Performance Modeling of Network-Attached Storage Device Based Hierarchical Mass Storage Systems

Odysseas I. Pentakalos* and Daniel A. Menascé†

October 13, 1995

Abstract

Network attached storage devices improve I/O performance by separating control and data paths and eliminating host intervention during the data transfer phase. Devices are attached to both a high speed network for data transfer and to a slower network for control messages. Hierarchical mass storage systems use disks to cache the most recently used files and a combination of robotic and manually mounted tapes to store the bulk of the files in the file system. This paper shows how queuing network models can be used to assess the performance of hierarchical mass storage systems that use network attached storage devices as opposed to host attached storage devices. Simulation was used to validate the model. The anaalytic model presented here can be used, among other things, to evaluate the protocols involved in I/O over network attached devices.

Keywords: Mass storage system, hierarchical mass storage systems, network attached devices, third party protocols, queueing network modeling.

1 Introduction

Most current mass storage systems at national laboratories and supercomputer centers are based on the server attached storage model. In this model all storage devices are attached to a single mini or supercomputer using high speed busses and I/O channels. A request for access to a storage object arrives at the server which transfers the data, through the server's main memory, to or from the devices attached to the server. As the arrival rate of requests increases, the server becomes a bottleneck since all data needs to flow through its main memory.

*Odysseas Pentakalos is with the University of Maryland Baltimore County and the Center of Excellence in Space Data and Information Sciences, odysseas@cs.umbc.edu

†Daniel A. Menascé is with George Mason University and CESDIS.

To overcome this problem the new paradigm of network attached storage has been emerging in the design of mass storage system hierarchies. Data storage systems based on network attached devices have been termed fourth-generation storage systems. Under the network attached storage model, storage devices are no longer attached directly to the server but rather to a network. Such systems are being developed currently at Los Alamos National Laboratory, the National Center for Atmospheric Research, and the National Storage Laboratory at the Lawrence Livermore National Laboratory [1, 2, 3, 4].

Hierarchical mass storage systems, such as the UNITREE [5] and HPSS [6, 7], use a large disk cache, usually consisting of many RAID disks, to store the files that have been used more recently, and robotically mounted tapes to store the bulk of the files in the file system. The two main operations on a mass storage system are file storage (put) and retrieval (get). Files stored in the mass storage system are placed on the disk cache and they migrate to the tapes if they are not used for some specified time. A request to retrieve a file may be satisfied from the disk cache (a cache hit) if the file is stored in the cache or from the tape via the disk cache (a cache miss).

In previous works [8, 9], the authors have used queuing network based models to assess the performance of hierarchical mass storage systems that use host-attached devices. In this paper we show how queuing network models can be used to study the performance of hierarchical mass storage systems that use network-attached storage devices.

The rest of the paper is organized as follows. Section two describes the general operation of network-attached storage devices including HIPPI, IPI-3, and the several transfer protocols considered in this paper. Section three describes the queueing network model, its parameters and service demand equations used for solving it. Section four describes the validation of the system and section five describes the numerical results from various experiments that we conducted. Section six concludes this paper with a discussion of our plans for future work.

2 Network-Attached Device Operation

This section starts by describing the typical topology of a system that uses network attached storage devices. Then, then the High-Performance Parallel Interface (HIPII) protocol stack and the device independent Intelligent Peripheral Interface (IPI-3) are summarized. Finally, several third-party protocols are described.

2.1 System Topology

A typical implementation of a storage system which uses network-attached storage devices is shown in Fig. 1.

The system consists of a File Server with a Host Attached (HA) Storage Device, a Storage Access Control Network (SACN), a Storage Unit Control Network (SUCN), a High Speed Data Network, and a number of Network-Attached (NA) Storage Devices. The File Server manages the storage of files in the storage system. It is responsible for maintaining knowledge of the location of each file within the system, performing name resolution for user requests, authorizing access to files,

