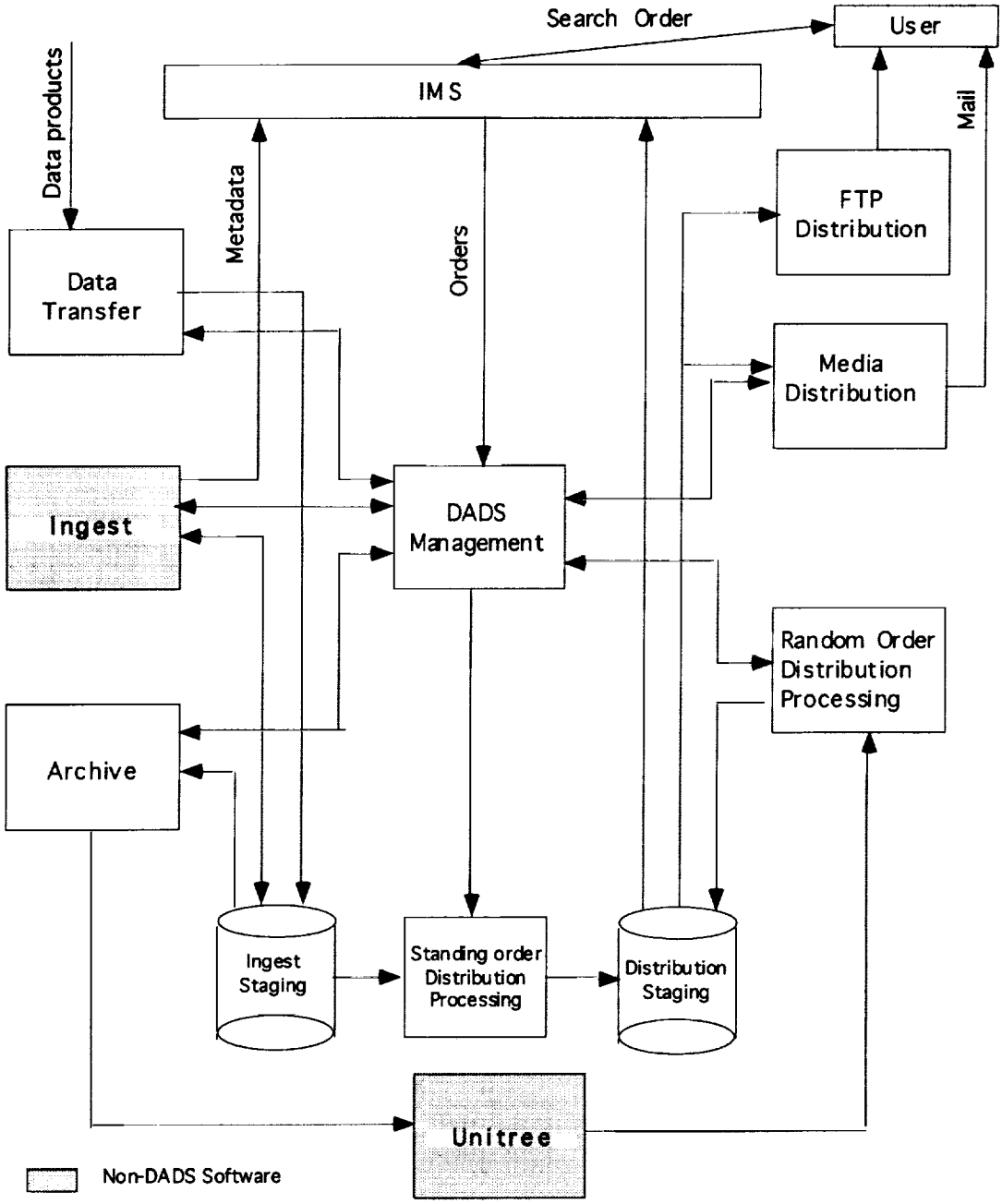


Figure 3 DADS Functional Design

GSFC DADS Ingestion, Archive, and Distribution Functions

The GSFC DADS currently ingests through network interfaces or directly from media datasets produced by the following scientific projects: AVHRR, TOVS, TOMS, 4 D Assimilation, CZCS, and UARS. The SeaWiFs project will be added to that list after the

launch of its satellite scheduled in Spring/Summer of 1995. Ingestion of data over the network is usually triggered when a scientific project invokes a client hosted on their computer called Data Transfer Program (DTP) (Figure 4).

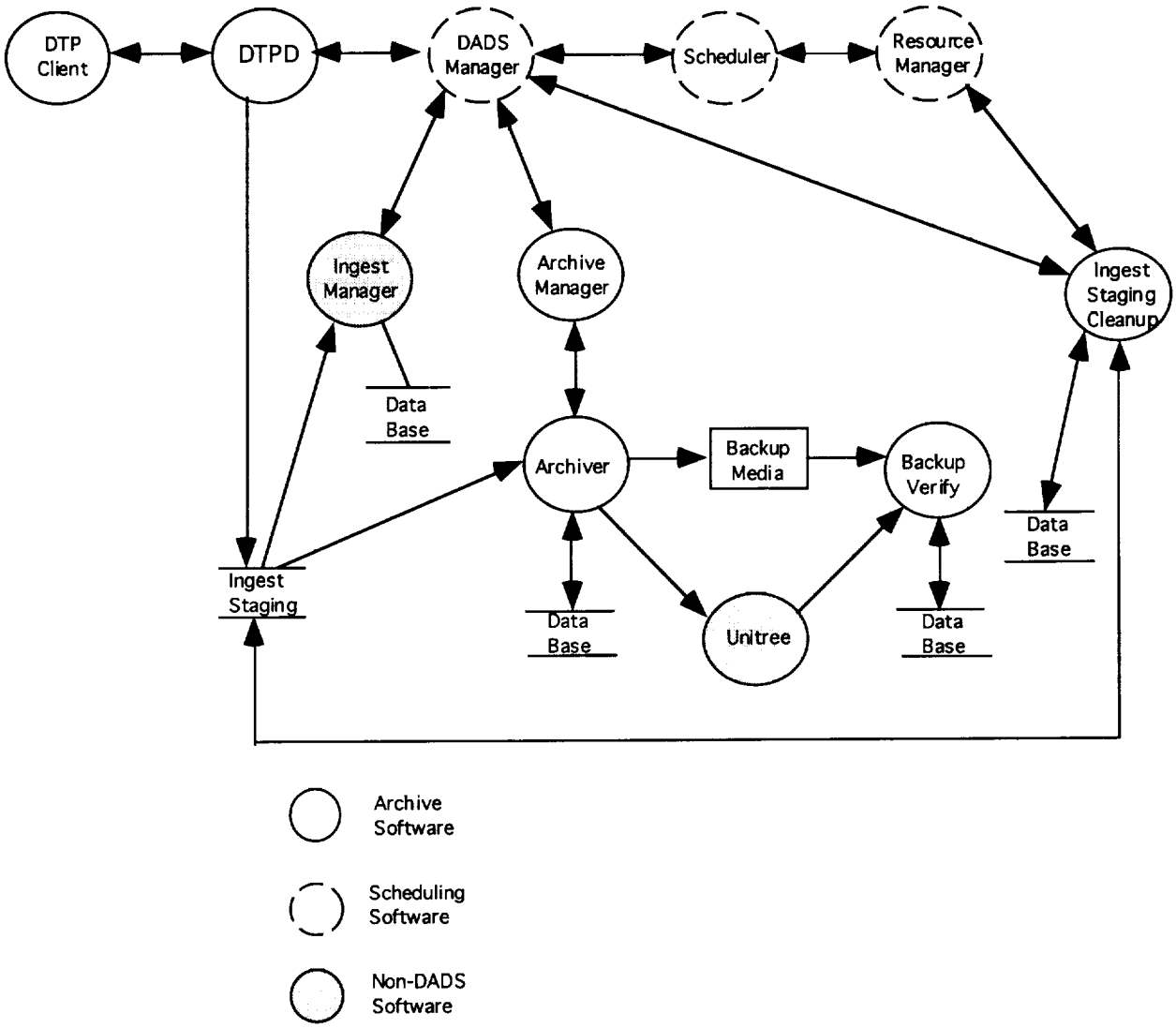


Figure 4 Archive Architecture

The transfer of data begins after DTP receives authorization from the DADS, which ensures the availability of resources to satisfy the ingest. The migration operations between the near-line devices (Cygnet jukebox and the Metrum ATL) are handled by the Hierarchical File Storage Management (HFSM) Unitree. The processing schedule and the resource allocation/deallocation are performed by the DADS modules: DADS manager, scheduler, and resource manager. A second archive copy is generated and handled by the archive manager, archiver, and backup verify. The ingestion and archive processes are described in detail in Table 3. In addition, Table 4 summarizes an Ingest/Archive walk-through.

Process	Description
DTP	Requests Ingest staging disk space from DADS Manager and Transfers files from the client system to the ingest staging area
DADS Manager	Sequences transfer, ingest, archive, verify, and staging cleanup
Scheduler	Interacts with the resource manager to allocate disk space, and Starts activities when resources are available
Resource Manager	Manages disk space in the ingest and distribution staging areas
Ingest Manager	Starts the correct processing script for each transferred file Script validates file, extracts metadata, and loads granule level database tables
Archive Manager	Batches archive requests Initiates archiving activities on a size or time basis
Archiver	Performs primary and backup archiving activities Computes and stores checksum values Exposes granules
Ingest Staging Cleanup	Checks successfully archived files against standing orders Copies files required by standing orders to distribution staging and adds items to open standing orders Removes successfully archived files from ingest staging area
Backup Verify	Run as chron job Retrieves backup archives files and recomputes checksum Compares checksum to value computed by archiver Sends E-mail to data producer on success

Table 3 DADS Ingest/Archive Processes

Step	Description
Transfer	1. DTP client and server establish connection 2. DTPD sends a request for disk space to Scheduler via DADS Manager 3. Scheduler, using Resource Manager, determines when to initiate the transfer and sends message to DTPD via DADS Manager to start transfer. 4. DTP Client and Server perform transfer 5. DTPD sends file completion message to the DADS Manager as each file completes transfer 6. DTPD sends termination message to the DADS Manager after all files are transferred
Ingest	1. DADS Manager sends ingest request to Ingest Manager for each transferred file 2. Ingest Manager starts appropriate processing script for each file 3. Ingest script extracts metadata, validates data, updates Data Base granule table, and sometimes does compression 4. Ingest Manager sends ingest complete message to DADS Manager for each file
Archive	1. DADS Manager sends archive request to Archive Manager for each transferred file. 2. Archive Manager adds file pending archive list 3. When archive list reaches a size threshold, the Archive Manager sends a batch archive message to the Scheduler via the DADS manager 4. The scheduler determines when to initiate the archiving activity and sends a message to the Archive Manager via the DADS Manager to start the Archiver 5. The Archiver copies the files to Unitree and to a backup tape, then sends an archive complete message to the Scheduler via the Archive Manager and the DADS.
Ingest Staging Cleanup	1. The DADS Manager starts the Staging Cleanup process 2. Ingest Staging Cleanup determines which files need to be staged for standing order distribution 3. Ingest Staging Cleanup requests disk space from the Resource Manager. 4. If the distribution space is available, Ingest Staging Cleanup copies the files and notifies the Resource Manager of the space used. 5. Ingest Staging Cleanup then adds the requested items to the standing order request. 6. Ingest Staging Cleanup removes the successfully archived files from the ingest staging area, notifying the Resource Manager of space made available.
Backup Verification	Runs as a chron job periodically Retrieves backup archive files and verifies using checksum Sends E-mail Notification to data producer that archive was successful

Table 4 DADS Ingest/Archive Steps

Another major function of the DADS software is the distribution of archived data to users. New order requests are generated by the user using the IMS and are then automatically submitted by the IMS to the DADS. Requests that are initially delayed are obtained later by the DADS by scanning the database using a program called pollreq (see Figure 5). Any known request can also be submitted manually for processing using ureproc. The staging operations between the near-line devices (Cygnet jukebox and the Metrum ATL) are handled by the HFSM Unitree. The processing schedule and the resource allocation/deallocation are performed by the Scheduler, Resource Manager, and Tape Manager. The DADS modules developed for the distribution function are summarized in Table 5. To clarify the distribution process, a walk-through is described in Table 6.

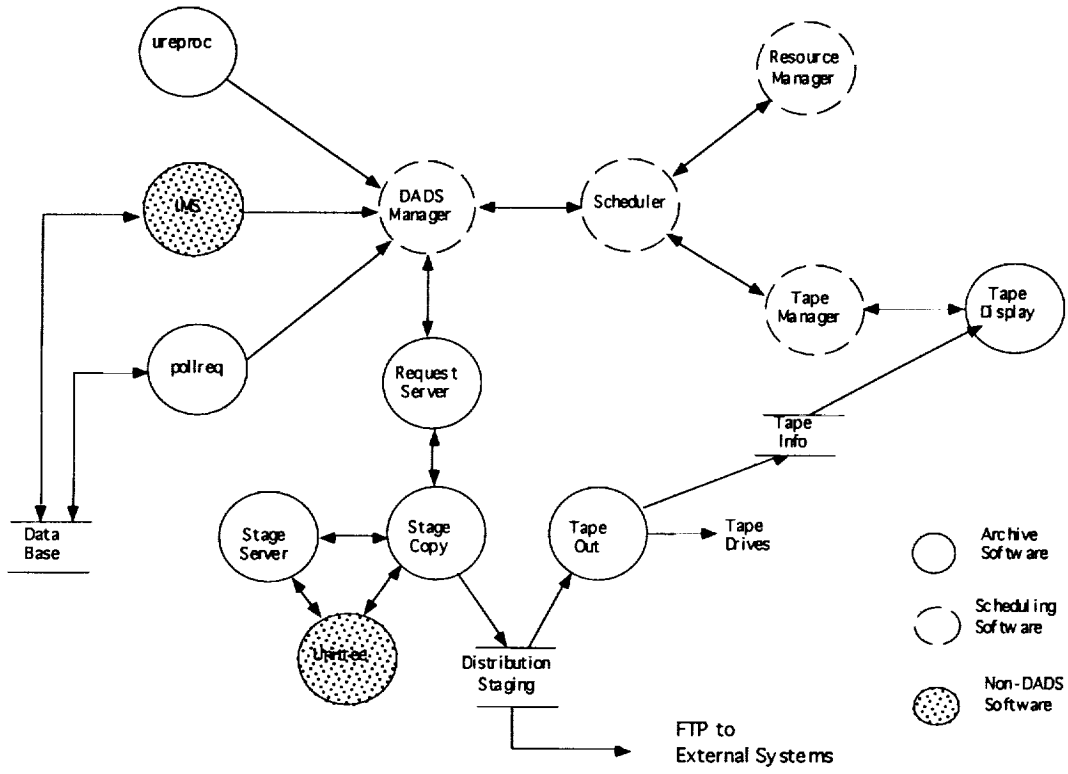


Figure 5 DADS Distribution Architecture

Process	Description
Request Poller (pollreq)	Scans data base for requests that have not been initiated Sends request ID to dadsmgr for each request found
DADS Manager (dadsmgr)	Sequences archive and distribution activities
Scheduler (schedsrvr)	Maintains queues of processing activities Interacts with resource & tape managers to allocate resources Starts activities when resources are available
Resource Manager (rsmansrvr)	Manages disk space in ingest and distribution staging areas
Tape Manager	Controls allocation and deallocation of tapes Controls automated (not manual) tape mounts for distribution
Tape Display	Show status of all tape drives Prompts operators to mount/dismount tapes
Request Sever (reqserver)	Locates all items in request and requests disk space Starts Stage Copy when disk resources are available Requests tapes required for request Starts Tape Out process when tapes are available
Stage Server	"Batches" Unitree staging requests
Stage Copy	Ask Unitree to stage files, and copies staged files to distribution staging area
Tape Out	Writes header and staged files to distribution tape

Table 5 DADS Distribution Processes

Step	Description
Staging	1. IMS or GenAutoOrder generates request in database 2. IMS or pollreq sends messages to DADS Manager to start processing request 3. DADS Manager sends request to Request Server 4. Request Server requests disk space from Resource Manager via DADS manager and Scheduler 5. Scheduler, using Resource Manager, determines when to process request 6. Scheduler sends message to Request Server via DADS Manager to start request processing 7. Request Server stages all files not already staged and creates symbolic links for all files 8. If no output tapes are required then Request Server signal completion of request
Tape Output	1. Request Server sends a message to Tape Manager via DADS Manager and Scheduler for tape drive 2. Scheduler determines when to write output tape, using Tape Manager (and Tape Display) to mount tape 3. Scheduler sends messages to Request Server via DADS Manager to write tape 4. Request Server creates child process to write tape header and files. Request Server signal completion of tape to Tape Manager via DADS Manager and Scheduler Tape Manager and Tape Display handle dismount of tape and bar-code label generation

Table 6 DADS Distribution Steps

Ingestion and Archive Functions

Files are ingested at the DAAC using DTP which incorporates a modified version of ftp. The regular ftp is not suited for background tasks and does not return error codes. The DAAC had to develop their own ftp that can be executed via a call routine and that returned error codes. The overhead associated with opening a connection and getting a response back via the DADS Manager turned out to be long (30 s). With small files (< 5 MB), the transfer time is much smaller than the opening connection time. It is therefore necessary to transfer a large number of small files with a single connection in order minimize overhead.

The DADS manager is a central point by which each message is received and sent. This design adds overhead and with a heavy load, this might become a bottleneck. Another alternative architecture would be to send messages directly to the recipient without passing through the DADS manager.

Scheduling the DADS activities efficiently is a difficult problem. The scheduler must dynamically schedule all the DAAC activities based on resource utilization and task priorities and some general policies. A resource can represent, for example, disk space, tape drives, or the number of concurrent ftp sessions. The scheduler must also prevent deadlock situations which would halt the system. In the first phase, the DAAC has developed its scheduler using a very simple scheme First In First Out (FIFO). This approach works fine when the resources are abundant. However, when there are contentions for resources, the schedule using a FIFO algorithm becomes extremely inefficient and slow. The granularity of the task is very important. Treating each process as a task is not a good solution because of the large number of processes involved. On the other hand, a task such as a distribution function has several sub-tasks that are to be scheduled separately while maintaining the order in which each subtask should be submitted. For example, a distribution function is composed of at least of a stage operation, a copy to the distribution area, and a copy to tape. It would be inefficient to allocate all resources needed at the beginning of the task. For instance a distribution tape drive should not be allocated until the data is staged to the distribution staging area. By dividing a task in a series of sub-tasks and by scheduling each individual subtask, the system resources can be better used and the overall performance can be improved. Each sub-task must allocate its own resources and the predecessors and successors of each sub-task must also be preserved. A general-purpose constraint-based scheduling engine based on the Time Map Manager (TMM) that uses a multi-level of tasks/subtasks is being studied for integration in the DADS software.

The DADS software was based on a client/server configuration. In the current architecture, each main function is a server that can be distributed over several platforms. The implementation of a client/server configuration turned out to be more complicated than expected. It is critical in this kind of environment to capture all errors and provide a mechanism to recover from these errors. It is also imperative to ensure that no single message is lost and that the communication protocol is very reliable. In the early stage of the development of the DADS software, messages were lost and processes were hanging. This could lock valuable resources indefinitely. One of the key problems with a client/server configuration is that when a server crashes, it takes many jobs along with it. A one process/one job philosophy would be better. Testing client/server software can also be a very difficult task because it is not always easy to reproduce errors that had occurred previously. With a client/server architecture, it is also important to limit the traffic of messages in order to achieve a good performance of the system.

Backup system

All V0 data are archived on several copies. The primary copy is on near-line storage (WORM platters or VHS tapes) using the HFSM Unitree. This implies that the data are stored with the Unitree Proprietary format. Relying on a single copy is prone for disaster sooner or later. During the first year of being operational the GSFC DAAC experienced unrecoverable errors on VHS tapes on six occasions, even though the life expectancy of the media was 10 years. Most of the problems were linked with a bad tape drive. In conjunction, the firmware of the Metrum drives used at that time did not limit the number of retries in search mode, and the media was damaged by an excessive number of passes. Unitree does not currently provide a mechanism to detect the number of soft errors or even the number of times a given tape is mounted/dismounted. With large archives it is imperative to detect such soft errors in order to predict when it is time to make another copy before the media is permanently damaged. The cost of creating a duplicate copy of a tape that has unrecoverable I/O errors can be a very expensive and time consuming task. Some data sets are in high demand and are used extensively. For instance one tape was mounted more than 2000 times in one year. With each mount, there are several passes and this exceeded the maximum number of passes (3000-6000 for the VHS tapes) provided by the manufacturers. Whenever possible, it is recommended to keep these highly requested datasets on magnetic disks or optical media, not only to minimize the response time but also to prevent such media degradation. It is not always easy to predict which datasets are going to be in high demand and the use of media such as VHS tapes must be closely monitored for high usage of individual tapes and a procedure put in place to copy these tapes to new tapes as needed.

Currently, the second copy of the data in the archive is done using the standard tar format on a VHS tape. This should facilitate the migration of the data to the EOS V1 system. A new backup system is under development. The plan is to copy all data by families (data set and level) on a VHS tape and on a DLT tape. The DLT media seems promising. It has a higher level of passes, stores a large volume of data and is relatively inexpensive. However DLT is a new media and because of its low cost, the project decided to make backup copies on both VHS and DLT until more is known about DLT drive and media reliability. On several occasions Unitree was unavailable for several days and the operations came to halt. The GSFC DAAC workload is going to increase several times with the SeaWiFs data sets, and another occurrence of Unitree unavailability for a long time would create difficulty in recovering from such long outage. To alleviate this problem, the DAAC has a contingency plan to use the backup system as an ingestion and distribution system. The backup is on a different machine, has its own drives and robotics, and is being designed to handle such eventuality.

Distribution Function

After conducting tests with a heavy workload, it became clear that the number of new distribution requests to process concurrently had to be limited (around 10). Several factors contributed to this condition. First, with a large number of files to stage, each stage command uses 3 processes, the maximum number of processes (500) available on the DADS could be exceeded in some cases. Secondly, the data had to be staged to a distribution staging area and too many concurrent nfs copies to disks resulted in severe degradation of the nfs throughput which is notoriously slow to begin with. Some factors contributing to the nfs poor performance were due to a maximum of eight group ids that can be sent and an nfs feature that locks directories until the files are opened. Replacing nfs by FTP should improve the throughput by 2 or 3 times. As with nfs, the number of

concurrent FTPs must be limited in order to achieve a good performance and scheduling becomes important.

Whenever a file is requested for distribution an Oracle database is searched to determine if it resides on the distribution staging area and to identify its physical location on the staging area. The access to this database was causing substantial delays (minutes) and the SQL code had to be optimized in the DADS software to achieve better performance. During the latest tests, the SGI 4D/440 VGX computer hosting the database was CPU bound and the DAAC is investigating the prospect of acquiring a more powerful machine as well as more optimal ways of accessing the databases.

The Stage server role is to group files belonging to the same family so that they can be submitted to Unitree as a single batch. This improves the overall performance of the system by minimizing the number of mounts/dismounts. The files selected that reside on the same tape are read with a single mount. Unitree philosophy is to have full data transparency and the users should not be aware of the physical location of the files. This concept may be fine with users but is completely inappropriate for system administrators, developers, and testers. If the physical location were known the stage server could group requests with files residing on the same media and schedule the stage from various orders to optimize the retrieval throughput.

An important parameter in designing the architecture of the system is the volume of data to be ingested and distributed. However it is also necessary to have good estimate on the size of the files. A system with many small files has more overhead than a system of the same size composed of larger files. With small files, more time can be spent searching the files on tapes than actually reading data from tapes. The size of the orders must also be well estimated in advance. Files belonging to the same orders are usually staged to distribution staging area prior to being copied to media or made available for ftp transfer. If the size of the orders are underestimated the distribution staging area may be too small creating delay and confusion at the operation level.

Orders are placed to the GSFC DAAC via the IMS. Data can be requested to be available over the network (ftp request) or distributed on media such as 4 mm, 8 mm, or 9 track (media request). With an ftp request, the data are automatically staged to disks to be copied immediately and the user is notified by E-mail. The User has 3 days to transfer the file(s) over their computer. As the number of requests increases the space needed to stage ftp may become so large that the 3 days policy may be cut to just a few hours and may not be long enough for the users. Sending data to users has other problems such as security, privileges and availability of user disk space.

Operation

One important role of the GSFC DAAC is the dissemination of the data requested to the scientific community. With respect to SeaWiFS only, 40 GB are expected to be distributed each day. To process this volume of data most functions have been automated by the DADS software. However, in this environment it is not unusual for something unexpected to occur (e.g. bad tape) and the operators must identify, and rectify these problems manually. This can be time consuming and one lesson learned was that operators needed more tools to be more productive. These tools are also used to monitor the system, its

resources and the requests. The tools must be defined by the operators and developed by programmers. There is a tendency for developers to design software without fully understanding the need or operation concept. This can result in a product that is too complicated to use, too cumbersome, or does not meet the needs. Tools were part of the preliminary designed but the scope of the task may have been underestimated. Some of these tools are also difficult to identify until you have a real system in place. Without these tools the overall productivity can be greatly reduced.

Another major challenge in building a system such as the GSFC V0 DAAC, is to design it from the beginning with operability, condition monitoring, error recovery, and performance. These aspects are often neglected as a project starts with some type of prototyping where the emphasis is on functionality.

Creating the data requested by the users is not the only task. Tapes must be labeled, tape contents verified, documentation must accompany the order, and everything has to be boxed and mailed. All these steps can be manual intensive, time consuming, and must be streamlined in order to be as efficient as possible. Without the right procedures and tools, operators can spend a lot of time performing these tasks. This could result in a degradation in quality as less time is spent monitoring the system for unusual events. To minimize the risk of inadvertently switching tapes for different orders, all tapes are labeled with bar codes and scanned by bar code readers. Mailing labels are printed with identical bar codes to insure that the correct tape is sent to the researcher.

Not all the requests are entered electronically via the IMS. Some users still need to order datasets over the phone or need assistance. To support the users, the DAAC has a User Support Office (USO). The interaction between USO and the operation group is important. Lack of communication between these two groups or any other groups within the DAAC would results in deterioration of the service provided to the Scientific community. In addition information that are often needed by the researcher (e.g. status of order) should be available on-line to minimize the workload of the USO staff.

The GSFC DAAC is a service oriented organization and as such has the responsibility to provide the best product to users. To help to achieve this goal, a quality team has been created at the DAAC. Its primary role is to identify quality issues and to suggest solutions. A strong emphasis has been placed on quality issues that mostly impact external users. This group was established after discovering that blank/bad tapes had been sent to users. One of the first tasks of the quality team was to review complaints within the DAAC and by our customers. Then, starting from the operation level, the DAAC processes have been reevaluated to identify deficiencies and propose solutions. For example, to preclude GSFC DAAC from sending bad/blank tapes, a directory of the tape is compiled. This solution is time consuming because it takes the same amount of time to create the tape as to read it and generate a directory. Other alternatives are to read only the first records or get a directory of tapes randomly selected. Capturing I/O errors during the creation of the tape is another way of insuring the quality of the tapes. 8mm and 4 mm have a read/verify operations after a write operation that could guarantee the data is stored properly on the media. The problem is that the I/O errors are reported at the bus level only and when several drives are connected to the same bus it is not always possible to determine which drive had an I/O error. 8 mm stackers have also been purchased to minimize human intervention and reduce the risk of errors. As simple as these functions may be, examining the processes in details

has revealed that their implementation is usually too complex, inefficient and filled with unnecessary manual steps that slows down the performance.

Testing

The GSFC DAAC has conducted numerous tests on the V0 System to measure the throughput of its peripherals running separately or concurrently. Basic functions such as ingest, stage, ftp have also been benchmarked in order to estimate the overall performance of the DAAC and to identify bottlenecks and limiting factors. These measurements have been summarized in Figure 6. The numbers listed in Figure 6 represent the best values obtained on a system that was not busy. The distribution tape drives (4mm, 8mm, and 9 tracks) transfer rates varied with the size of the files copied. Writing a large number of files on tapes with the tar format was found to be faster than copying the same data on the same drive using dd command. Currently, the only mechanism to transfer data in and out of the Unitree cache is via nfs or ftp. The best throughput of a single file transferred was measured at 1570 KB/s with ftp and 430 KB/s with nfs in local host. The ftp and nfs throughput is a function of the number of concurrent transfers as illustrated in Figure 7 and Figure 8 . Having too many ftps or nfs running at the same time can reduce considerably the overall throughput. If the files reside on the same disk, there may also be some disk contention. Compression and decompression are CPU intensive operations that may create a bottleneck. As expected these operations are executed faster on the SGI Challenge L than on the SGI power series (see Figure. 9). Several compressions or decompressions running simultaneously will contend for the CPUs and potentially the disk I/Os resulting in degradation in the overall individual compression/decompression transfer rates. A hardware solution for compression/decompression would alleviate this problem. The GSFC DAAC has investigated for such a hardware board, but in vain. The stage operations have been tested using the RSS-600 Metrum Automated Tape Library (ATL). It is difficult to measure the throughput of these operations because they depend on the size of the files retrieved and the position of the files on the tapes. Using a large file (270 MB) positioned at the beginning, in the middle, and at the end of the tape it was found that the overall effective transfer rates that include all the overheads (pickup time, load time, time for Unitree to read header, search time and read time) was respectively 545 KB/s, 604 KB/s, and 612 KB/s. These rates are roughly one third the native rates of the Metrum drives. These tests were for a large file and reflect best case scenarios. The latest tests conducted during several hours with 3 Metrum drives show that with 30-200 MB files the transfer rate was around 170 KB/s per drive. Even with multiple drives (5), this can become a bottleneck and it is important to schedule these stage operations in order to minimize the number of mounts/dismounts and therefore maximize the overall throughput.

In addition to these individual tests, GSFC DAAC has conducted "mini-tests" each time a new version of the DAAC was released. The initial objective of these mini-tests was to demonstrate that the center could process 40 GB/day of SeaWiFS data. After conducting the first mini-test it became apparent that the goals of these mini-tests should be expanded. For instance, software bugs which could occur only when the system was under a heavy workload, were discovered. The mini-test was in itself an extension to thorough testing performed by an independent test team. These mini-tests also contributed to identify deficiencies in operation procedures. This resulted in increase productivity and improved the overall quality of the data ingested and distributed. The problem associated with these tests is that the operations are delayed while they are conducted. However the benefits outweigh the drawbacks.

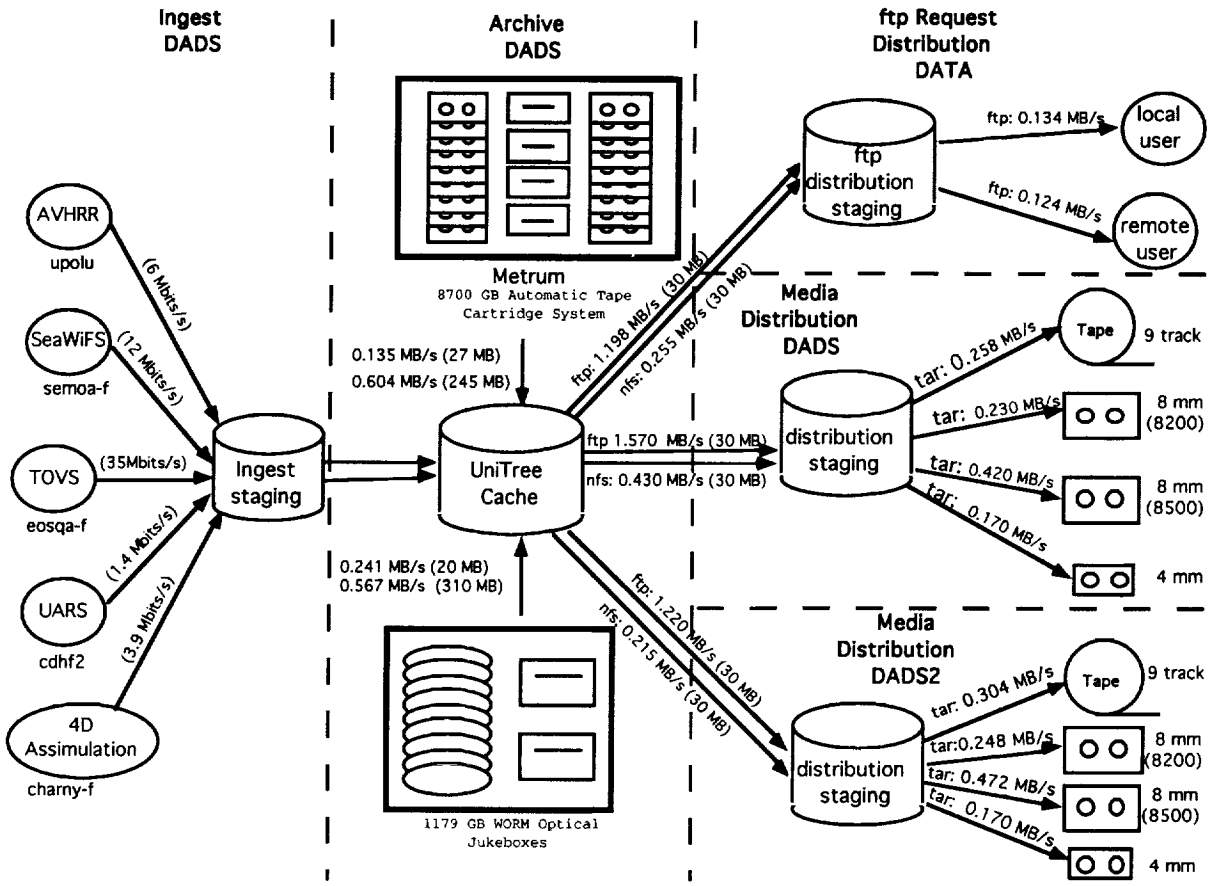


Figure 6 GSFC DAAC V0 Testing

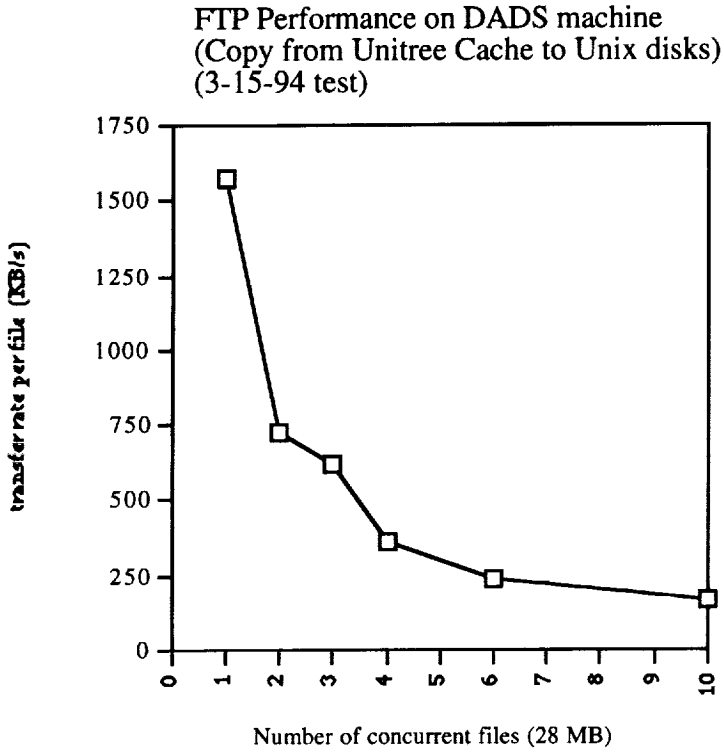


Figure 7 FTP Performance

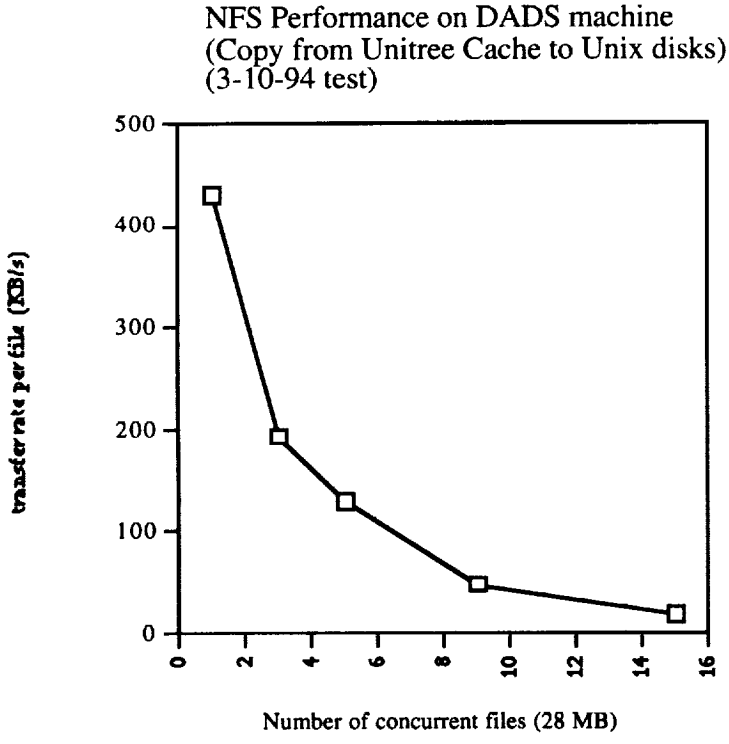


Figure 8 NFS Performance

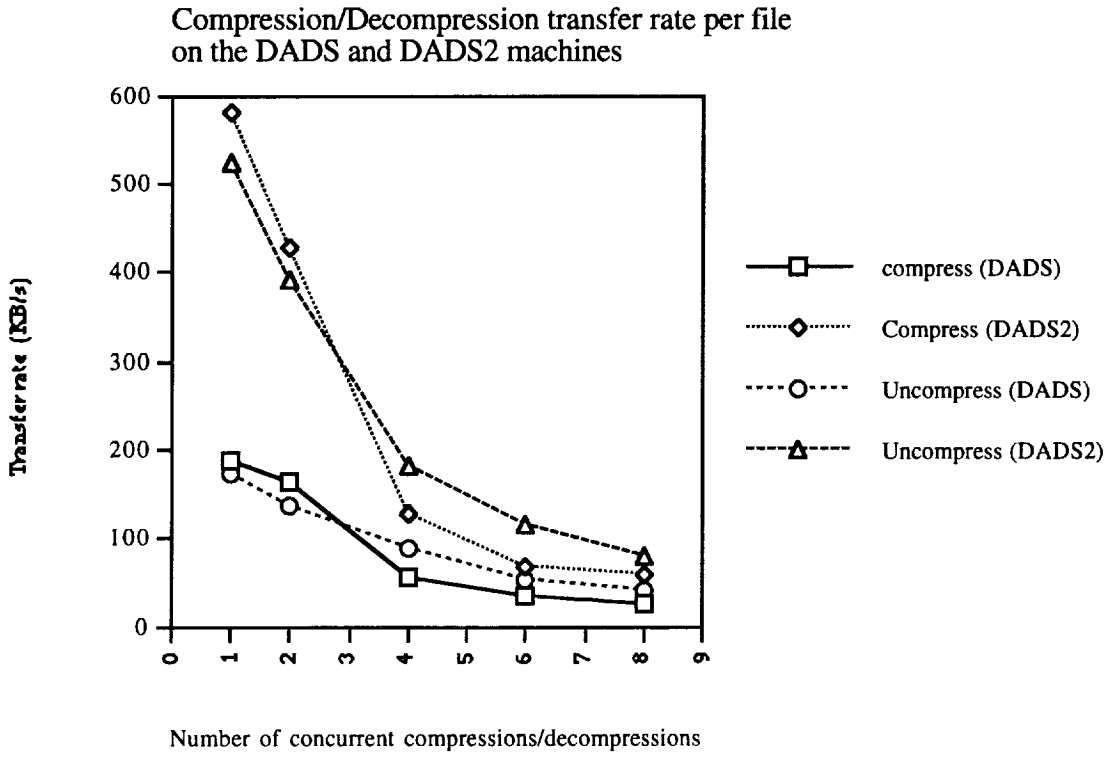


Figure 9 Compression and decompression performance

A 16 hour test was conducted on Tuesday, December 13, 1994. The primary objective of this test was to demonstrate that the GSFC DAAC could distribute 40 GB of SeaWiFS orders each day. No ingestion was processed during this test. The total number of orders and total volume of orders processed exceeded the target goal for both standing orders and random orders. During the test, the DADS software proved to be very robust. All the SeaWiFs test orders were completed more than 3 hours before the end of the test. During the test, all data copied from the Unitree cache to the distribution staging area, were transferred at the speed of the nfs because the disks were nfs mounted This is currently the main bottleneck in the system. However, preliminary tests have shown that by using ftp, the transfer rates between the Unitree cache and the distribution staging area should be 2 or 3 times higher

Hardware

GSFC DAAC bought hardware peripherals (disk & tape drives) at a discount price from third party vendors. The initial saving was not always a good investment as the DAAC system staff had to work very hard to integrate the peripherals. This distracted the system engineering from other urgent tasks, increasing system downtime, and generally caused grief to developers and operators. However, because staff time and system downtime do not get accounted directly we were able to procure significantly more disk capacity than otherwise. Another risk associated with purchasing peripherals from small third party vendors is that they are more prone to go out of business and with them go the warranty.

The GSFC DAAC experienced serious network throughput with its Science producers. After some investigation, it was discovered that older routers and bridges could not handle the load of Ethernet and FDDI, and had to be replaced.

Unitree

To automate the migration and staging operations between the robotic devices and the magnetic disks, the GSFC DAAC is using Unitree. At the time of the selection process, Unitree was the only product that fulfilled some of the requirements of the version 0 GSFC DAAC. The initial design of the DADS was to read files directly from Unitree cache and to copy them to the distribution media selected by users. On several occasions, files that were needed for distribution were purged from the disk cache by Unitree before they could be copied to tape. Another problem associated with Unitree is its poor performance in getting data in and out of its cache. The GSFC DAAC had to resort to developing and managing a second cache (i.e., various disk staging areas) to avoid the problems listed above. The duplicate cache increased the complexity of the DADS software and is expensive in terms of additional disk space needed. Having an Application Program Interface (API) would have been very useful in the development of the DADS software. Titan/Avalon recently delivered an API for Unitree but it was too late for the project to incorporate it and be ready for the SeaWiFs launch

One of the main drawbacks of Unitree is its lack of robustness. The GSFC DAAC has one person dedicated to monitoring Unitree at all times. This is not unusual as we discovered by talking to other Data Centers. This is a serious problem during the weekends as ingestion and distribution were disrupted because of problems related to Unitree and there is no one to monitor it. Unitree has come a long way, and its new version is more thoroughly tested and provides added functionality. However, it has not yet reached the maturity where it can run unattended, and it is still very expensive.

There are other issues that have been reported to the last Unitree users' group meeting held at GSFC on November 9-10, 1994. Most of them are related to inadequate documentation, cryptic error messages, lack of monitoring and administration tools, and no mechanism to capture soft errors detected by the drives during a read or write operation. This latter function is important as an increasing number of soft errors is an indication that the media might be degrading and that a new copy of a tape should be made. Because this function is not available, GSFC DAAC is currently monitoring the number of mounts/dismounts for each tape and copying tapes after a set number of mounts.

The overall performance of Unitree has been measured during the numerous tests that the DAAC conducted. In particular the stage operations were identified as a major bottleneck. The Metrum drives were benchmarked to read at 1.6 MB/s from UNIX. The same tests running with Unitree show a degradation of an individual Metrum drive to 1 MB/s in the best case scenario. When an ATG drive from the Cygnet jukebox was doing I/O at the same time as a Metrum drive the transfer rate of this later drive was reduced by at least half. All these tests were conducted with a system that had no activity.

Conclusion

The V0 System of the GSFC DAAC has gained valuable experience from building a few terabytes archive and distribution system and has demonstrated that it is capable of distributing 40 GB of data per day. Unitree needs to be more robust and easier to manage. The DADS software has turned out to be a real challenge. The difficulty being primarily in developing a reliable product that is fully automated with a good error recovery and with good performance. The operability, reliability, and performance aspects should all be major considerations in designing such a system. Special attention should be paid when buying hardware from third party vendor. It is usually cheaper, but the integration may be difficult and time consuming. Selecting the right media is very critical because of the high cost to migrate to another media. With larger and larger archives it is imperative to monitor media degradation and make new copies before unrecoverable I/O errors.