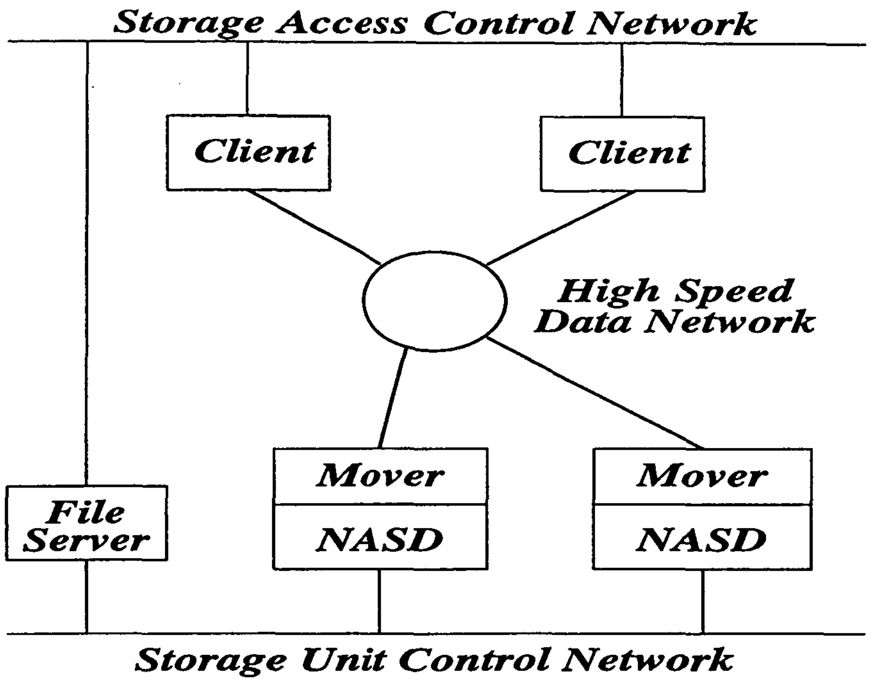

Figure 1: **Mass Storage System with Network Attached Storage Devices**

and managing the flow of files among the different levels of the mass storage system. Clients send requests for retrieval or storage of files through the Storage Access Control Network. The File Server controls the operation of the network attached storage devices through the Storage Unit Control Network. Finally, data flows between a the File Server and a storage device through the High Speed Data Network. In many cases, the Storage Access Control Network and the Storage Unit Control Network are both the same Ethernet physical network, whereas the High Speed Data Network is a point-to-point HIPPI connection. Associated with each NA device there is a Mover process. The mover runs in a processor that controls the NA device and controls the flow of data between the two devices involved.

In the model described in section 3, we assume that the disk cache is implemented by several NA disks and that the tape level of the hierarchy is implemented by NA robotic tapes. There are already commercially available RAID-5 NA disks [10, 11] but there are not NA Tape Drives on the market. However, they are expected to become available pretty soon.

2.2 Description of the HIPPI and IPI-3 protocols

The High-Performance Parallel Interface (HIPPI) is a simplex point-to-point interface with a data transfer rate of 800 Mbit/sec or 1600 Mbit/sec based on either a 32 or 64 bit word size. A full-duplex link can be created using two HIPPI links, one in each direction. The complete HIPPI specification can be found in a series of ANSI standards [12, 13, 14, 15]. Figure 2 describes the various layers of the HIPPI protocol stack and their relationship to other relevant protocols. Starting from the bottom, HIPPI-PH describes the mechanical, electrical, and signaling specifications of the physical layer. The HIPPI-SC describes the protocol for control of HIPPI physical layer switches. HIPPI is a point-to-point protocol so switches are needed for sharing of devices without the need for multiple HIPPI adapters on a device. The HIPPI-FP is the framing protocol which defines the format of the packets. Requests for the transfer of large blocks from an application layer protocol are broken into multiple bursts at the HIPPI-FP layer. HIPPI-LE and HIPPI-FC define the mapping to the IEEE 802.2 protocols and Fibre Channel upper-layer protocols. The IPI-3 Disk and Tape define the mapping to the generic command set protocols for disks and tapes.

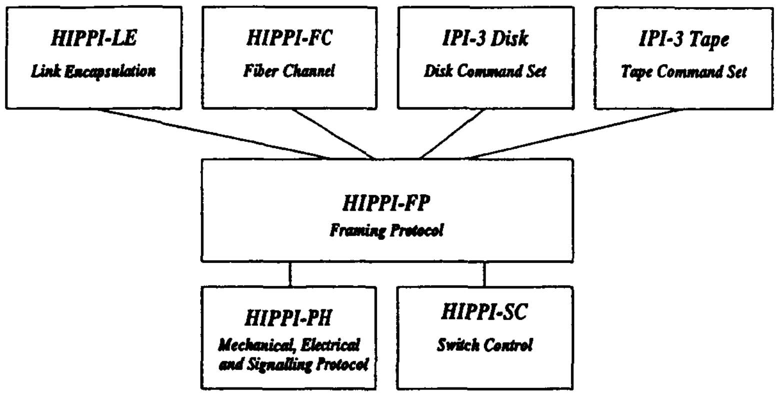

Figure 2: **The HIPPI Protocol Stack**

The HIPPI-PH specification defines the protocol for transfer of data from a source to a destination. Before data can be transferred, the source must first establish a connection with the destination by placing the address of the destination in the bus. This address is used by the HIPPI switches for establishing a connection between a source and a destination. Once the connection has been established, a packet of data can be transferred in the form of multiple bursts. A burst can be from one to 256 words. The source indicates that it is ready to transfer a burst and the destination must send an acknowledgement, thus allowing for flow control. Once the source has transferred all the data, it initiates a disconnect to tear down the connection. Note that a file transfer over a HIPPI switch may imply in many rounds of connect, transfer, and disconnect.