1. L. Bodden, P. Pease, JJ. Bedet, W. Rosen: Goddard Space Flight Center Version 0 Distributed Active Archive Center. In Third Conference on Mass Storage Systems and Technologies. NASA CP-3262, 1993, pp. 447-453.

2. JJ. Bedet, L. Bodden, A. Dwyer, PC. Hariharan, J. Berbert, B. Kobler, P. Pease: Simulation of a Data Archival and Distributed System at GSFC. In Third Conference on Mass Storage Systems and Technologies. NASA CP-3262, 1993, pp. 257-277.

The Growth of the UniTree Mass Storage System at the NASA Center for Computational Sciences: Some Lessons Learned

Adina Tarshish
Mass Storage and Scientific Computing Branch
NASA/Goddard Space Flight Center, Code 931
Greenbelt, MD 20771
k3art@dirac.gsfc.nasa.gov
phone: (301)286-6592
fax: (301)286-1634

Ellen Salmon
Mass Storage and Scientific Computing Branch
NASA/Goddard Space Flight Center, Code 931
Greenbelt, MD 20771
xrems@dirac.gsfc.nasa.gov
phone: (301)286-7705
fax:(301)286-1634

Abstract

In October 1992, the NASA Center for Computational Sciences made its Convex-based UniTree system generally available to users. The ensuing months saw growth in every area. Within 26 months, data under UniTree control grew from nil to over 12 terabytes, nearly all of it stored on robotically mounted tape. HiPPI/UltraNet was added to enhance connectivity, and later HiPPI/TCP was added as well. Disks and robotic tape silos were added to those already under UniTree's control, and 18-track tapes were upgraded to 36-track. The primary data source for UniTree, the facility's Cray Y-MP/4-128, first doubled its processing power and then was replaced altogether by a C98/6-256 with nearly two-and-a-half times the Y-MP's combined peak gigaflops. The Convex/UniTree software was upgraded from version 1.5 to 1.7.5, and then to 1.7.6. Finally, the server itself, a Convex C3240, was upgraded to a C3830 with a second I/O bay, doubling the C3240's memory and capacity for I/O.

This paper describes insights gained and reinforced with the burgeoning demands on the UniTree storage system and the significant increases in performance gained from the many upgrades.

Introduction of UniTree at the NASA Center for Computational Sciences

The NASA Center for Computational Sciences (NCCS) provides services to more than 1200 space and Earth science researchers with a range of needs including supercomputing and satellite data analysis. The UniTree file storage management system first arrived at the NCCS on July 6, 1992. As UniTree was to be the primary system for mass storage management, the existing Convex C220 was upgraded to a C3240 with four CPUs, 512 megabytes of memory, and 110 gigabytes of disk. Also included in this initial configuration were 2.4 terabytes of robotic storage provided by two StorageTek 4400 silos. Although UniTree supported both NFS and ftp as access methods,

access to UniTree was permitted only through ftp in order to meet the throughput demands of users of the NCCS's Cray Y/MP (UniTree's primary storage client), IBM ES9000, and workstation clients.

The mass storage contract under which Convex/UniTree was obtained required that it be able to handle 32 concurrent transfers while 132 other sessions supported users. The size of files transferred in acceptance tests was realistically large, about 200 megabytes each. The initial Convex UniTree system ultimately showed itself able to manage this workload, and by the third week in September it had passed acceptance.

In those first early months, the growth in UniTree usage was steady, but manageable. There was about 5 GB of new data being stored each day, about 10 GB a day total network traffic to and from UniTree. Ethernet access to UniTree was slow but generally reliable. As Convex UltraNet/HiPPI connectivity was not yet available, many users still preferred the block-mux channel speeds supported by the MVS Cray Station and continued to use the IBM/MVS legacy system to hold the bulk of their Cray-generated data.

In the course of the next two years we would observe repeated instances where UniTree usage would increase sharply and components of the software and supporting operating system services would fail under the heavy strain. We would note that upgrades to the NCCS's primary compute server would require corresponding upgrades to the mass storage system. We would become painfully aware of the relative immaturity of UNIX-based mass storage software in general and UniTree in specific when compared with other types of software in their availability of tools and ability to take advantage of high performance hardware. Nevertheless, contending with these obstacles, the NCCS's Convex/UniTree system has evolved to one of the most active worldwide, often transferring over 100 GB per day and over half a terabyte a week (Figure 3) while concurrently handling repacking tape activity to free over 150 400-MB tapes per day.

Effects of Compute Environment Upgrades on the UniTree System

With the arrival of UltraNet access for Convex/UniTree in January 1993, the UniTree usage curve took its first sharp upward turn. It was now routine for UniTree to receive 10 GB of new data each day, and for the total traffic to reach 20 GB a day. More and more Cray users began to use UniTree to store their data. In February 1993 the Cray Y-MP/4-128 was upgraded to double its previous CPU power (Figure 2), and the rate of new data stored in UniTree also doubled to 20 GB/day. By the end of the month more than 7500 silo tapes out of an available total of 10,000 had been written with UniTree data.

Upgrade I: UltraNet and Cray Y/MP

UniTree's growing popularity soon exposed a serious impending threat—we were running out of storage. The only production-level versions of UniTree that existed at that time did not allow for more than 10,000 tapes to be managed by the system, but the NCCS UniTree system had consumed three quarters that amount in its first five months of operation. At our prodding, in early March 1993 Convex developed and installed a modification to allow for up to 100,000 tapes, 18,000 of them for robotic storage and the rest for vaulting, or deep archive. A second modification allowing for 36,000 tapes in robotic storage was installed in mid-April. *Lesson: Find out hard-coded limits as early as possible; have them modified if necessary.*

UniTree vaulting and repacking remained a concern. Our version of Convex UniTree 1.5 included an executable to handle repacking, or removing the "holes" from tapes caused by deleted files, as well as vaulting, or the copying of little-used files onto free-standing tape for deep archive, but neither function worked properly at our site. It was apparent that the additional 8000 "robotic-controlled" tapes now defined by software as the top level in the storage hierarchy would not last for more than a couple of months; without repacking or vaulting, this newly added capacity would merely postpone the consumption of the entire top-level hierarchy. In addition, the two UniTree silos were nearly full: without vaulting, most of the additional 8000 tapes in the top level would not be mounted by robotics but by human operators. On active days, that would amount to hundreds of manual tape mounts a day to read and write users' most recent data. We did not have the operations staff necessary for such an undertaking, nor did we want to slow users' access to most recent files while humans located and mounted the tapes. For these reasons, the NCCS insisted on fully functional repacking and vaulting.

By April 5, 1993, we finally had a working tape repacker for UniTree 1.5. Immediately we began to repack in earnest, freeing hundreds of tapes for new data. By April 22,1993, we had also succeeded in vaulting to free-standing tapes. Working with Convex, we developed utilities that operators could invoke to write an internal UniTree label on new free-standing tapes, so that they could be used for vaulting. Operators were soon mounting vault tapes 24 hours a day, in an effort to keep the silos from filling. *Lesson: Include tests for repacking and vaulting along with tests for all other essential functions in initial acceptance testing.*

Upgrade II: Cray C98

At the end of August 1993 the Cray YMP was replaced by a Cray C98 with six CPUs. Network traffic to and from UniTree increased to 40 - 70 GB a day, 25 - 35 GB of which was new data. Due to inefficiencies in tape writing, UniTree 1.5 could handle no more than 24 GB of new data in the course of a day. As a result, by November 1994 we began to experience periods when the disk cache would fill and users were unable to store or retrieve any more data. A full disk cache also meant that vaulting and repacking would come to a halt, eventually causing the silos to fill. When UniTree ran out of eligible silo tapes for new data it would simply crash. Attempts were made to facilitate the writing of new data to tape, thereby slowing the filling of disk cache, by isolating the channel paths used for writing. Patches were installed optimizing the order in which files were migrated to tape to free disk cache space sooner. Despite these measures, UniTree had to be scheduled unavailable to users on six separate occasions (totaling 140 hours) for standalone migration and vaulting. The tape writing inefficiencies were not significantly improved until UniTree+ 1.7.5 was installed in late March 1994. *Lessons: In data-intensive environments with storage systems already near maximum load, resource plans to upgrade supercomputers must include provisions to upgrade the storage system if the supercomputer is to be used effectively. Include performance requirements in acceptance testing.*

UniTree Stresses Supporting Subsystems

Heavily used mass storage systems stress the supporting operating system services and hardware in ways unlike those of the traditional compute-intensive applications run on the high-powered machines now serving storage. In the NCCS's experience, networking and tape subsystems are particularly vulnerable. Limitations in these systems have sometimes affected UniTree's ability to write retrievable data.

UltraNet and HiPPI/TCP

Although it capably handled 90% of Cray-UniTree traffic when it was working well, UltraNet's history at the NCCS was troubled. Testing it after it first arrived, we discovered several serious bugs and had to wait for microcode fixes and software patches. (Initially the UltraNet native path was limited to 16 concurrent transfers; use of the host-stack path would crash the Convex; and the Convex would hang if UltraNet executables were used for Ethernet transfers.) While waiting for a patch to fix the latter problem, Ethernet access was disallowed on the port used by the UltraNet executables, and Ethernet transfers were given a separate port. After these initial bugs were fixed, a subtle timing problem between Cray and Convex UltraNet transfers intermittently afflicted transfers, sometimes affecting over a thousand connections a day. None of the vendors involved had experienced these failures between machines on their own floors. Concerted efforts by Cray and Convex staff resulted in an improved, but not cured, situation. *Lesson: A high-performance product that works well in the homogeneous environment on your vendor's floor won't necessarily work well in your heterogeneous shop.*

Under UniTree+ 1.7.5 we discovered that an abrupt abort of a single Cray UltraNet transfer would cause all other UniTree transfers to hang. Such an abort was regularly caused by a Cray user's deleting an NQS job that was actively transferring to UniTree. Attempts were made to have NQS job deletion and the "kill" command terminate processes less abruptly on the Cray, but with mixed results. Again Cray and Convex staff worked together to mitigate the problem, but their efforts were impeded by the difficulty in finding expertise from CNT/UltraNet. The problem was encountered during a period of financial uncertainty for the UltraNet corporation, before its acquisition by CNT, and many key UltraNet experts had left the company. *Lesson: Especially for relatively small markets and exotic architecture's, your vendor's company or critical staff may go away; encourage interoperating/dependent vendors to present alternatives.*

UltraNet interoperability problems were not limited to Cray/Convex transfers. The UltraNet hub adaptor repeatedly "autodowned" whenever transfers over a certain size were attempted from the IBM/MVS mainframe. This and related MVS/UltraNet problems were severe enough that the planned transfer via UltraNet of over 500 GB of data from the legacy MVS/HSM system to UniTree was instead detoured via the Cray. Block-mux Cray station transfers moved MVS data sets from IBM/MVS to Cray disk, then the legacy files were transferred via HiPPI to Convex/UniTree. While this was not the preferred use for the costly Cray disk, the duration of this workaround was limited and use of these C98 resources was favored over burdening an already saturated Ethernet with an additional 500 GB in transfers. UltraNet connections on the Convex and Cray were ultimately replaced with a HiPPI/TCP connection to an 8 x 8 HiPPI switch in September 1994. *Lesson: Significant systems problems sometimes require creative short-term contingency plans that use resources in unconventional ways.*

Initial experiences with a point-to-point HiPPI connection between Cray and Convex were also inauspicious. These initial problems were resolved after it was determined that the two vendors had been adhering to different parts of the standard. *Lesson: Despite acceptance of standards, interoperability between vendors cannot be taken for granted because the standards are subject to interpretation.*

Network Resource Allocation

Difficulties also arose when, to add a point-to-point HiPPI connection between the Cray and Convex, we upgraded the ConvexOS operating system from release 10.2 to 11.0. Aiming to maximize network performance, we increased certain UniTree networking parameters to values that had produced best results in testing at Convex, and noted promising performance during testing. Running with these parameters in production mode, we began to see numerous networking allocation failures, a phenomenon not observed during the HiPPI point-to-point stress testing. In addition, some users reported discovery of certain UniTree files that had been corrupted. We immediately reduced the networking parameters values to minimize the occurrence of the allocation failures. Convex staff identified the problem as a mishandling of the allocation failures and worked steadily on a patch to prevent the data corruption when these failures recurred. Evidence pointed to heavy Ethernet traffic as a primary factor in the allocation failures, as the slower Ethernet transfers tie up resources for a longer period of time than do HiPPI transfers. After painstaking analysis of the UniTree log files, the NCCS identified and published the list of all files at risk of having been corrupted by the problem. We installed and tested the ConvexOS patch as soon as it was available, and, although network allocations continue to fail under heavy Ethernet loads, the failures are now handled properly with no further data corruption. However, periods of these network allocation failures result in some user transfers failing, migration and repacking slowing to a crawl, and the annoying inability to use UNIX pipes and sockets. *Lesson: Stress tests aimed at pushing high-speed interfaces won't catch all systems problems; include stress tests with lower-performance interfaces in your test suites and add tests for new potentially concealed problems ("gotchas")as you find them.*

Tape Driver Travails

In February 1994, the discovery was made that a flawed Tape Library Interface (TLI) driver was causing thousands of consecutive tape marks to be imbedded within UniTree data files, making those files irretrievable by UniTree. Detection and resolution of the problem was belated because this behavior apparently occurred only with UniTree 1.5, and not with any other application. The workaround for the excessive-tape-mark problem was a Convex-written utility designed to wade through the reading of up to half a tape's worth of tape marks before reading data. Attempts to add this tolerance to UniTree's tape system failed because other UniTree processes still timed-out waiting for the files to be read. The suspect driver also caused some internal tape labels to be overwritten by tape marks after the tapes had been written with data. Some of these tapes were recoverable simply by re-labeling them (sans end-of-tape mark), but large blank areas following initial tapemarks on other tapes made the data beyond unreadable. With assistance from Convex, we copied and reconstructed these tapes manually. Installation of the patched tape driver, when it became available, ensured that no new tapes would be written with either of these problems. *Lesson: Mass storage applications may reveal system flaws not exposed by other testing; encourage vendors to include characteristics of mass storage systems under load in their system quality assurance test suites.*

Also troublesome were problems eventually attributed to the interaction between an older Convex TLI driver and our freestanding Memorex tape drives, which were used to write least recently used files to operator-mounted tapes. 7.5 percent of tapes written on the Memorex drives with this older version of the TLI driver were discovered to have one or more "null bytes" prepended to the beginning of data blocks. The additional imbedded bytes prevented UniTree's retrieving many files on tapes with this problem. The Convex-written utility that enabled the retrieval of files with embedded multiple tape marks included provisions to retrieve files with "null bytes" as well. This transparent handling of spurious prepended null bytes was successfully added to a customized-for-NCCS version of UniTree tape executables. While the exact cause of the extra null bytes has not

been pinpointed, evidence suggests that differences in interpretation of the FIPS-60 standard was a factor. A more serious problem with no known cause occurred on 199 of 27,000 Memorex-written tapes (i.e., fewer than 1 in 1000): entire blocks of data were missing. UniTree retries unsuccessful writes (on a new tape, if necessary); apparently the driver had not notified UniTree of some unsuccessful block writes. Affected files could not be recovered at all; if a driver problem had caused something extra to be written to UniTree tape, a method could be devised to reconstruct users' files. But there was no way to reconstruct missing data blocks that had no copy on disk.

The Tape daemon/ACSLS silo software saga

As the data under UniTree's control increased, so did the number of requests to retrieve data from UniTree tape. The Convex's tape daemon, used to allocate and deallocate tape drives, was frequently overwhelmed by the load, and communication timeouts and failures between it and the STK ACSLS silo-control software abounded. UniTree 1.5 aggravated the situation considerably by re-requesting the entire list of unsatisfied tape mounts every 2 minutes. There was some discussion about differences in packet addresses and versions being used by the two vendors, and engineers made numerous modifications to both ACSLS silo software and the tape daemon in an effort to mitigate this problem. In addition, the Sun server running the ACSLS silo software was also isolated on a private subnet to eliminate effects of extraneous network traffic on tape daemon/ACSLS communications. Ultimately we were forced to disallow the UniTree "stage" subcommand, which users had been using (and abusing) to request scores of tape mounts simultaneously.

The measures above have significantly reduced the frequency of severe tape daemon/ACSLS communications failures, but another intermittent tape daemon problem persists. Several times a week the tape daemon exhausts its available file descriptors and must be killed and restarted, causing loss of the state of current tape drive allocation and often requiring careful monitoring to restore normal tape allocations while ensuring minimal impact on UniTree. The problem's cause remains elusive after some investigation, and Convex has elected to use its resources to work on the ConvexTMR system which will replace the tape daemon instead of pursuing the file descriptor problem. Delivery of the TMR replacement has been delayed, resulting in some frustration at the prolonged exposure to tape daemon shortcomings—but also some solace in knowing these resources are being applied to resolve remaining TMR problems before its insertion into a production environment.

Science User-Driven Storage System Performance Requirements

In early summer 1993, the NCCS UniTree system was handling about 20 GB new data per day, with some effort. We anticipated delivery of a Cray C98 with more than twice the CPU power of the Cray Y/MP at the end of the summer. The NCCS's users and staff expressed concern about the ability of the UniTree system to handle the additional storage load from the C98. Convex asserted that with the right hardware and software configuration, the NCCS would be able to meet the users' requirements. Science users were canvassed to determine specific mass storage needs for the foreseeable future (in essence, until augmentation or replacement of the C98). Their responses formed the basis of our acceptance requirements (Table 1) for the upgrades proposed by Convex and the project integrator, FDC Technologies. Although performance requirements

appeared strenuous compared to production traffic in summer 1993, we have subsequently seen many instances where production usage approaches the peak loads artificially sustained during acceptance testing.

Reliability:
• The Convex/UniTree system must be available 95% of the total scheduled time as well as 95% of the prime shift • No data loss is acceptable • Performance and reliability requirements must be measurable within a normal production environment

Table 1a: Acceptance Requirements—Reliability

Performance (Phase 1):
• Store (put) and migrate 85 GB/day; retrieve (get) 300 GB/day from disk and tape, and free 85 GB/day through repacking and vaulting, all operations simultaneously occurring • Demonstrate 96 concurrent transfers of 32 MB each plus 64 "idle" sessions (doing a "dir" or "pwd") • Sustain an average aggregate transfer rate of 9.75 MB/sec • Demonstrate a migration rate of .98 MB/sec

Table 1b: Acceptance Requirements—Performance (Phase 1)

Performance (Phase 2):
• Store and migrate 100 GB/day, retrieve 300 GB/day, and free 100 GB/day through repacking and vaulting, all operations occurring simultaneously • Sustain an average aggregate transfer rate of 13 MB/sec • Demonstrate a migration rate of 1.32 MB/sec

Table 1c: Acceptance Requirements—Performance (Phase 2)

Acceptance Testing

The proposed configuration included a Convex C3800 series machine running Convex/Unitree+ 1.7.5. It became clear that peripheral hardware resources required for acceptance testing (UltraNet/HiPPI or HiPPI/TCP connections to the Cray C98, multiple robotic tape drives and controllers) were only available in the NCCS production environment. The NCCS user community was briefed on the need to make the UniTree production system unavailable during acceptance testing; although they preferred 24-hour/7-days-a-week access to UniTree, they recognized the sacrifice would result in longer-term benefits. Testing progressed more slowly than anticipated, complicated by the critically saturated UniTree 1.5 production system and problems discovered in then-Beta UniTree+ 1.7.5 software. Acceptance tests completed in early June, 1994, using production-released Convex/UniTree+ 1.7.6. Performance results are shown in Tables 2 through 5.

Test 1	1.5 Production Observed	Phase 1 Requirements	UniTree+ 1.7.6 Testing	Phase 2 Requirements
ftp "puts" (stores)	58.3 GB/day (0.691 MB/sec)	85.0 GB/day (1.007 MB/sec)	1.183 MB/sec	100 GB/day (1.185 MB/sec)
migration rate	36.0 GB/day (0.427 MB/sec)	85.0 GB/day (1.007 MB/sec)	1.016 MB/sec	100 GB/day (1.185 MB/sec)
ftp "gets" (retrieves)	34.1 GB/day (0.404 MB/sec)	300 GB/day (3.56 MB/sec)	11.558 MB/sec	300 GB/day (3.56 MB/sec)
vault./repack rate	20 GB/day (0.237 MB/sec)	85.0 GB/day (1.007 MB/sec)	1.0528 MB/sec	100 GB/day (1.185 MB/sec)

Table 2: Performance test #1

Test 2	1.5 Production Observed	Phase 1 Requirements	UniTree+ 1.7.6 Testing	Phase 2 Requirements
total ftp sessions	128	160	168	
ftp transfer sessions	32	96	100	none
"idle" ftp sessions	96	64	68	

Table 3: Performance test #2

Test 3	1.5 Production Observed	Phase 1 Requirements	UniTree+ 1.7.6 Testing	Phase 2 Requirements
aggregate network transfer rate	6.5 MB/sec	9.75 MB/sec (150% of observed 1.5 baseline; test system must include tape activity)	12.7417 MB/sec	13.0 MB/sec (200% of observed 1.5 baseline;test system must include tape activity)

Table 4: Performance test #3

Test 4	1.5 Production Observed	Phase 1 Requirements	UniTree+ 1.7.5 Testing	UniTree+ 1.7.6 Testing	Phase 2 Requirements
migration rate	0.658 MB/sec observed on a quiet system	0.98 MB/sec (150% of observed 1.5 baseline)	1.33 MB/sec	1.016 MB/sec	1.32 MB/sec (200% of observed 1.5 baseline)

Table 5: Performance test #4

Current Storage Hardware

The machine that completed acceptance was a 3-CPU Convex C3800 configured with 2 I/O bays. The C3830 has double the memory of the C3240 and more than twice the I/O bandwidth. The addition of the second I/O bay increased the maximum number of channel control units (CCUs) from 8 to 16; 12 CCUs are currently installed, including 2 enabling HiPPI/TCP connections to the Cray C98. Figure 1 shows this storage configuration.

UniTree disk cache has increased from the initial 50 GB to 155 GB for user data. We also obtained 40 GB of disk for RAID, after experiencing disk failures that caused repeated disk process crashes days later, during attempts to access a file with a fragment on the failed disk. *Lesson: RAID has successfully protected user files from disk hardware problems on a number of occasions, and has proven a valuable investment we consider to be worth the reduction in space available for user files.*

NCCS robotic storage has increased to 5 STK 4400 silos with 24 transports. Eight operator-mounted tape drives have been added for vaulting of least-recently-used files. 28 of these 32 transports have been upgraded from 18-track to 36-track. In addition, 22,000 cartridges of 3480 and 3490 tapes are being replaced by 3490E cartridges, which hold approximately 800 MB per tape. Movement of existing files to denser media is accomplished by creative use of repacking.

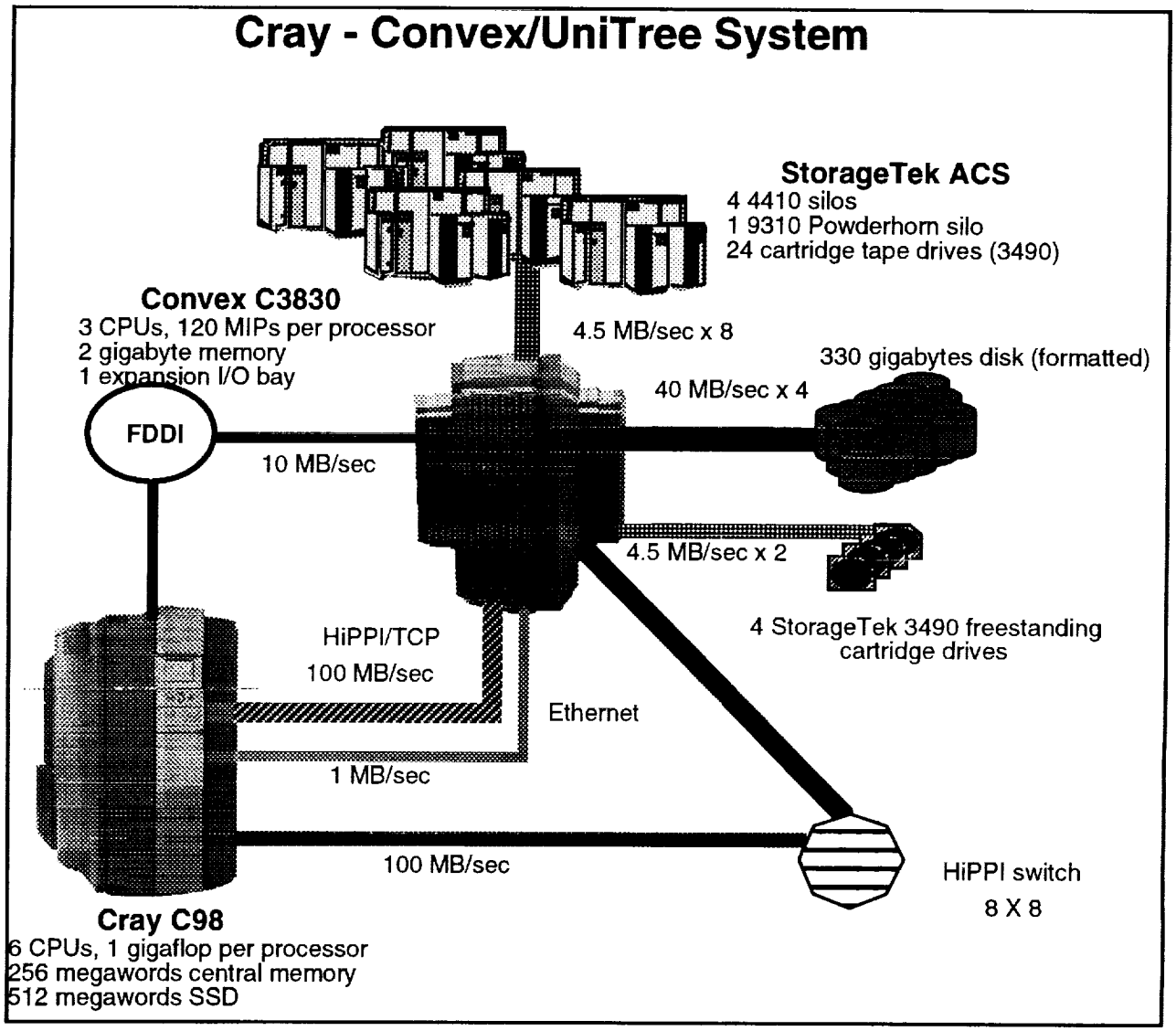


Figure 1: Cray - Convex/UniTree Configuration

Conclusion

The Convex UniTree system in production use at the NCCS today has seen significant improvements since its installation in 1992, and today meets or satisfies most of our expectations, and most of our users' current needs. From a system that could comfortably handle only 25 GB in transfers a day in early 1994, we now routinely handle over 100 GB/day with a high degree of user confidence. Robotic storage capacity has increased an order of magnitude, from 2.4 to 24 terabytes, with minimal down time due to problems. We are now beta-testing a release of UniTree with features that anticipate our future requirements, unlike 12 months prior when we anxiously

awaited a release that would meet our current needs. The process of reaching this current state, however, was not without considerable problems and frustrations. From experiences gathered during the last two years, three themes seem to dominate:

- *Users' input can be a valuable resource.* Their input on future requirements is essential for planning and justifying future acquisitions and for performance requirements in acceptance testing. Our users' feedback and cooperation during critical load times and acceptance testing was crucial to the evolution of performance and capacity improvements on our floor today.

- *Standards don't guarantee interoperability.* At least four problems cited above resulted from several vendors' different interpretations of standards. The standards/interoperability issue also applies to the mass storage software itself. UniTree was among the first UNIX-based mass storage systems to be ported and licensed on a wide variety of platforms. In light of delays on bugfixes and new releases from the previous UniTree originator, and demands for improvement from their customers, individual vendors have made significant modifications to UniTree. Some of these modifications affect a site's ability to move their UniTree tapes and databases to a different vendor's platform. Leveraging strength in numbers, the UniTree Users' Group has gotten vendors and the new originator of UniTree to agree to work together to resolve portability issues.

- *Stress testing:* Include high performance and low performance interfaces in stress testing, and add tests for "gotchas" to the suite as new problems are discovered. If it's possible in your environment, have vendors run acceptance testing with your equipment on your own floor, because it's virtually impossible for vendors to duplicate your environment. If practical, set up a test instance of your mass storage system and Beta/stress test new releases so that problems are detected and resolved before the product is installed on your production system.

The NCCS's science users project the need to transfer 2 terabytes a day by 1999. Up-and-coming high performance media, networks, and the like will achieve the rates required by our high-performance computing users, although the lag between the introduction of new hardware and the operating system and mass storage software's full utilization of its capabilities remains a concern. Our current beta test of Convex UniTree+ 2.0, which better exploits hardware via its enhanced tape resource configurability and multiple migration writes, should provide some insights on system behavior with higher-performance peripherals. But the increased sharing of data fostered by national and global information infrastructure efforts is already broadening the needs and the nature of the NCCS user community. Consequently, the NCCS is investigating interim methods to accommodate the "long haul," lower-speed needs of numerous remote users while sustaining high levels of service to local high-performance computers, although we anticipate researchers' and vendors' eventual development of more elegant means to handle these divergent needs. Current NCCS study involves creative use of UniTree families, tape types, and callout scripts to control the impact of many simultaneous remote sessions on high-demand needs of Cray processing. Our storage system progress to date, although not without its turbulence, induces great optimism about our future ability to meet the needs of both our lower-speed and high-performance science users, whose research activities drive one of the most active mass storage sites world-wide.

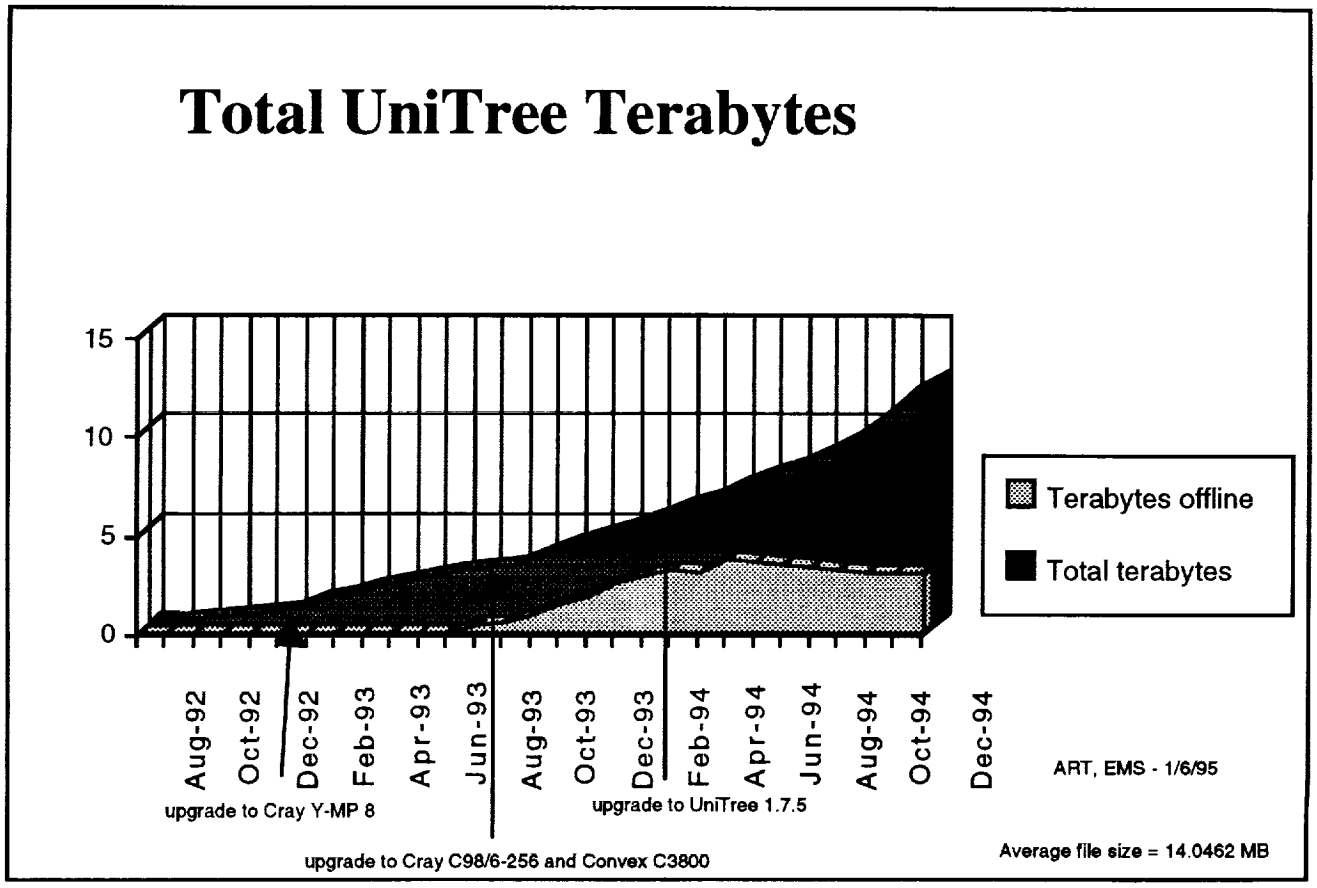


Figure 2: UniTree storage growth at the NCCS

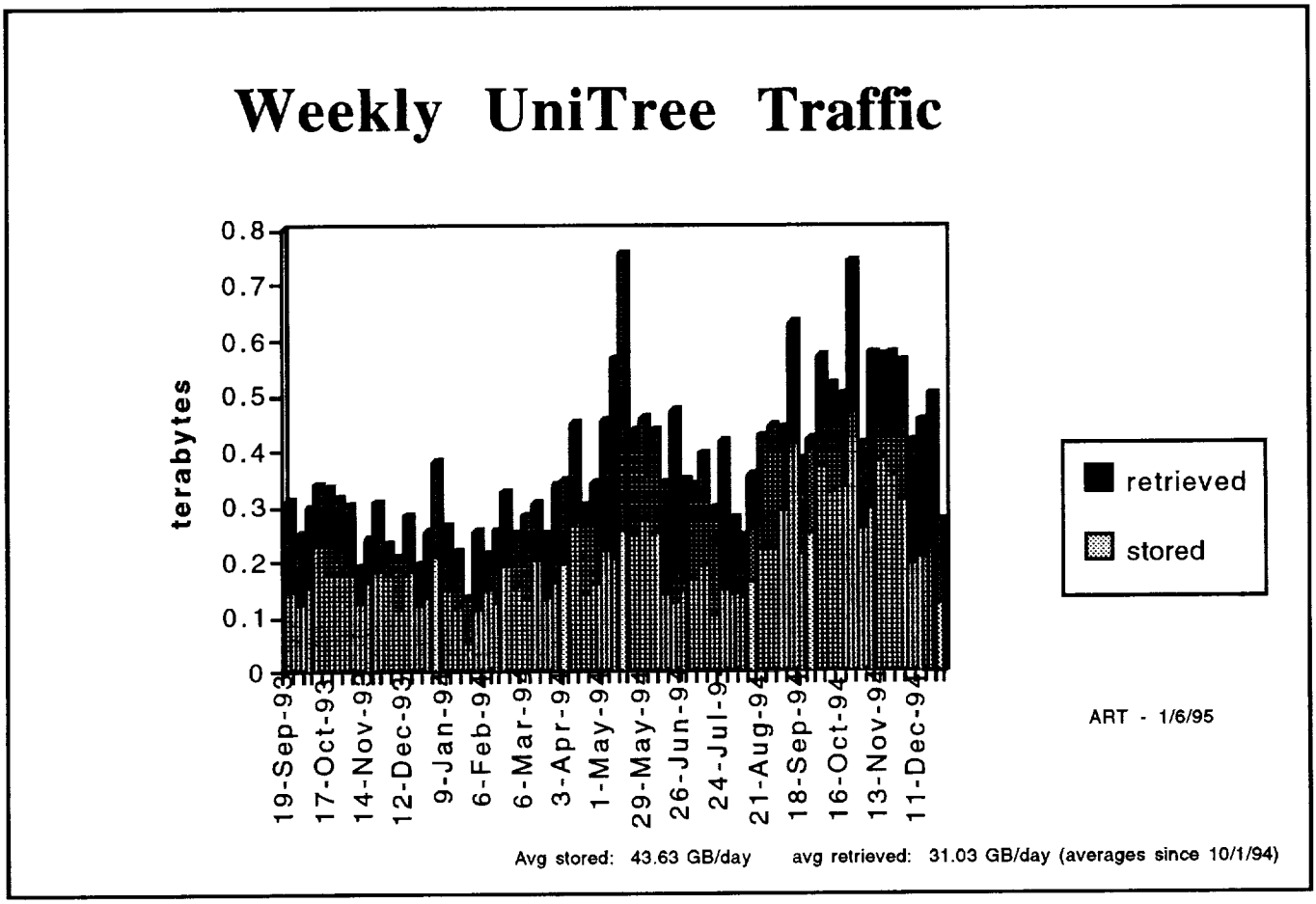


Figure 3: UniTree weekly network activity

NSSDC Provides Network Access to Key Data via NDADS

Jeanne Behnke and Joseph King
National Space Science Data Center
NASA/GSFC/Code 633
Greenbelt, Md 20771
behnke@nssdca.gsfc.nasa.gov
king@nssdca.gsfc.nasa.gov

Abstract:

The National Space Science Data Center (NSSDC) is making a growing fraction of its most customer-desirable data electronically accessible via both the local and wide area networks. NSSDC is witnessing a great increase in its data dissemination owing to this network accessibility. To provide its customers the best data accessibility, the NSSDC makes data available from a nearline, mass storage system, the NSSDC Data Archive and Dissemination Service (NDADS). The NDADS, the initial version was made available in January 1992, is a customized system of hardware and software that provides users access to the nearline data via ANONYMOUS FTP, an e-mail interface (ARMS), and a C-based software library. In January 1992, the NDADS registered 416 requests for 1,957 files. By December of 1994, NDADS had been populated with 800 gigabytes of electronically accessible data and had registered 1458 requests for 20,887 files.

In this report, we describe the NDADS system, both hardware and software. Later in the report, we discuss some of the lessons that were learned as a result of operating NDADS, particularly in the area of ingest and dissemination.

1. Introduction

The focal point of the NDADS is the mass storage components of two Cygnet jukeboxes, each configured with two SONY 6.5 gigabyte optical disk drives. The two jukeboxes provide the NSSDC 1.2 terabytes of nearline optical disk storage. A VAX cluster computer configuration drives the two jukeboxes, as well as providing network connections to the NASA science community including NSI-DECnet, Internet and US SprintNet. Although the numbers of data sets in the space physics and astrophysics areas are comparable, about 90% of the NDADS data, by byte count, are astrophysics data. These data include a mix of data currently arriving at NSSDC, plus selected data being promoted from NSSDC's offline archives to NDADS. To date, NSSDC has focused on loading space physics and astrophysics data to NDADS. Key space physics data sets presently available from NDADS come from the IMP-8, ISEE-3, DE-1 and 2, Hawkeye, Yohkoh, and Skylab missions. Key NDADS-accessible astrophysics data sets typically include the basic observation data files and accompanying ancillary files (calibration, etc.). The astrophysics missions with data in NDADS are IUE, ROSAT, IRAS, Ginga, VELA5B, HEAO-1 and 2, OAO-3 and the Astronomical Data Center Source Catalogs.

The NSSDC developed the NDADS to support the following requirements:

(1) the loading of data files to nearline storage and of associated metadata files to an inventory database;
(2) user access to the (relational) inventory database;
(3) user access to and retrieval of data;
(4) data security;

(5) user understanding of the system (through online user guides, etc.);
(6) aggregation of files according to individual project needs;
(7) capability to support additional types of mass storage devices as acquired.

Item 6 on file aggregation is a special concept, whereby related files are grouped into predefined "granules" or "entries." Users are thereby able to request, for example, an astrophysical observation by unique granule/entry ID, and have the system retrieve and stage all the relevant files without the user having to specify each one. This feature makes NDADS more than a typical "file server" system.

The NSSDC must meet several obligations as part of its mission as an archive. One of the primary obligations is that the data must be kept safe and secure. Data integrity is an important requirement as well. Of equal importance is our obligation to disseminate data from the archive. For its own sake, the NSSDC must determine ways to archive the data that are scalable and cost effective. It is important to emphasize that the NDADS is much more than a file server, and hence the reason for the development of the specialized software system, discussed in section 2. Functionally and operationally, NDADS can be divided into two NSSDC activities, ingest and disseminate. In sections 3 and 4, we discuss some of the characteristics and lessons of the ingest and disseminate functions.

2. NDADS Software System

NSSDC developed a specialized software system to manage storing and locating data on NDADS. The NSSDC Storage System (NSS) software was prototyped in mid-1991 and experienced a highly successful two year "experimental" public access period resulting in a second version of the software system completed in 1993. The NSSDC required a system that would support data stored on multiple platforms (UNIX-like) as well as the VAX/VMS™ system platform used in the initial system. The resulting NDADS must also support migrations from the current given hardware and software platforms and mass storage systems. The current NSS software is written for a VAX VMS™ 6.1 platform and uses two commercial-off-the-shelf software packages; the SYBASE relational database management system and CYGNET Jukebox Information Management System (JIMS). It also uses the Software for Optical Archiving and Retrieval (SOAR) for formatting the WORM optical platters, a package that was developed at NASA and available through COSMIC. The modular NSS software is written in C Language to provide us a measure of portability. A client/server approach was used in the development of NSS, allowing a client located on a system outside the NDADS facility to access the NSS server on the NDADS host. The NSSDC also requires a direct applications interface to the NDADS giving the staff better access and control over the system to increase data ingest throughput. The NSS direct applications interface is available through a command line interface and C Call routines.

An important feature of the NDADS is a high level of security and recovery applied to the storing and staging of data from storage devices. The core NSS software processes the data to be stored as part of the transaction management features of the SYBASE. The 'store' transaction is performed in a sequential, 'batch' mode, first storing the pointer to the data on the mass storage system in the database and then actually storing the data on the mass storage device. Since the data is 'stored' as a transaction, any failure that occurs during the store process will trigger the operation to exit and notify the ingest team. Data granules can be tagged as non-proprietary or proprietary, thus restricting access to certain individual user accounts. Proprietary data is that data which has not been granted access to the general public. A complex 'logging' mechanism has been created to track all NSS steps and are used to monitor problems and performance.

The modular design of the NSSDC storage system allows device specific modules ("fetchers") for new storage devices to be integrated into the system quickly and with minimal impact on the rest of the code. Each fetcher module is expected to provide a certain small set of critical services to the "master fetcher", such as mounting a volume, copying a file onto the device, copying a file out of the device, etc. The system is designed to enable the NSSDC to add additional storage devices transparently to the external users without modification of the base software system. Currently, the NSSDC has fetchers for the Cygnet-SONY WORM jukebox, online magnetic disk devices and there are plans to include several other mass storage devices. The NSSDC recently augmented the NDADS with a Digital Linear Tape jukebox connected to an SGI Indigo 2/IRIX workstation (1/95). Figure 1 shows a conceptual design of the NSS system.

3. Ingest Lessons

The NSSDC expects to receive and ingest close to a terabyte of data per year beginning in 1996. To meet ingest requirements, the NSSDC has been studying ways to improve ingest rates. The NDADS ingest process is influenced most by the fact that the nearline system has been WORM disk-based. This fact results in many idiosyncrasies that drive NSSDC processes, for example, the slow transfer rates of the disks, the permanence of the write operation, and the limitation of the number of drives. The ingest process is composed of more steps than was described in the section 2 as part of the NSS software system. Typically, the ingest steps are:

1) assemble the data and determine data staging requirements
2) verify the data (check headers, gross bounds checking,...)
3) archive the data to nearline devices using the NSS software

Ingest is differentiated at the NSSDC by whether the dataset is current and arriving directly (electronically) from a NASA project or if it has been a resident of the NSSDC offline archive.

3.1 Offline Data

If the data is already in the NSSDC, it is typically one of 80,000+ 9-track 'legacy' tapes in the archive. In most cases, the data must be converted to files before it is placed in NDADS. Although this step requires customized software, the NSSDC reuses many software modules for the data conversion elements. This step can be time consuming based on the number of errors that are encountered in the dataset conversion process. Data in the archive has several common characteristics:

- it is always an 'old' dataset, often with limited documentation
- a dataset is typically all on the same media and has a finite size
- responsibility for the dataset is completely the NSSDC's
- it is difficult to predict how popular the dataset will be for electronic dissemination
- requires a high degree of human interaction to move the data into the archive

In the case of offline data, the NSSDC uses techniques learned from previous data restoration tasks. It is important to:

1) peer review these legacy datasets before selecting them for placement in the NDADS
2) vigilantly maintain a schedule for transferring the data to NDADS
3) select datasets that have good documentation to support the dataset
4) pre-determine the amount of verification required for storing the datasets

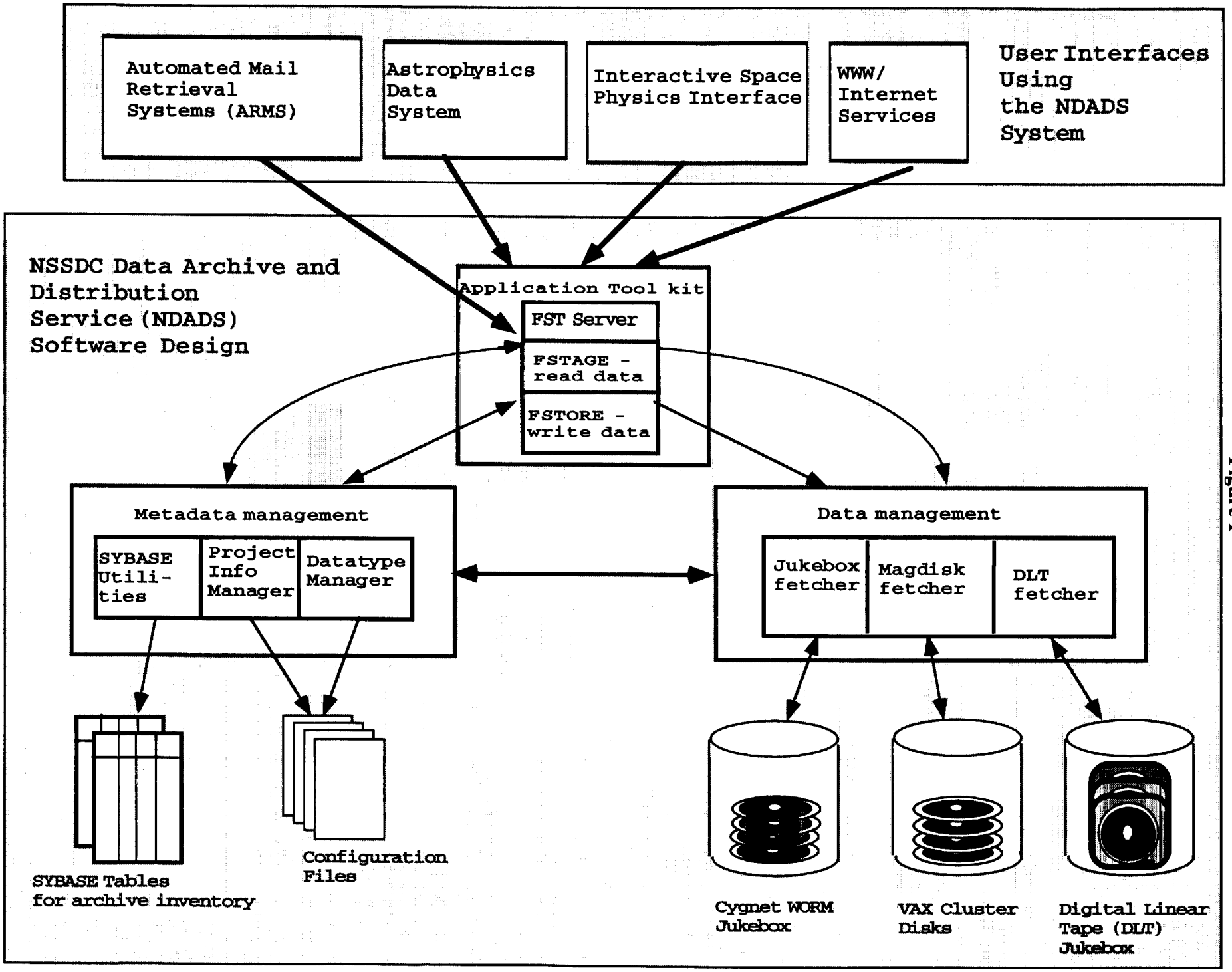

Automated Mail Retrieval Systems (ARMS)
Astrophysics Data System
Interactive Space Physics Interface
WWW/ Internet Services
User Interfaces Using the NDADS System
NSSDC Data Archive and Distribution Service (NDADS) Software Design
Application Tool kit
FST Server
FSTAGE - read data
FSTORE - write data
Metadata management
SYBASE Utili-ties
Project Info Manager
Datatype Manager
Data management
Jukebox fetcher
Magdisk fetcher
DLT fetcher
SYBASE Tables for archive inventory
Configuration Files
Cygnet WORM Jukebox
VAX Cluster Disks
Digital Linear Tape (DLT) Jukebox
362

Figure 1

5) pre-determine the amount of error correction required before the data is stored

The NSSDC must consider the 'setup' time associated with the above steps as well as the time spent preparing custom programs for reformatting tape data and data verification. We have discovered that a significant portion of manpower resources can be absorbed in these steps.

The NSSDC has reviewed different scenarios involved in ingesting different types of tape-based data to NDADS. Principally, 4mm, 8mm and 9 track offline tapes have been studied to determine the length of time involved in ingest. On the VAX cluster, our evaluation shows that 4mm and 8mm tapes are slower to physically ingest then the 9-track tapes. However, the set-up time for 9-track tapes is almost 4 times longer than that of 8mm and 4mm tape. The shorter set-up time is in partly due to the fact that the data on the 4mm and 8mm's is newer data and in some type of standardized format. The use of standards such as FITS, CDF, and SFDU simplifies the data verification phase as well as accelerates the step of converting to disk files.

3.2 Electronically Delivered Data

The NSSDC has been receiving newer datasets via the network. In these cases, the projects are still actively collecting data and transfer a processed dataset on a regular basis into NSSDC disks. If the dataset is delivered electronically, the NSSDC typically is only required to do basic checks of the data and then copy the data into the nearline system. Several characteristics make these datasets both easier for the NSSDC to work with and more difficult to control, for example:

- the NSSDC can review and affect formats of the data prior to their delivery
- both the NSSDC and the project share responsibility for the data
- easier to predict the popularity of a dataset and its eventual electronic retrieval
- software can be written to completely automate the ingest process, requiring little human intervention
- difficult to predict the quantity of data that will be delivered to the receiving/staging disk, thereby making it difficult to cost effectively determine the size of the disk

As part of the delivery function, the NSSDC contracts with each project a formal arrangement of delivering a list of what was transferred. These transfer lists are commonly referred to as Bills of Lading (or BOLs). In 1992, the NSSDC devised a BOL format that has served as a model for data delivered by other projects. The use of BOLs simplifies the NSSDC's the ability to cross check data delivered electronically by use of routine code. This permits us more accuracy and faster ingest into the nearline system.

The NSSDC does rudimentary verification and validation of the datasets before they are committed to the nearline system. Verification software is written in several programming languages, usually reusing existing code and often supplied by the data provider. The minimum set of tests is applied to newer datasets; i.e. check the filenames and header information, etc... The NSSDC will be working on ways to automate this aspect of data ingestion during this fiscal year. It is becoming increasingly clear that electronically delivered data must be spot checked rather than systematically checked given the large quantities received and the turnover rate from disk to nearline. It is difficult to find the CPU cycles to review all data received electronically.

The NSSDC staff has experimented with several different ways to schedule ingest and it remains are most difficult problem. Problems are routinely encountered in receiving

electronically delivered datasets, either due to system problems for both the project and the NDADS or due to network transfer delays. The data flow problem is compounded by difficulties in scheduling free staging disk space. Electronically delivered data tends to vary in size delivery-to-delivery. To alleviate these problems, it is important to get as much information on delivery plans from the project and to maintain close communications. The NDADS ingest staging space is planned to have available three times the maximum size of a delivery, this allows for potential hardware delays and unforeseen difficulties on NDADS. Along with scheduling of the ingest staging disks, we have in the past tried to manually map out the use of the optical drives to least impact the users who are retrieving data. This way we could insure that all of the drives in a single jukebox were not committed to ingest, thereby prohibiting access to the data for retrieval. This past year, we have developed selected batch queues controlled by the operating system to eliminate the manual intervention. This has improved our ingest throughput without affecting retrieval rates.

Many of the processes used to move the data through ingest pipeline are manually executed and monitored. An ingest team member will manually start one of the steps and monitor to completion. Following successful completion, another job is started and in some cases the jobs are performed in the batch queue. In our evaluation, manual pipeline processing nominally requires at least 4 hours per dataset. By eliminating manual pipeline processing for several electronically delivered datasets, we have increased the ingest throughput without affecting the quality of the load. The steps used for automated ingest of the data are often similar from project to project. The NSSDC is collecting these common steps into a generic ingest software system that can be customized with appropriate configuration files and used on any new dataset to be ingested into NDADS. Because of these measures, the NSSDC shows an increase in ingest rates in 1994, see Table 1.

1994 INGEST RATE IN GB											
JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC
9.6	10.2	3.5	18.2	4.4	15.3	21.3	35.2	11.5	17.2	32.4	85

TABLE 1.

4. Disseminate Lessons

The NSSDC is committed to providing its users, both in-house and outside community, four ways to access the NDADS archive:

1) via command line interface
2) via C callable routines
3) via FTP service
4) via an E-mail interface

The first two methods are used principally in-house to directly manipulate the nearline mass-storage systems for better management of ingest and disseminate functions on the NSSDC's behalf. Methods 3 and 4 are provided principally for the outside community. The NSSDC Automated Retrieval Mail System (ARMS) provides an E-mail interface to the NDADS archive. Users send an E-mail request to the account *archives@nssdca.gsfc.nasa.gov*. Within the message, users specify the need for information or data files by adhering to a fixed protocol for the content of the E-mail SUBJ line and, for data requests, by specifying granule ids in the body of the E-mail message. The ARMS Users Manual, detailing the protocols, may be obtained by specifying MANUAL as the subject of the message and leaving the message body blank. The E-mail system is very popular and has supported the distribution of over 260 GBytes of NDADS

data. In 1995, we will be working on providing a more fault-tolerant and modular ARMS system to our customer community.

The E-mail system has to its advantage a simplistic interface, but it also requires users to understand the NDADS granule-naming conventions and the granule-file hierarchy. Because of this requirement the NSSDC developed an FTP server to NDADS that makes the full NDADS archive appear to users as a massive FTP-accessible disk farm. The FTP interface allows the NSSDC more versatility in connecting to client/server based user interfaces. One advantage of the FTP service is that NDADS files now have Uniform Resource Locators (URLs). The FTP service incorporates well into World Wide Web pages developed at the NSSDC by space physics and astrophysics disciplines. These Web pages allow retrieval of NDADS data without specifically knowing granule names.

5. Conclusion

The NDADS has been developed to serve the specific needs of the NASA science community. It combines specialized hardware with customized software to significantly enhance the power of the NSSDC scientific database system. The success of this facility can be measured in several ways: the number of requests for data, the turnaround time, capacity, and convenience to the community. Available 24 hours a day every day, NDADS currently satisfies in excess of 1000 requests per month in an average of less than ten minutes. The NDADS service represents three-quarters of all NSSDC data requests. NSSDC believes its NDADS nearline data management environment is evolvable to exploit future changes in both hardware and software. By providing a well-constructed and secure infra-structure, NSSDC will be able to meet the future requirements of managing terabytes of data, cooperatively supporting NASA missions and supporting user interfaces that rapidly change to best meet the needs of scientists and others on the information superhighway.

In the future, the NSSDC expects to need additional storage devices to support the growing archive. The inclusion of the data and storage devices in use at the HEASARC, Compton-Gamma Ray Observatory and other related archives will be of primary importance to the NSSDC as well as intriguing in its possibilities of resource sharing across organizations. Careful planning and consideration will be required to phase-in the future computing requirements of the data center and not disrupt existing capabilities. The NSSDC will also consider improved access to the NSSDC data through Wide Area Information Service (WAIS), World Wide Web (WWW) and related network-based services as well as software application systems used in-house.

Analysis of the Request Patterns to the NSSDC On-line Archive[1]

Theodore Johnson
ted@cis.ufl.edu

Code 630
NASA Goddard Space Flight Center
Greenbelt MD 20771

Dept. of CIS
University of Florida
Gainesville, Fl 32611-2024

Abstract

NASA missions, both for earth science and for space science, collect huge amounts of data, and the rate at which data is being gathered is increasing. For example, the EOSDIS project is expected to collect petabytes per year. In addition, these archives are being made available to remote users over the Internet. The ability to manage the growth of the size and request activity of scientific archives depends on an understanding of the of the access patterns of scientific users.

The National Space Science Data Center (NSSDC) of NASA Goddard Space Flight Center has run their on-line mass storage archive of space data, the National Data Archive and Distribution Service (NDADS), since November 1991. A large world-wide space research community makes use of NSSDC, requesting more than 20,000 files per month. Since the initiation of their service, they have maintained log files which record all accesses the archive.

In this report, we present an analysis of the NDADS log files. We analyze the log files, and discuss several issues, including caching, reference patterns, clustering, and system loading.

1 Introduction

On-line scientific archives are an increasingly important tool for performing data-intensive research. Building a large-scale archive is an expensive proposition, however, and system resources need to be carefully managed. To date, there has been little published research that studies the performance of on-line scientific archives.

The National Space Science Data Center (NSSDC) of NASA Goddard Space Flight Center has run their on-line mass storage archive of space data, the National Data Archive and Distribution Service (NDADS), since November 1991. A large world-wide space research community makes use of NSSDC, requesting more than 350,000 files in 1994. Since the initiation of their service, they have maintained log files which record all accesses to the archive.

In this paper, we present an analysis of access patterns to the NDADS. These analyses are based on the information contained in the log files. We discuss several aspects of system performance, including the performance of several caching algorithms on the recorded request stream, and the effectiveness of the data clustering used by NDADS. We show that the request for a file are bursty, and that user requests are bursty. Finally, we present an analysis of the system load.

Several studies on the reference patterns to mass storage systems have been published. Smith [12] analyzes file migration patterns in hierarchical storage management system. This analysis was used to design several HSM caching algorithms [13]. Lawrie, Randal, and Burton [7] compare the performance of several file caching algorithms. Miller and Katz

[1]This work was performed while Theodore Johnson was an ASEE Summer Faculty Fellow at GSFC. This research is partially supported by grant from NASA through USRA, #5555-19

have made two studies on the I/O pattern of supercomputer applications. In [9], they find that much of the I/O activity in a supercomputer system is due to checkpointing, and thus is very bursty. They make the observation that much of the data that is written is never subsequently read, or is only read once. In [10], they analyze file migration activity. They find a bursty reference pattern, both in system load and in references to a file. Additional studies have been made by Jensen and Reed [5], Strange [14], Arnold and Nelson [1], Ewing and Peskin [3], Henderson and Poston [4], Tarshish and Salmon [15], and by Thanhardt and Harano [16]. However, all of these studies apply to supercomputer environments, which can be expected to have access patterns different from those of a scientific archive.

1.1 Log Files

The National Space Science Data Center is the primary archive for all space data collected by NASA. The NSSDC distributes its data using a variety of methods and media. For example, one can request photographs, CD-ROMs and tapes from the NSSDC. Manually filling orders for data is labor intensive and hence expensive. In addition, service is slow. To reduce data distribution costs and to improve service to the user community, the NSSDC created the National Data Archive and Distribution Service to store electronic images and data, and serve the data electronically.

The archive consists of a two jukeboxes storing WORM magneto-optic disks, one with a capacity of 334 GB, the other with a capacity of 858 GB. A user submits a request by naming a project, and the files of the project. Request submission is most often done by email, but can also be done using a program on the host computer, and through a new World Wide Web service. NDADS will fetch the requested files from nearline storage, place the requested files on magnetic disk, then notify the user that the files are available for transfer via ftp (alternatively, the files can be ftp'ed automatically). More information about NDADS can be found by sending an email message to archives@nssdc.gsfc.nasa.gov with a subject line of "help".

A user specifies the files of interest by naming them explicitly. In general, specifying files by predicate matching is not possible (although this capability is being developed).

NDADS is an evolving system, and log file collection is part of the evolution. Version 1 logs were recorded between November, 1991 and December, 1993. These logs record the files requested, the start and stop times of request service, and the name of the requester. Unfortunately, these log files do not include the file sizes or the name of the media from which the file was fetched. These log files were intended to aid in monitoring and debugging the system, not for performance modeling. Many of the deficiencies of the version 1 logs were fixed in version 2. The version 2.1 and 2.2 logs were collected between January, 1994 to mid-July, 1994. These logs include file size and media name information, permitting a much more detailed analysis. Version 2.3 logs start in mid-July, 1994 and are still being collected at the time of this writing (January 1995). These logs include information about ingest as well as request activity.

2 Caching

When a user requests a file, the file is fetched from tertiary storage into secondary storage and made available to the requester. The file typically has a *minimum residency requirement* (three days in NDADS) to give the requester time to access the file. The archive systems needs to have enough disk storage to satisfy the minimum residency requirement.

While the file is disk-resident, a second request for the file can be satisfied without fetching the file from tertiary storage. These cache hits can reduce the load on the tertiary storage system, and also improve response times.

A large body of caching literature exists when all cached objects are of the same size. The Least Recently Used (LRU) replacement algorithm is widely recognized as having good performance in practice, although statistical algorithms with better performance have been proposed recently [6, 11].

Caching objects of widely varying sizes is somewhat more complicated, and has not received the same amount of attention. If one wants to minimize the number of cache misses, then it is much better to choose large files than small files for replacement, because removing large files frees up more space. The optimal replacement algorithm for variable size objects, with respect to cache misses, is the GOPT algorithm [2]: Let F be the set of cached files, and for file $f \in F$, let N_f be the time until the next reference to f and let S_f be the size of f. Choose for replacement the $f' \in F$ whose product $N_{f'} * S_{f'}$ is the largest.

The GOPT algorithm cannot be implemented (because it requires knowledge of future events), but it can be approximated. The Space-Time Working Set (STWS) algorithm [13] approximates GOPT be substituting P_f, the time since the last reference to f, for N_f.

While STWS can be implemented, it also requires a great deal of computation. For this reason, STWS is often approximated by what we call the STbin algorithm [8]: A file is put into a bin based on its size. The files in a bin are sorted in a list using LRU. To choose a file for replacement, look at the file at the tail of each bin and compute its $P_f * S_f$ product. Choose for replacement the file with the largest space-time product.

In our caching analysis, we use the LRU, STWS, and STbin algorithms. We assume a disk block size of 1024 bytes, and set a limit on the number of disk blocks that are available for caching. We trigger replacement when fetching a new file will cause the space limit to be exceeded, and we remove files until the space limit will not be exceeded. For the STbin algorithm, bin i holds files that use between 2^i and $2^{i+1} - 1$ blocks.

We execute the caching algorithms on traces generated from the 1994 log files (which have size information attached). We divide the logs into three month periods, to make the logs large enough to capture the steady-state hit rates, but also indicate changes in the access patterns.

The hit rate information is summarized in Table 1. The STWS and STbin algorithms have much better performance than the LRU algorithm. The STbin algorithm usually has performance comparable to that to the STWS algorithm, and sometimes has better performance. One surprising result is the high hit rate (up to 50%) that is possible with a moderate sized (5 Gb) cache. Given the nature of the archived data, hit rates were expected to be much lower.

When a file is fetched from tertiary storage, it remains on magnetic disk for at least three days. For a comparison, we present the disk storage requirements and the hit rates if an 3-day residency is observed, in Table 2. As the table shows, considerable more than 5 Gb of disk storage is required to satisfy the minimum residency requirement.

The resources required to fetch a file depend on the size of the file. For this reason, STWS is suboptimal is practice. Most vendors allow the user to tune the caching algorithm to reduce the penalty paid by very large files. A common technique is to assign to each file a weight computed as $P_f * S_f^c$ for a constant $c \leq 1$. In Table 3, we list the number of bytes transferred by each of the caching algorithms. LRU generally transfers the fewest bytes, closely followed by STWS. In these log files, STbin requires the transfer of many bytes (STWS transfers fewer bytes than STbin because it has a lower miss rate).

Disk Blocks (1k bytes)	Hit Rate LRU	STWS	STbin	Hit Rate LRU	STWS	STbin
	January, 1994 - March 1994			April, 1994 - June 1994		
1048576	.144	.234	.195	.155	.235	.189
2097152	.243	.314	.267	.202	.313	.194
3145728	.288	.341	.337	.272	.362	.286
4194304	.309	.355	.349	.292	.423	.448
5242880	.320	.364	.360	.304	.471	.500
	July, 1994 - September 1994			October, 1994 - December 1994		
1048576	.173	.270	.205	.127	.215	.198
2097152	.248	.308	.259	.155	.243	.222
3145728	.271	.328	.309	.181	.266	.245
4194304	.302	.340	.340	.219	.291	.256
5242880	.327	.366	.377	.252	.311	.287

Table 1: Hit rates for different cache replacement algorithms.

	1/94 - 3/94	4/94 - 6/94	7/94 - 9/94	10/94 - 12/94
hit rate	.175	.183	.193	.129
Storage	7.2 Gb	8.4 Gb	14.0 Gb	11.0 Gb

Table 2: Disk storage and hit rates for 3-day residency.

Disk Blocks	Hit Rate LRU	STWS	STbin	Hit Rate LRU	STWS	STbin
	January, 1994 - March 1994			April, 1994 - June 1994		
1048576	35.0 Gb	34.9	36.7	43.2	43.4	44.8
2097152	30.1	32.1	34.9	41.2	41.1	44.6
3145728	29.1	30.8	31.8	39.1	39.6	42.6
4194304	28.4	29.8	30.7	38.0	38.1	38.8
5242880	27.8	29.1	29.9	37.2	36.6	36.6
	July, 1994 - September 1994			October, 1994 - December 1994		
1048576	56.7	57.7	61.0	35.4	35.9	37.3
2097152	52.7	55.1	59.6	32.7	34.8	36.2
3145728	50.1	53.3	57.9	31.7	33.5	35.2
4194304	47.4	52.2	55.8	29.2	31.6	34.3
5242880	46.3	50.0	52.7	28.3	29.8	32.1

Table 3: Disk blocks moved for different cache replacement algorithms.

Year	Jan. - March		April - June		July - Sept.		Oct. - Dec.	
	total	unique	total	unique	total	unique	total	unique
1992	12806	73%	28899	61%	41296	67%	53253	65%
1993	47046	66	31513	69	49058	65	26106	61
1994	77415	62	92325	45	113594	55	74606	67

Table 4: Total and unique references (in percentage) to NDADS.

2.1 File Access Pattern Analysis

The success of caching depends on the access patterns. In this section we examine some aspects of the access patterns.

The number of possible cache hits depends on the number of duplicate references in the reference stream. In Table 4, we show the number of references and the number of unique references by three month period. The 1994 data shows that the STWS and STbin algorithm are getting close to the best possible hit rates.

The effectiveness of caching also depends on the average time between references to a file (the *inter-reference* time). In Figure 1, we plot the distribution of inter-reference times during 1994. To generate this plot, we scanned through all file accesses and searched for repeat accesses. Whenever a repeated reference was found, we incremented a histogram based on the number of days since the last reference. The plot shows that most repeat references occur shortly after an initial access, but that the inter-reference time distribution has a long tail. The average number of days between an access to a file, given that the file is accessed at least twice in 1994, is 27.5 days. There is a sharp peak at 3 days that does not fit well with the curve. We speculate that this is a side effect of the three-day residence period (users re-submit their request when they find that the files have been removed from disk storage area).

The effectiveness of STWS also depends on the distribution of file sizes. In Figure 2, we plot the distribution of sizes of the files accessed in 1994. The average size of a file accessed in 1994 is 560 Kbytes. Because of the wide range of sizes, we created the histogram by binning on the base two logarithm of the file size. Most of the files accessed in this archive are between 128 Kbytes and 1Mbyte in size. Few files larger than 2 Mbytes are accessed, but this number does not go to zero. Figure 3 shows the file access rate weighted by the file size. When we examine the number of bytes moved, files larger than 2 Mbytes account for a significant fraction of the system activity.

Finally, we look at the rate at which files of different sizes are re-accessed. In Figure 4, we plot the percentage of file accesses which are repeat accesses, binned on file size. This plot shows that small files (except for the very smallest) have a low re-access rate, and that the very large files have a high re-access rate. If the cost of transferring large files is significant, then STWS is a distinctly suboptimal caching policy. Because STWS strongly discriminates against very large files, it will often incur the cost of their transfer.

3 User Access Analysis

A model of request to the archive depends on the users of the system. In the accumulated log files, we have notices that the user population is growing. We first note that the user

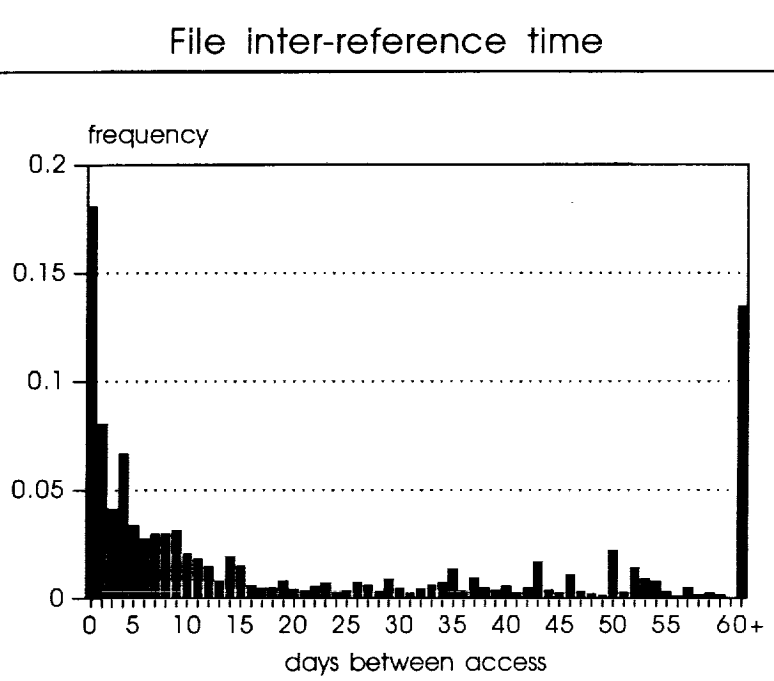


Figure 1: Distribution of file inter-reference times.

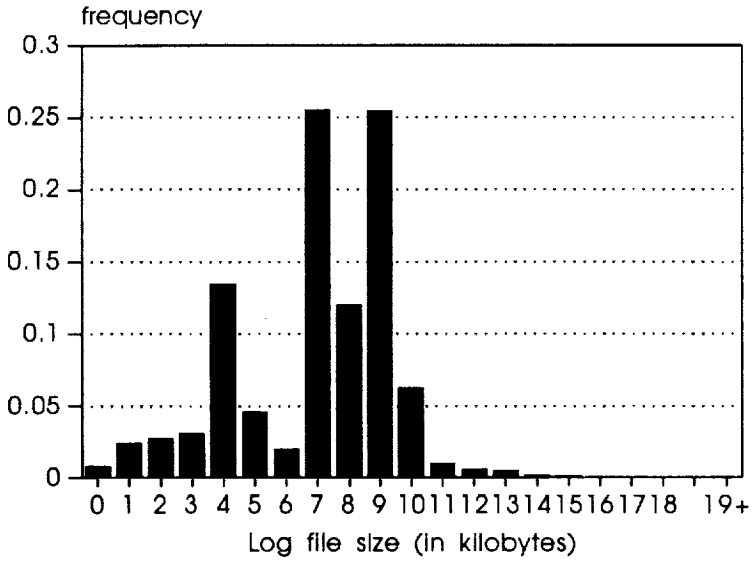


Figure 2: Distribution of file sizes.

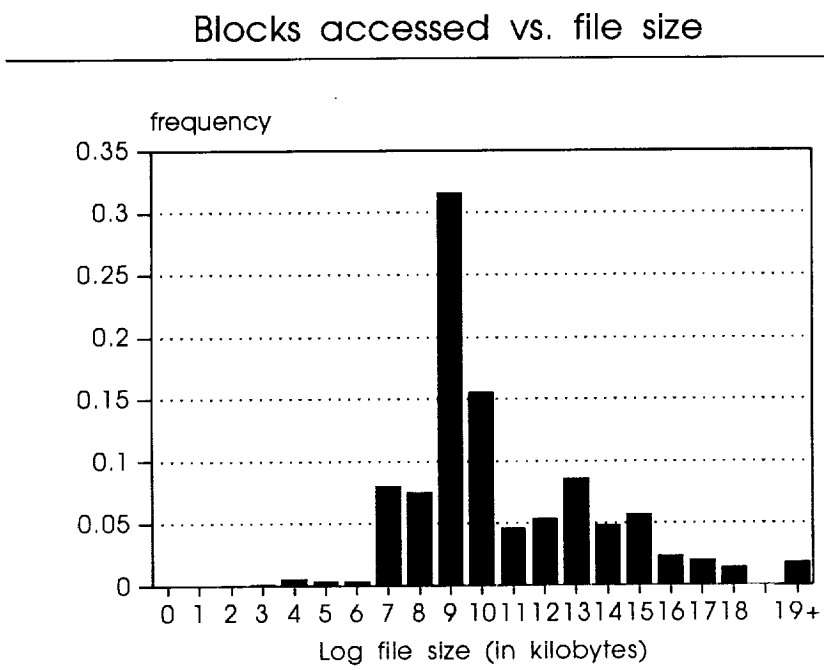


Figure 3: Distribution of file sizes, weighted by number of blocks.

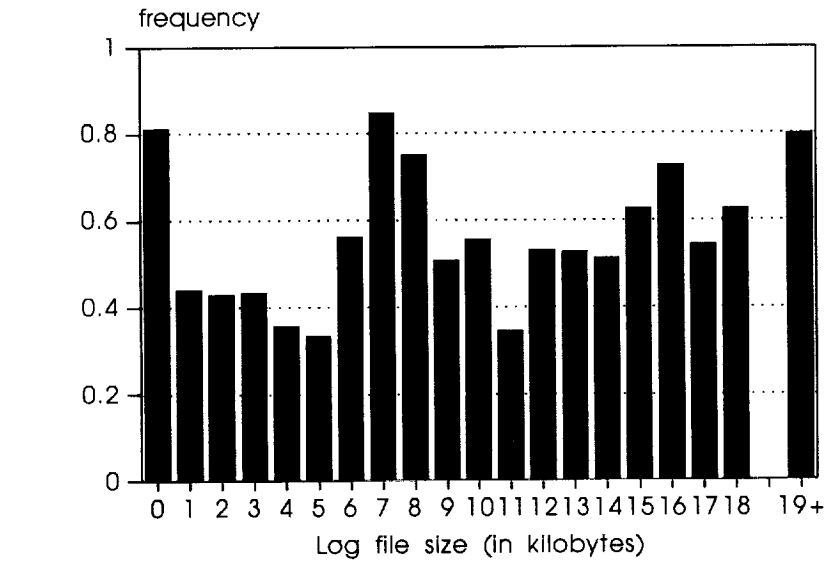


Figure 4: Probability of re-referenceing a file, by file size.

	Number of users			
year	Jan. - March	April - June	July - Sept.	Oct. - Dec.
1992	153	185	230	300
1993	430	552	677	607
1994	678	689	692	670

Table 5: Growth in the user population.

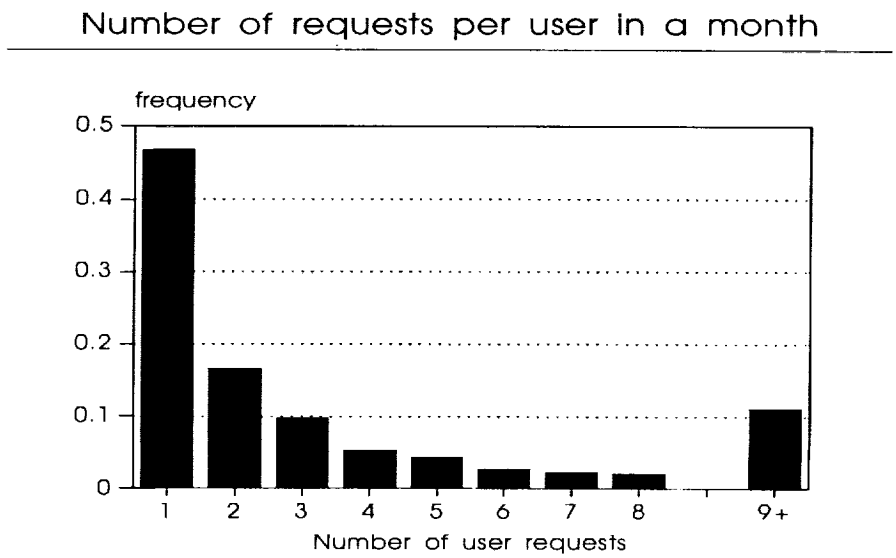


Figure 5: Requests per user.

population is growing, as is indicated by Table 5.

Most users make only a few requests to the archive. Figure 5 plots a histogram of the percentage of users that make different numbers of requests to the archive in a single month. This plot also shows that there is a moderate size core of users who make make requests. The top ten heaviest users make an average of 30 to 50 requests per month.

Finally, we plot the time between requests from a user in Figure 6. This plot shows that user activity is very bursty, as more than 80% of repeat requests occur within 3 days of the previous request.

We found that a large fraction of repeat requests for a file are due to the user that previously requested the file. The fraction of repeat request due to the same user is plotted in Figure 7, binned on the number of days since the last reference.

4 Clustering

The efficiency of a tertiary storage system depends in large part on how well the data in the archive is clustered with respect to the average request. The throughput of a drive in the tertiary storage device is zero while new platters are being loaded, or while the drive is seeking the file on the media. If the files of a single request are scattered throughout many media, and at widely varying locations in the media, the throughput of the tertiary storage device will be much lower than its potential.

The NDADS system is built over WORM storage, which has short seek times. Therefore,

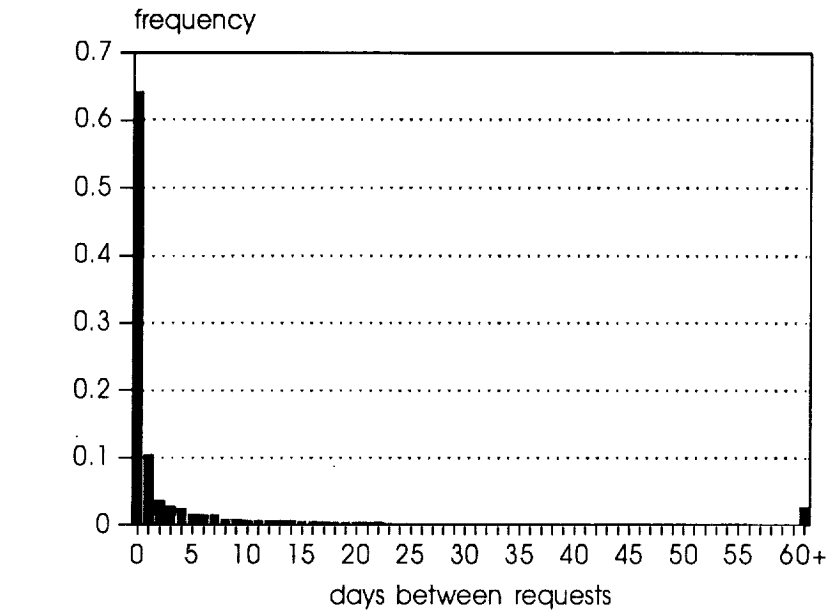


Figure 6: Time between repeat requests from a user.

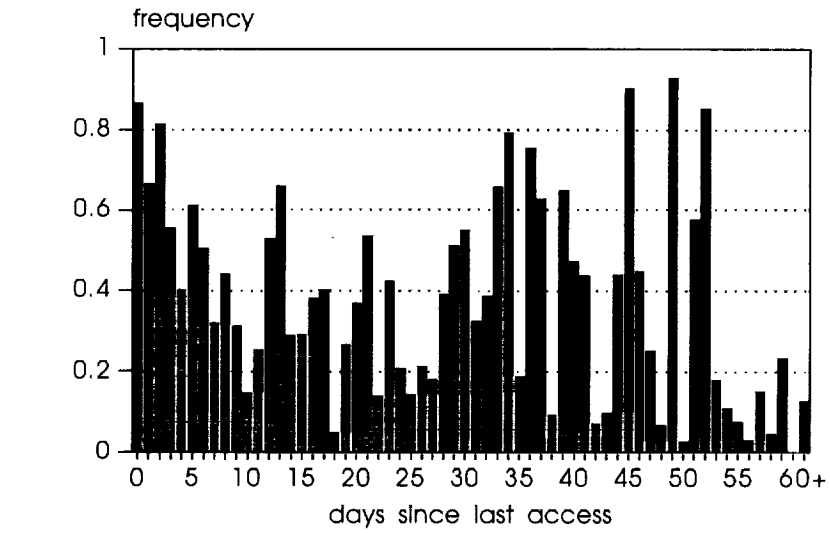


Figure 7: Fraction of repeat accesses from the original user.

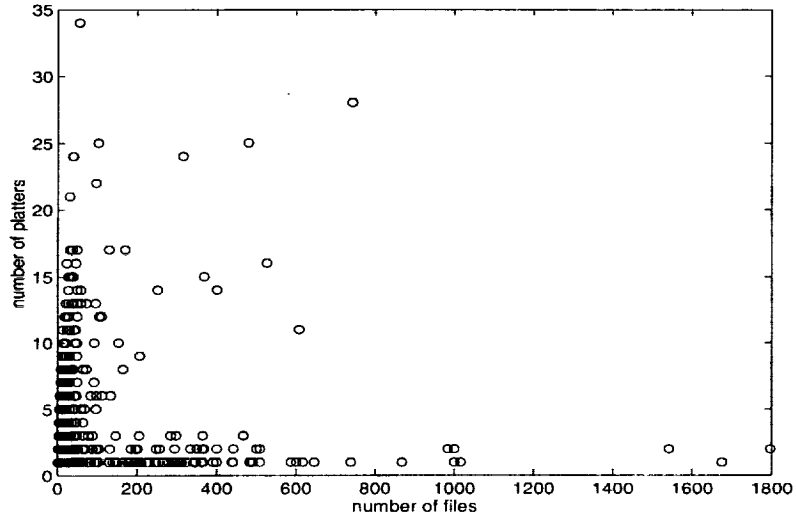


Figure 8: Scatter plot of platters per request vs. files per request. The data is was collected between July and September 1994.

the most expensive overhead occurs when a new platter has to be loaded to fetch a file. Also, the 1994 log files contain the platter on which each file is stored, but not the tracks on the platter.

Files in NDADS are divided into *projects* (i.e., the satellite that generated the images contained in the file). An optical platter contains files for only one project, (but a project may be spread over many platters) to simplify the management of the platters. This policy actually aids in clustering, because all files in a request must be from the same project. If a project generates enough data to require several platters, the files are assigned to the platters in a way that is hoped to reduce the number of platters that must be accessed to satisfy a typical access. This method of placement depends on the project and the expected type of access.

For every user request, we collected the number of files requested and the number of platters needed to satisfy the request. We found that the NDADS clustering of files onto platters is effective, as the average request asks for about 27 files, spread across about 2 platters. The number of platters required to satisfy a request is not correlated with the number of files in the request. This property is illustrated in Figure 8, which is a scatter plot of the number of platters to satisfy a request versus the number of files in the request. The data in this plot is taken from the period July 1994 to September 1994, but is typical of the total data. Most of the points in the plot are close to either the X or Y axis. The shape of the plot indicates that the clustering is appropriate for most requests, but a small fraction of the requests require a completely different clustering pattern (as is to be expected).

We also noted that some platters are accessed much more frequently than others. In Figure 9, we plot the number of platters that have different numbers of references. The plot shows that many of the platters receive only a few references, but that the distribution has a long tail. Eighty four of the platters are very hot, serving more than 500 files during 1994. The hottest platter served 112646 files.

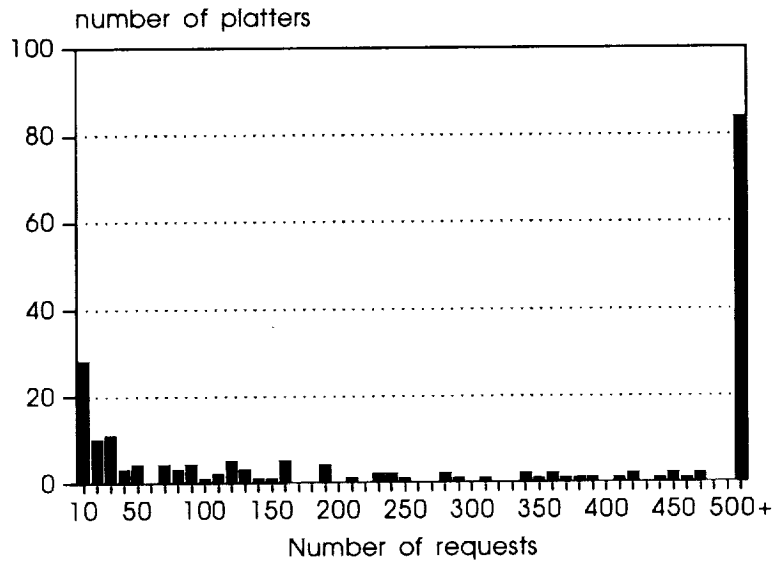


Figure 9: Number of platters receiving different numbers of references.

5 System load

We computed the *system load* by summing up the number of seconds required to service all request submitted during a period of time, then dividing by the number of seconds of the observed period. We plot the system load per month for 1993, 1993, and 1994 in Figure 10. While the month to month load shows great variability, the load per month does not follow a pattern that is strongly adhered to in all three years of the observation. However, we can note that there is a usage peak in March-April, another in July-August, and low demand in January.

We next plot the system load per day of the week in Figure 11. Here, we can find a strong trend, that people tend to submit requests on weekdays instead of weekends.

Finally, we plot the system load per hour of the day. We again see that people tend to submit requests during normal working hours. The strong peak in load during (Greenbelt) working hours also indicated that most users of NDADS work in the western hemisphere. We note that a survey of the email addresses of user requests shows international use of NDADS.

We have also recorded the system load due to ingest. Ingest contributes about .16 to the system load, and shows a pattern that varies in time that is similar to that of file requests.

6 Conclusions

We have studied the access characteristics of the access requests to the National Data Archive and Distribution Service (NDADS) of the National Space Science Data Center (NSSDC), of NASA's Goddard Space Flight Center. Much of NASA's electronic science data is available on through the NDADS archive. The log files present an opportunity to understand the access patters of requests to scientific archives.

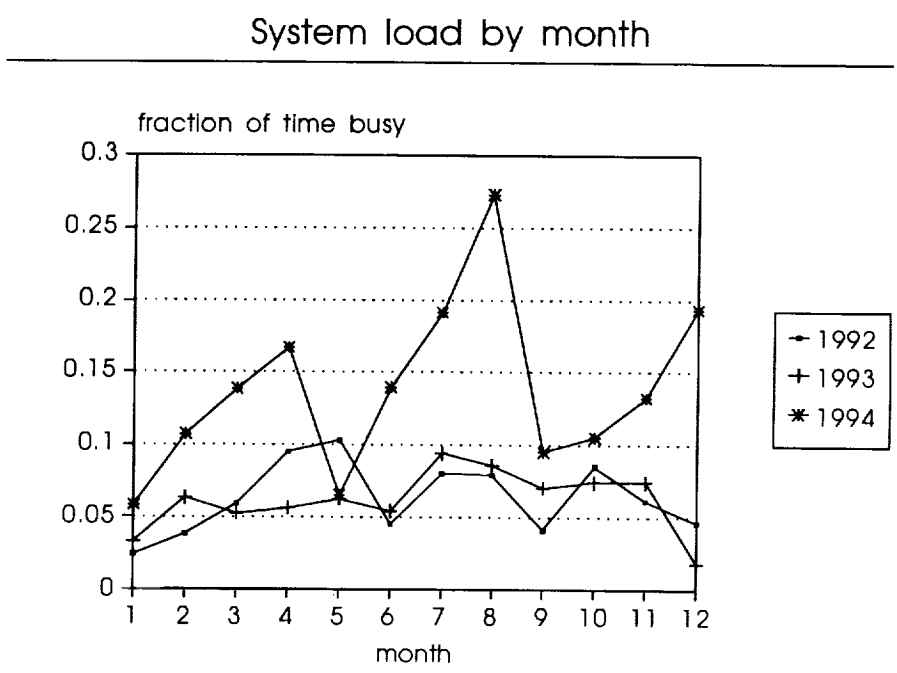


Figure 10: Average load per month. 1 is January and 12 is December.

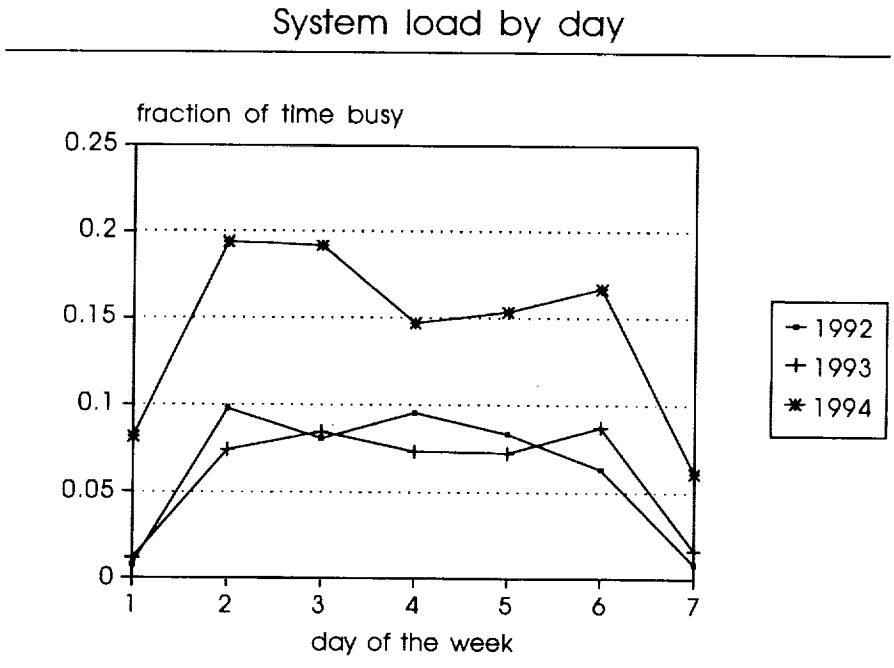


Figure 11: Average load per day of the week. 1 is Sunday and 7 is Saturday.

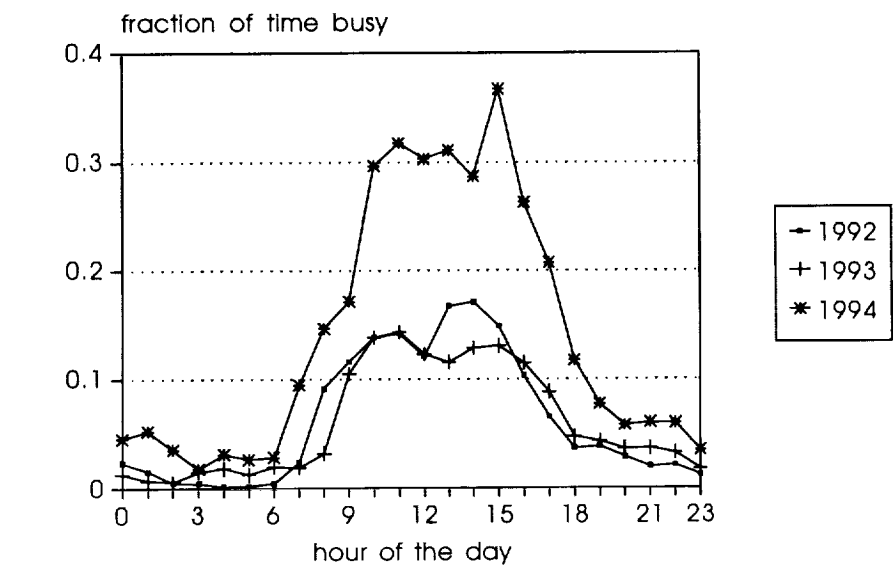


Figure 12: Average load per hour of the day.

We can make the following observations about the user request pattern:

- Caching can be effective. 59.4% of all files requested in 1994 had been requested previously in 1994. High hit rates (30% to 50%) can be achieved by using a space-time working set algorithm.
  - Many of the repeat requests are due to the same user. The high proportion of short-term repeat requests from the same user indicates some users are uncertain of whether their request was received. The high proportion of long-term repeat request from the same user indicates that NDADS is being used as a substitute for local storage.
  - While very large files constitute a small proportion of the total number of requests, they constitute a moderately large proportion of all bytes fetched from tertiary storage. Very large files tend to have a high repeat access rate. These two facts indicate that a caching algorithm should not discriminate too strongly against large files.
  - Access to a file is bursty. A large proportion of repeat accesses occur within 4 days of the previous access. The distribution of the time to the next access also has a long tail. These two facts suggest that a model of file access rates should have steady-state component and a bursty component triggered by an access.
- User request patterns tend to be very bursty. This fact combined with the high proportion of repeat requests that are due to the same user explains some of the bursty nature of file accesses.
- Access to NDADS tends to follow normal working hours. There is an increase of activity preceding important scientific events.

- The user community grew rapidly during the first year of operation, then grew at a slower pace during 1993 and 1994. The intensity of use by each user grew from 1993 to 1994.
- File access shows a great deal of clustering.
  - Most requests are satisfied by a few one or two platters. There is little correlation between the number of files requested and the number of platters required to service the request.
  - Clustering is important for performance. Although the average system utilization is low, the system load increases significantly during working hours and during certain months. If the time required to service a request doubled, NDADS would have difficulty in meeting peak demand.
  - Some data volumes are much more popular than others. During 1994, there were 84 very hot platters (i.e., served more than 500 files) and 28 very cold platters (i.e., served 1-10 files).

Acknowledgements

We'd like to thank Jeanne Behnke, Joe King, and Michael Van Steenberg for their help with this project.

References

[1] E.R. Arnold and M.E. Nelson. Automatic Unix backup in a mass storage environment. In *Usenix - Winter 1988*, pages 131-136, 1988.

[2] P.J. Denning and D.R. Sluts. Generalized working sets for segment reference strings. *Communications of the ACM*, 21(9):750–759, 1978.

[3] C.W. Ewing and A.M. Peskin. The masstor mass storage product at brookhaven national laboratory. *Computer*, pages 57–66, 1982.

[4] R.L. Henderson and A. Poston. MSS II and RASH: A mainframe unix based mass storage system with a rapid access storage heirarch file mamangement system. In *USENIX - Winter 1989*, pages 65–84, 1989.

[5] D.W. Jensen and D.A. Reed. File archive activity in a supercomputing environment. Technical Report UIUCDCS-R-91-1672, University of Illinois at Urbana-Chanpaign, 1991.

[6] T. Johnson and D. Shasha. 2Q: a low overhead high performance buffer management replacement algorithm. In *Proc. of the 20th Int'l Conf. on Very Large Databases*, pages 439–450, 1994.

[7] D.H. Lawrie, J.M. Randal, and R.R. Barton. Experiments with automatic file migration. *Computer*, pages 45–55, 1982.

[8] E. Miller, 1994. Provate communication. Thanks also to comp.arch.storage.

[9] E.L. Miller and R.H. Katz. Analyzing the i/o behavior of supercomputing applications. In *Supercomputing '91*, pages 557–577, 1991.

[10] E.L. Miller and R.H. Katz. An analysis of file migration in a unix supercomputing environment. In *USENIX - Winter 1988*, 1993.

[11] E.J. O'Neil, P.E. O'Neil, and G. Weikum. The lru-k page replacement algorithm for database disk buffering. In *Proceedings of the 1993 ACM Sigmod International Conference on Management of Data*, pages 297–306, 1993.

[12] A.J. Smith. Analysis of long-term reference patterns for application to file migration algorithms. *IEEE Trans. on Software Engineering*, SE-7(4):403–417, 1981.

[13] A.J. Smith. Long term file migration: Development and evaluation of algorithms. *Communications of the ACM*, 24(8):521–532, 1981.

[14] S. Strange. Analysis of long-term unix file access patterns for application to automatic file migration strategies. Technical Report UCB/CSD 92/700, University of California, Berkeley, 1992.

[15] A. Tarshish and E. Salmon. The growth of the UniTree mass storage system at the NASA Center for the Computational Sciences. In *Third NASA Goddard Conf. on Mass Storage Systems and Technologies*, pages 179–185, 1993.

[16] E. Thanhardt and G. Harano. File migration in the ncar mass storage system. In *Mass Storage Systems Symposium*, pages 114–121, 1988.

Evaluating the Effect of Online Data Compression on the Disk Cache of a Mass Storage System

Odysseas I. Pentakalos[1] and Yelena Yesha[2]
Computer Science Department
University of Maryland Baltimore County
Baltimore, Maryland 21228
and
Center of Excellence in Space Data and Information Sciences
Goddard Space Flight Center
Greenbelt, Maryland 20771

APPENDIX

A trace driven simulation of the disk cache of a mass storage system was used to evaluate the effect of an online compression algorithm on various performance measures. Traces from the system at NASA's Center for Computational Sciences were used to run the simulation and disk cache hit ratios, number of files and bytes migrating to tertiary storage were measured. The measurements were performed for both an LRU and a size based migration algorithm. In addition to seeing the effect of online data compression on the disk cache performance measure, the simulation provided insight into the characteristics of the interactive references, suggesting that hint based prefetching algorithms are the only alternative for any future improvements to the disk cache hit ratio.

I. Introduction

Mass storage systems are used in research environments for storing data generated by scientific simulations and satellite observations in amounts on the order of terabytes. The cost of storage devices of that capacity is still very high while the rate of increase in disk space requirements by the users grows continuously. This problem is especially evident in scientific research centers where enormous amounts of data are generated on a daily basis which must be archived so that they can be analyzed at a later time [1], [2].

In this study the actual system under consideration is the Unitree Mass Storage System (UMSS) used at NASA's Center for Computational Sciences (NCCS). The system administrators are experiencing a situation where they constantly need to purchase additional storage devices which are filled to capacity in a decreasing amount of time. The main resource whose utilization must be optimized in this case is storage capacity. Removing the redundancy in the data stored in the file system, by inserting an online compression/decompression module, is one method of increasing the effective capacity of the system without the addition of expensive hardware devices.

After considering various alternative locations in the system at which the compression algorithm could be placed we determined that the user interface would be the best choice. Some of the advantages of placing compression at the user interface are: a) does not impose an additional load on the storage servers CPU, b) reduces the amount of data that flows through the network, and c) does not require modifications to the Unitree code.

To evaluate the performance of compression on the specific data stored at NCCS, the ftp clients were modified to implement Ziv-Lempel and LZW compression transparently [3], [4], [5]. Sequential and pipelined implementations were tested against two sets of files and the performance of each implementation was compared based on file compression ratio and compression rate. An earlier paper describes the implementations and the results in detail [6].

[1]Email: odysseas@cs.umbc.edu
[2]Email: yeyesha@cs.umbc.edu

In this study we examine the effect of compression on the disk cache of the mass storage system. A simulation is used to determine the effect of compressing data on the hit-ratio of the disk cache, the number of migrations of files from the disk cache to robotic storage, and the total number of bytes migrating to robotic storage. We also look at two different migration algorithms and their effect on the hit ratio and the file migrations.

Section two gives a description of the system under consideration and reviews terminology that will be used throughout the rest of the paper. Section three describes the simulation used in this study. Section four describes the simulations performed and analyzes the results. Section five concludes the paper and discusses future work.

II. System Overview

The UMSS is a hierarchical mass storage management system which runs as a centralized application program on top of the Unix operating system and manages a hierarchical mass storage file system. The specific installation offers three levels in the storage hierarchy. Figure 1 shows the typical storage pyramid provided by most hierarchical mass storage systems. At the higher level it provides a disk array, with a total capacity of 150 GBs, which serves mainly as a cache for the lower levels. The second level has a capacity of 4.8 terabytes provided by four near-line robotic tape storage units. The third level is the off-line storage vault which has the slowest transfer rate serving as the long-term repository.

Users access files stored in the UMSS using the ftp protocol from their local workstations via a local area network. In addition to the ftp protocol, UMSS also provides an NFS interface to the file system but due to performance and security reasons the NFS protocol is not used by many installations including the one at NCCS. The UMSS was designed in a modular fashion in order to make possible its distribution over multiple host machines. Figure 2 shows a block diagram of the UMSS components [7].

Each of the components shown in figure 2 is represented by one or more independent daemon processes and is responsible for certain tasks. The "Name Server" resolves string file names used by the users, into unique integer identifiers, used internally by all the other components of the UMSS. The "Disk Server" keeps track of the files stored in the disk cache, providing the view of a Unix file system to the user. The "Disk Mover" is responsible for all transfers to and from the disk cache. The "Migration Server" controls the migration of files from the disk cache to lower levels in the disk hierarchy to ensure that the disk cache always has sufficient free space to operate efficiently. The "Tape Server" keeps track of the files stored in the tape storage units whether online or off-line. The "Tape Mover" performs all file transfers to and from a tape device. The physical device manager is responsible for managing the tape mounts, scheduling them in an order which maximizes the utilization of the system resources. Finally, the "Physical Volume Repository" is responsible for mounting and dismounting both automated online and off-line storage physical volumes [8]. Any files retrieved from the UMSS are first placed in the disk cache, if they are not already there, and then are transferred to the user. Likewise, any files stored into the UMSS are first stored in the disk cache and then they are moved to a lower level of the hierarchy through migration.

In an earlier paper we investigated the effectiveness of an online data compression algorithm placed at the user interface of a mass storage system [6]. For a sequential implementation the following inequality describes the trade-off in time of compressing the data online.

$$\begin{aligned} \frac{S}{R_t} &> \frac{S}{R_c} + \frac{S(1-r_c)}{R_t} \\ \frac{1}{R_t} &> \frac{1}{R_c} + \frac{1-r_c}{R_t} \\ R_t &< r_c R_c \end{aligned} \tag{1}$$

where S is the size of the file, R_t is the file transfer rate, R_c is the compression rate and r_c is the compression ratio normalized to the range $[0,1]$. The left hand side is the time it takes to transfer the file without compression and the left side with compression. If the compression rate of the compression algorithm used is faster than the transfer rate of the network between the client and the server then the embedded compression increases the effective capacity of the storage server at no

additional cost. Note that by cost here we mean the amount of time it takes to store a file into the mass storage system. If this inequality does not hold, the online compression algorithm increases the effective capacity of the system at the expense of added time when storing the file. The above inequality applies only to the sequential implementation. Assuming that the communication time between the parent and child processes is negligible we can derive a similar relation for the pipelined implementation as shown in inequality 2.

$$\frac{S}{R_t} > \max\{\frac{S}{R_c}, \frac{S(1-r_c)}{R_t}\} \qquad (2)$$

The total time of the pipeline is bounded by the maximum of each of its components. Which of the two components prevails will depend on the particular client making the request and on the network topology. If the client is connected locally relative to the server but is a slow machine then the compression component will prevail whereas on a fast machine which is a few hops from the server the transmission component will prevail.

III. Disk Cache Simulation

A trace-driven simulation of the disk cache was used to ascertain the effect on the hit ratio and on the migration of files caused by file compression and migration algorithm. A discrete event simulator was developed using the ftp request traces to drive the simulation. The disk cache size was varied from 150GB, which is the actual disk cache size at the NCCS site, to 250GB. Initially the cache was assumed to be empty. The disk cache was represented by a doubly linked list of structures which described each file entry. The information stored for each file were a unique file identifier, the file size, a timestamp of the time the file entered the disk cache, and an indicator of whether the file is stored in the disk cache or in the lower levels of the hierarchy.

Put requests were placed in the disk cache. If the file already resided in the cache or lower in the hierarchy the operation was processed as an update, ensuring that only one copy of the file existed in the entire mass storage system. For get requests, if the file existed in the disk cache then the request was considered a hit. If the file existed lower in the hierarchy it was staged in the disk cache. If the file requested did not exist in the hierarchy, it was processed as if it was in the lower levels of the hierarchy and a new entry was created for the file in the disk cache.

Migration in simulated time was performed using a high water mark as in the UCFM. If the amount of free space in the cache went below the high water mark of 75% the total disk cache capacity, files were migrated to the lower levels of the hierarchy to create more space. Two different migration algorithms were tested. The first one, was LRU based, selecting files to migrate which had resided in the cache the longest without being referenced. The second algorithm was based on the file size, migrating larger files first.

Since it would be impractical to collect the compression ratios for each of the files in the mass storage system each simulation run used a fixed compression ratio. The simulation was run for various compression ratios ranging from 0% to 60% compression.

IV. Results

The ftp interactive request logs for a period of three months were used to run the simulation. The total number of references in that three month period was approximately 106,000. The references from the first two months were used for bringing the disk cache to a warm state. Then the number of hits, the hit ratio, the number of files migrating to tertiary storage, and the total number of bytes migrated were measured for fixed values of compression ratio. The simulation was run also for two different migration algorithms. The first migration algorithm, which selected files to migrate if they had resided on the disk cache the longest without being referenced, will be referred to as the *LRU based algorithm.* The second algorithm which selected files to migrate based on their file size will be referred to as the *Size based algorithm.* The hit ratio was computed as the number of hits per day over the number of get requests on that specific day.

One important observation that was made about the reference patterns used in this mass storage system was that the requests do not exhibit significant temporal locality. Users do not tend to

re-use their files very frequently as in a typical file system. This implies that this specific mass storage system is used more as an archive than as a typical file system. Since the working set of the get request stream continuously changes, only low hit ratios are possible regardless of size increases to the disk cache.

In order to be able to compare the hit ratios measured with some sort of an optimal hit ratio we run the simulation on the same trace data setting the compression ratio to a value very close to zero. This allowed all the files to fit within the disk cache, imitating a disk cache of an enormous size, generating no migrations. This experiment was used to generate the optimal (OPT) disk cache hit ratios. The same method was used to compute the hit ratio of this cache as in the other cases. Table I summarizes the effect of compression on the number of hits for each of the experiments. The table is divided in three major column groups for each of the migration algorithms. The first column group shows the results for the LRU based migration algorithm, the second column group for the Size based migration algorithm, and the last column shows the results for the OPT disk cache. The first two column groups consist of three columns, one for each of three different compression ratios attempted. Comparing the results from the two migration algorithms against the results under OPT we see that the number of hits for both algorithms are very close to the optimal. Compression does not affect the hit ratio very much and this is because the disk cache is large enough to support the hits in the reference patterns. It should be noted that the LRU based algorithm exhibits the inclusion property as expected since the number of hits is non-decreasing with increases in the disk cache size. On the other hand, the size based algorithm in certain cases decreases with a larger effective disk cache size.

The hit ratios were also plotted in figure 3 for various compression ratios. The plot on the top shows the hit ratio variation with respect to the compression ratio for the size based migration algorithm and the bottom plot shows the variation for the LRU based migration algorithm. It is apparent from these figures that size based migration provides higher hit ratios than the LRU based algorithm. The variation in compression ratio does not have significant effect on the hit ratio and the reason for this is the same as discussed in the previous paragraph. This implies that adding additional disks to the disk cache will not have any effect on the hit ratio based on the references analyzed. Also any further effort in improving the hit ratio by varying the migration algorithm will not generate any significant improvement on the hit ratio. The only possible method of increasing the hit ratio would be to develop a prefetching algorithm that is based on hints provided by the user.

The second part of the simulation analysis focused on the migrations. Since migration involves the use of tape drives from the robotic silos it is an expensive operation. Thus, reducing the number of migrations or the total number of bytes migrating to the tape will improve the mass storage system's performance. Figure 4 shows the number of files migrating versus compression ratio for the two migration algorithms. The LRU based algorithm maintains a consistent number of migrations and tends to smooth the migration operations over time. It appears that the effect of file compression is minimal. Looking at the peaks in the LRU based algorithm it appears that compression simply shifts the migration effects but does not reduce their number. The size based migration algorithm decreases significantly the number of migrations but it has the negative effect of generating on certain days tremendous migration traffic. Analyzing the file sizes for both get and put requests we found that the mean file size of files stored in the storage system is an order of magnitude larger than the mean file size of files retrieved. Since the size based algorithm removes larger files first, eventually it runs out of large files and it has to remove a huge number of small files to free space in the disk cache.

Figure 5 shows the number of bytes migrating to robotic storage for various compression ratios. It is apparent that for both migration algorithms the higher compression ratio provides significant reduction in the number of bytes that need to migrate. The size based migration algorithm provides better performance throughout the simulation period. The time it takes the system to pro-

TABLE I
Number of Cache Hits over Compression Ratio

	LRU Based			Size Based			
r_c	0.0	0.2	0.4	0.0	0.2	0.4	OPT
1	285	285	286	285	286	286	286
2	87	87	87	104	104	104	105
3	186	186	186	186	186	186	202
4	342	342	342	343	343	343	352
5	235	241	242	435	435	435	493
6	1086	1087	1088	1089	1088	1087	1130
7	1323	1323	1323	1500	1500	1500	1698
8	143	143	143	145	145	145	153
9	60	60	60	63	63	61	63
10	250	250	250	248	248	248	252
11	321	321	321	317	317	318	324
12	422	422	422	434	434	434	464
13	371	371	371	354	355	355	409
14	376	381	381	376	376	377	436
15	1249	1249	1251	1244	1243	1244	1256

cess a migration involves an overhead time and a data transfer time. The overhead time consists of mounting the tape on a tape drive, a seek time to place the tape drive heads at the proper location, a rewind time after the data have been written, and an unmount time. Reducing the number of migrations from the disk cache affects the overhead time while reducing the number of bytes migrating to robotic storage reduces the data transfer time.

V. Conclusion

We evaluated the performance of an online compression algorithm on the disk cache of a mass storage system. A trace driven simulation of the disk cache was used for the evaluation. The traces used to drive the simulator were collected from the ftp logs of the system. The simulation was configured to match the disk space and migration algorithm of the system at NCCS. The effect of compression was simulated by uniformly reducing the file size of the get and put requests. Various compression ratios were used in the simulation. The simulation also evaluated two different migration algorithms, specifically an LRU based and a size based algorithm.

One important observation that was made about the references at this mass storage system was that the working set continuously changes. This implies that the disk cache hit ratio cannot be improved significantly by increasing the disk cache size since get operations are usually to files that were stored in the mass storage system a very long time in the past. This effect was evident by comparing the two migration algorithms against a disk cache which was large enough to store all files stored during the three month evaluation period. As a result both algorithms attained hit ratios very close to the optimal hit ratios of the huge cache. Comparing the two migration algorithms we found that the size based algorithm decreases the total number of bytes migrating to tertiary storage at the expense of causing occasional peaks in the number of files migrating. Both algorithms were not affected by the compression ratio due to the fact that the disk cache is of large enough size to cover the intereference pattern of the requests.

Future work will focus on evaluating various prefetching algorithms. The current simulation suggested that only the use of user hints and an appropriate prefetching algorithm can improve the hit ratio of this system. The use of transparent informed prefetching could be applied to improve

the hit ratio of the disk cache by exploiting application level hints about future file accesses [9]. Another area of future research is the implementation and evaluation of migration algorithms based on a combination of file size and cache residency time as described in [10], [11]. This simulation analysis showed that size based migration reduces the number of bytes that migrate to tertiary storage but occasionally it produces a large number of migration loads. By using a migration algorithm based on the space time product we expect that the migration peaks will disappear, while maintaining the lower number of bytes migrating.

Acknowledgements

We would like to thank Adina Tarshish, Ellen Salmon and George Rumney from NASA's Center for Computational Sciences at Goddard Space Flight Center for providing the ftp traces and the NCCS file set used for testing our ideas.

REFERENCES

[1] Randy H. Katz, Thomas E. Anderson, John K. Ousterhout, and David A. Patterson, "Robo-line Storage: Low Latency, High Capacity Storage Systems over Geographically Distributed Networks", Tech. Rep. UCB/S2K-91-3, University of California, Berkeley, March 1994.

[2] Ethan L. Miller and Randy H. Katz, "An Analysis of File Migration in a Unix Supercomputing Environment", Tech. Rep. UCB/CSD-92-712, University of California, Berkeley, March 1993.

[3] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression", *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[4] Debra A. Lelewer and Daniel S. Hirschberg, "Data Compression", *ACM Computing Surveys*, vol. 19, no. 3, pp. 261–296, September 1987.

[5] Terry A. Welch, "A Technique for High-Performance Data Compression", *IEEE Computer*, vol. 17, no. 6, pp. 8–19, June 1984.

[6] Odysseas I. Pentakalos and Yelena Yesha, "Online Data Compression for Mass Storage File Systems", manuscript available from the author, July 1994.

[7] Adina Tarshish and Ellen Salmon, "The Growth of the Unitree Mass Storage System at the NASA Center for Computational Sciences", in *3rd NASA GSFC Conference on Mass Storage Systems and Technologies*, College Park, Maryland, October 1993, pp. 19–21.

[8] Convex Computer Corporation, *Unitree++ System Administration Guide, First Edition*, Convex Press, Richardson, Texas, 1993.

[9] Hugo R. Patterson, Garth A. Gibson, and M. Satyanarayanan, "A Status Report on Research in Transparent Informed Prefetching", *Operating Systems Review*, vol. 27, no. 2, pp. 21–34, April 1993.

[10] Alan Jay Smith, "Analysis of Long Term File Reference Patterns for Application to File Migration Algorithms", *IEEE Transactions on Software Engineering*, vol. SE-7, no. 4, pp. 403–417, July 1981.

[11] Alan Jay Smith, "Long Term File Migration: Development and Evaluation of Algorithms", *Communications of the ACM*, vol. 24, no. 8, pp. 521–532, August 1981.

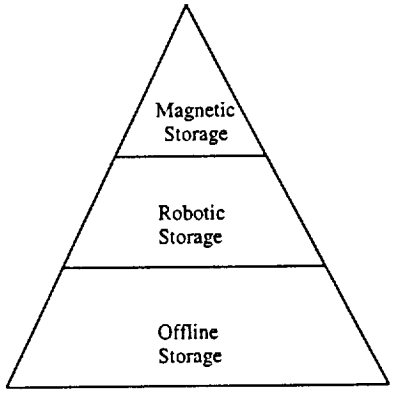


Fig. 1. **Hierarchical Storage Pyramid**

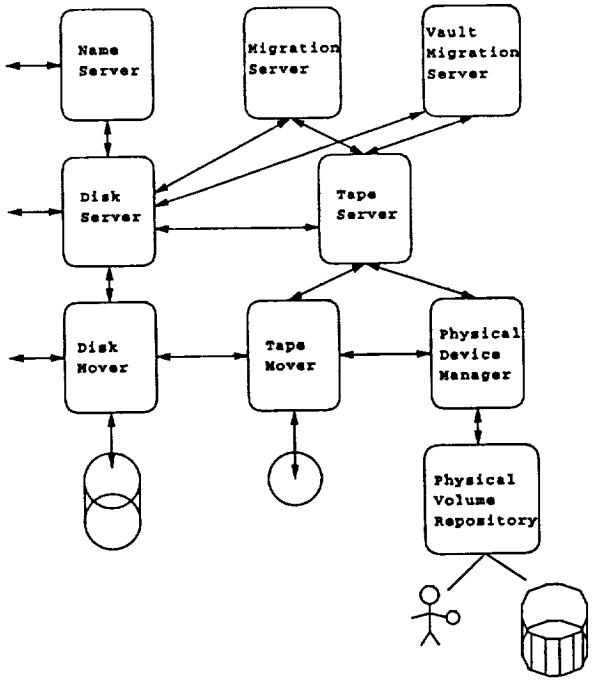


Fig. 2. **UMSS Block Diagram**

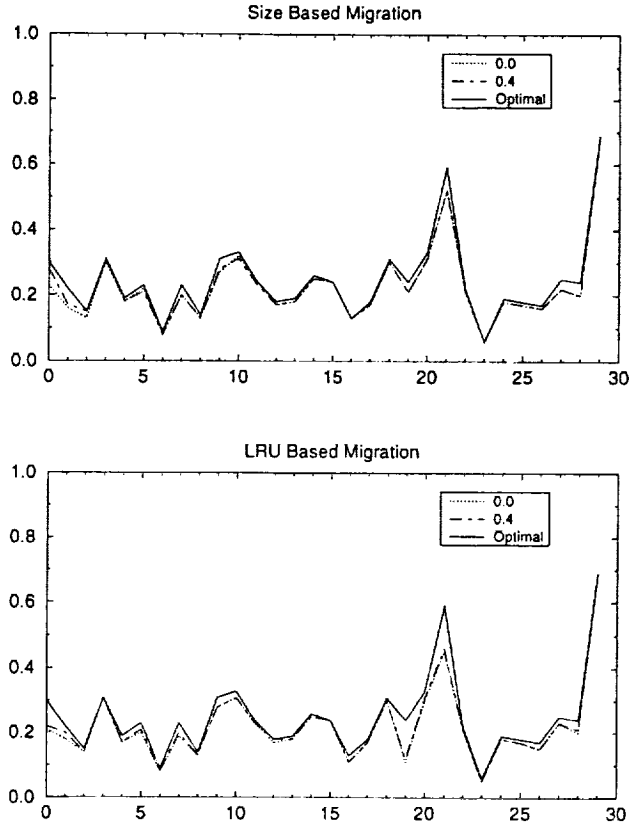


Fig. 3. **Hit-Ratio versus Compression Ratio**

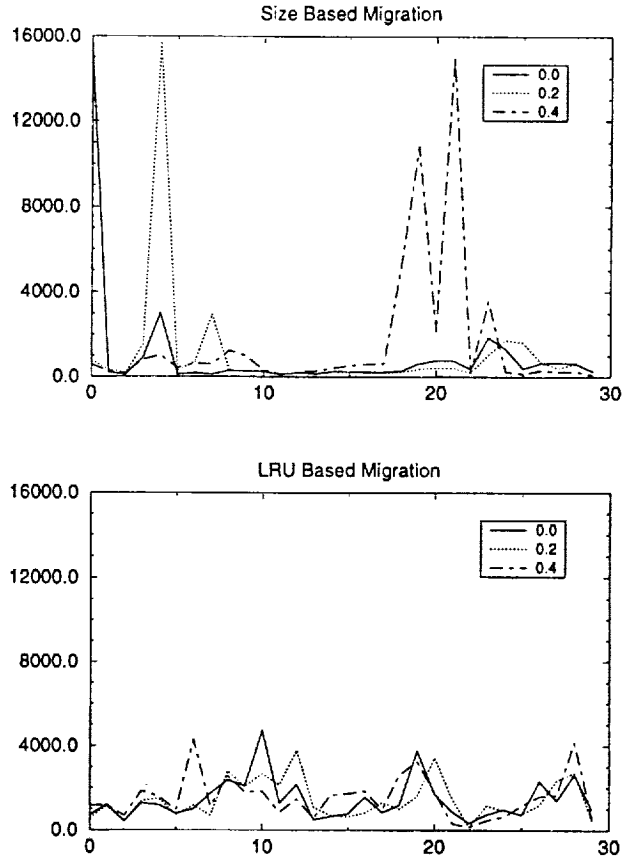


Fig. 4. **Number of Migrations versus Compression Ratio**

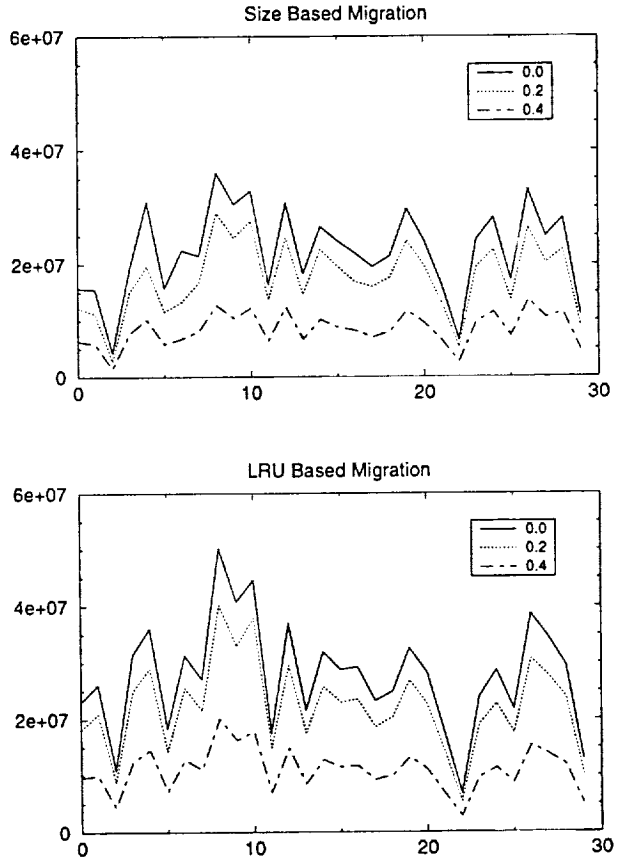


Fig. 5. **Bytes Migrated versus Compression Ratio**

A Terabyte Linear Tape Recorder

John C. Webber
Interferometrics Inc.
8150 Leesburg Pike
Vienna, VA 22182
(703) 790-8500
webber@interf.com

A plan has been formulated and selected for a NASA Phase II SBIR award for using the VLBA tape recorder for recording general data. The VLBA tape recorder is a high-speed, high-density linear tape recorder developed for Very Long Baseline Interferometry (VLBI) which is presently capable of recording at rates up to 2 Gbit/sec and holding up to 1 Terabyte of data on one tape, using a special interface and not employing error correction. A general-purpose interface and error correction will be added so that the recorder can be used in other high-speed, high-capacity applications.

The VLBA recorder was developed specifically for recording VLBI data using the Very Long Baseline Array of radio astronomy antennas built by the National Radio Astronomy Observatory. It is an evolution of the technology developed for the NASA Mark IIIA VLBI recording system at MIT Haystack Observatory. Its characteristics may be summarized as follows:

Recording medium:	1-inch-wide D1-equivalent 16 m thick tape
Head type:	38 μm width, single-crystal ferrite
Bit density:	56,000 flux transitions per inch
Format:	continuous linear tracks, NRZM, magnetic saturation
Tape speed:	For 9 Mbit/sec data, 160 inches per second
MTBF:	10,000 hours typical
Head life:	5,000-20,000 hours depending on environment
Head replacement cost:	$1/hour
Maintenance:	headstack and tape path cleaned with a cotton swab at each tape change.

A single headstack writes and reads 34 data tracks at a time. The heads are 38 μm wide and are separated by 698.5 μm, so that potentially 698.5/38 = 18.38 passes could be written on the tape. However, some allowance for guard bands between tracks must be made, since the magnetic gap is exactly perpendicular to the direction of tape motion and there must be no crosstalk between tracks. A practical limit is 16 passes, which gives a track spacing of 43.7 μm with a guard band of about 5 μm. There are thus 544 data tracks on the tape. Future improvements in tapes and heads are expected to increase this number.

The tape is a D1- or S-VHS equivalent tape available from both 3M and Sony. This tape is 16 m thick, and 20,500 usable feet are contained on a 16-inch reel (only 14-inch reels are currently used). The bit density supportable on this tape is 60,000 bits per inch, so each track contains 14.4 Gigabits. The whole tape with 544 tracks then holds about 8.03 Terabits, or one Terabyte. Using the error-correcting format to be developed, this becomes 788 Gbytes of user data. The cost of one reel of this tape is presently about

$1500, or $1.90 per Gigabyte. At the maximum user data rate supported by one headstack, namely 69.44 Mbyte/sec, one tape lasts approximately 192 minutes. Up to 4 headstacks may be mounted on one transport, yielding an aggregate recording rate of 278 Mbyte/sec with a recording time of 48 minutes. This is also the time required to duplicate a tape.

Typical tape life is several hundred read-write cycles including shipping once per month in uncontrolled conditions.

INTERFACES

The data interface in the present VLBA recorder is a set of parallel data lines, each supplying data directly to a single head in the headstack. Formatting of the user data consists of adding synchronization words, time codes, identification data, and parity bits in an external formatter. These bits are simply transferred directly to tape. On playback, the signals from each head are amplified, equalized, and routed to bit synchronizers which recover the clocks from each data stream. All further processing is performed in an external unit which recovers the synchronization codes, de-skews the tracks, and combines the data into a desired format.

In the system under development, the recorder will be responsible for all functions of formatting and deformatting. The user will supply data over a standard interface and recover data from the recorder over the same interface.

For the data rates of concern, there is a prime "standard" interface, namely the High-Performance Parallel Interface, or HIPPI, defined in ANSI X3.183-1991, as defined by the ANSI Task Group X3T9.3. We have adopted this interface as the standard recorder data interface for both record and playback for all applications. The HIPPI channel consists of 32 balanced ECL signals with a common 25 MHz clock and a transfer protocol allowing bursty transmissions. For applications requiring less than the full channel capacity, the recording speed may be varied or the data padded with dummy data to be stripped at playback time.

The HIPPI channel is a one-way device, so two HIPPI channels are needed in order to accommodate both record and playback functions. Commercially available chips provide a single 100 Mbyte/sec HIPPI interface. In the prototype system, two headstacks will be employed, enabling a maximum of 138.88 Mbyte/sec to be recorded. Since the recorder speed can be set with very fine resolution, it will be chosen such that the bit density remains at 60,000 bits per inch.

Since other high-speed protocols and fiber optic interfaces such as advanced ATM are coming into use, it is essential that the recorder be expandable to accommodate them. The plan is to add, for example, an ATM-to-HIPPI interface and continue to operate the recorder exclusively from the HIPPI interface. This simplifies the interfacing problem by placing it outside the recorder proper.

A single low-speed interface will suffice to set the recording mode and control the playback process. This is envisioned to be a 9600 baud RS-232C interface, permitting operation by any computer.

ERROR CORRECTION

For an individual tape track, which is the minimal recording sub-channel, the important characteristics for the VLBA recorder as it is presently used are as follows:

Random errors:	Bit Error Rate (BER) $< 3 \times 10^{-4}$ with 10-year-old tape; typically 3×10^{-6} with new tape, 3×10^{-5} with 3-year-old tape
Burst errors:	Typical length 100-1000 bits; typical rate 1 burst in 10^7 bits; bursts usually cause loss of bit synchronization

Random errors are caused by low signal-to-noise ratio (SNR), imperfect equalization, and imperfect bit synchronization. Burst errors are caused primarily by tape defects and are consequently highly dependent on the particular tape in use; any system of error correction must accommodate the worst-case tape. In order to minimize the data lost to dropouts, the distance between sync words should be comparable to the size of the dropouts.

For typical imaging data, a bit error rate better than 1×10^{-9} is required. Other applications require bit error rates as low as 1×10^{-13}.

Recently, VLSI chip sets which implement Reed-Solomon error correction algorithms have become available, and some can run at the data rates in which we are interested. Such chips are available from such manufacturers as LSI Logic and CNR, Inc. It appears that suitable devices for this application are available from Advanced Hardware Architectures (AHA). In particular, the AHA4010 device is attractive because of the following:

Input data format:	8-bit parallel (byte organization)
Data rate:	10 Mbyte/sec (80 Mbit/sec)
Max block length:	255 bytes (programmable)
Max errors correctable:	10 per block (or 20 erasures)
Other features:	No external buffers or control required
Cost:	$30 each in large quantities

Data will first be coded into data blocks of length 235 bytes. For each such block, we will program the AHA4010 to add 20 error correction bytes to make the total block length 255. This is an overhead of 8.5%.

This scheme will permit correction of up to 10 errors or 20 erasures (or a linear combination of both). Even with a raw bit error rate of 1×10^{-4}, the corrected code block error rate is predicted to be 7×10^{-15}, which satisfies our requirements. The errors will be decorrelated by interleaving the data over a distance long enough so that only one or two bytes from each code block are contained within a single error burst.

CONFIGURATION

The VLBA recorder uses a Metrum Model 96 tape transport, which is a full-size rack of hardware. In the new configuration, it will contain:

VME-based control computer
HIPPI interfaces with buffers
Error correction/formatting boards

Analog write drivers
Analog playback amplifiers/equalizers
Clock recovery (bit sync) boards
Deskewing buffers
Transport and headstack motion controllers
Power supplies

The prototype system will be equipped with two headstacks and so will be capable of recording up to about 138 Mbyte/sec of user data.

RECORDING AND PLAYBACK PROCESS

In addition to error correction, some formatting must be introduced in the form of synchronization words, modulation, and longitudinal parity. Sync words are absolute identifiers of tape block start points. Data modulation by a pseudo-random sequence will guarantee roughly equal numbers of ones and zeroes in the data regardless of content. The addition of longitudinal parity bits on each track will force sufficient transitions in the NRZM format so that good bit synchronizer performance can be maintained.

Upon playback, the parity, modulation, sync, and other formatting information must be undone and stripped out, and the error correction bytes used to restore the original user data. This data must then be reformatted so that it can be transmitted over the HIPPI interface back to the user. A summary of the recording and playback process is shown as Figure 1.

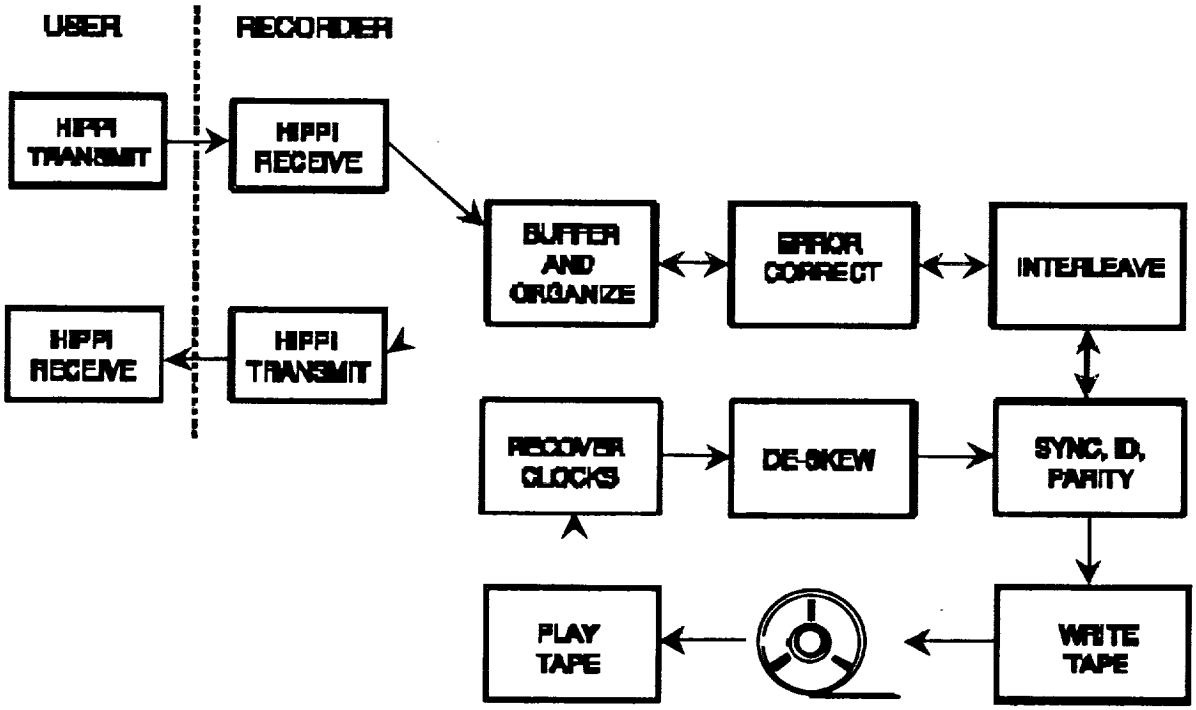


Figure 1. Data flow to and from the recorder

In much of the recording and playback electronics, the same circuitry can be used to perform both the recording and playback processing.

REFERENCES

"A High Data Rate Recorder for Astronomy", H. F. Hinteregger, A. E. E. Rogers, R. J. Cappallo, J. C. Webber, W. T. Petrachenko, and H. Allen, *IEEE Transactions on Magnetics*, **27**, No. 3, p.3455 (1991).

"Very Long Baseline Radio Interferometry: The Mark III System for Geodesy, Astrometry, and Aperture Synthesis", A. E. E. Rogers *et al.*, *Science*, **219**, p.51 (1983).