Figure 3 shows the format of a HIPPI frame as defined by the HIPPI-FP. The ULP-id field specifies the upper layer protocol to which the data should be delivered. The P bit when set to zero indicates that the D_1 area is empty. The B bit when set to one indicates that the D_2 area starts at the beginning of the second burst. The D_1 Area Size field defines the size of the D_1 area in bytes and ranges from 0 to 1016 bytes. The Off field specifies the offset of the first byte of data within the D_2 area. This is useful for transferring non-word-aligned data. The D_2 Size field defines the size of the D_2 area in bytes and ranges from 0 to 4GBytes minus 2 bytes. The D_1 area is intended for transferring user control information. By limiting the size of the header plus the D_1 area to 1024 bytes, the command can be transferred to the destination within the first burst and be delivered and processed while the data stored within the D_2 area is still being transferred using additional bursts [16].

The Intelligent Peripheral Interface (IPI-3) is a device independent protocol which defines a generic command set for data transfer to and from magnetic and optical disk drives. The use of a generic interface allows vendors to use a single device driver for communicating with a number of devices. The detailed specification of the standard by the International Standards Organization

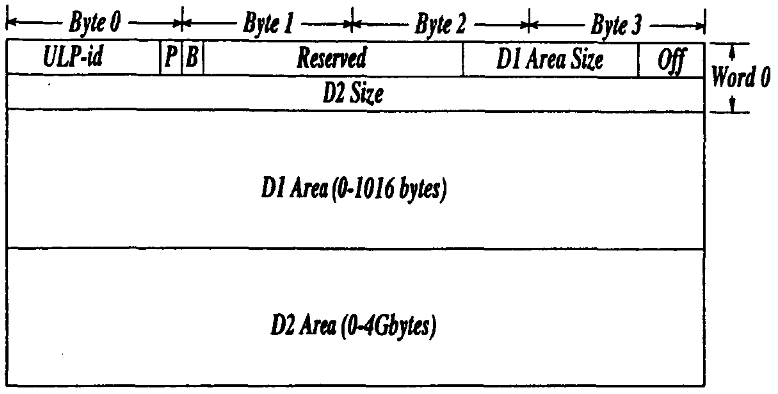

Figure 3: **HIPPI Frame Format**

is described in [17]. As an example, an IPI-3 data transfer from a master to a slave device proceeds in the following sequence. The master sends a command which indicates that this is a read operation. The command contains enough information to uniquely identify the data that needs to be transferred. The bus is then relinquished until the slave is ready to respond. At that time the slave takes control of the bus and sends a transfer notification followed by the data. The sequence of execution of a write command is similar.

2.3 Transfer Protocols

This section describes third-party protocols for executing data transfer transactions in a mass storage system with network attached storage devices. Three protocols will be described based on whether the transfer is device-to-client or device-to-device and whether it is source or sink initiated. A device-to-client protocol is used when data is transferred between the File Server and a NA Disk device, whereas a device-to-device protocol is used when the transfer is between a NA Disk device and a NA Tape device. In all protocols the control messages are sent over the Storage Unit Control Network (SUCN) while the data transfer messages are sent over the High Speed Data Network (HSDN). We make the assumption that, as in most existing sites with network attached devices, the SUCN and the SACN are implemented by a single local area network. Also, the HSDN in this case is a HIPPI network as shown in Fig. 1.

Read from a NA Disk to the File Server Figure 4 will be used to describe the sequence of messages used in the protocol followed to read a file from the disk cache implemented in a NA disk into the File Server.

The sequence of message in this case is as follows:

1. A user sends a read request to the file server over the SUCN.
2. The File Server processes and forwards the request to the NA Disk Mover over the SUCN. This request indicates the direction of data flow (from the NA Disk Mover to File Server), the name of the file to be transferred, the location of the File Server Mover, and that the NA Disk Mover will be the responder and the File Server Mover the initiator.

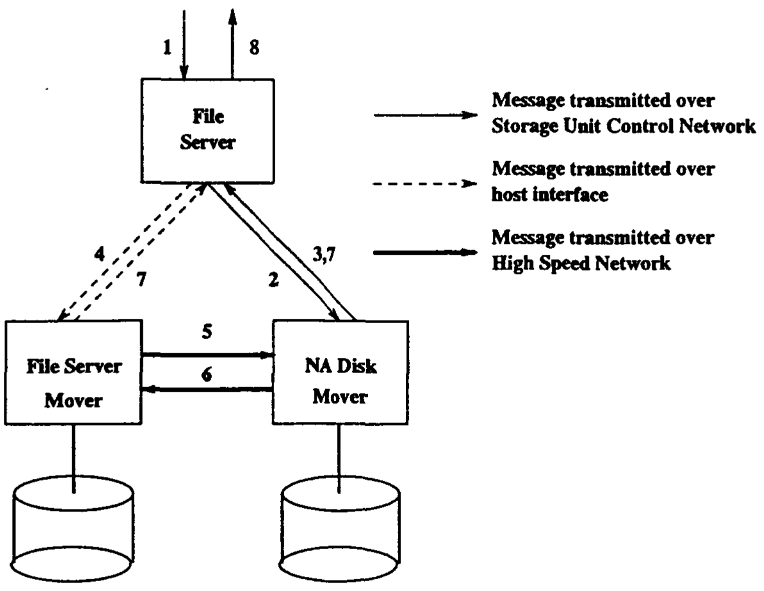
1
8
File Server
Message transmitted over Storage Unit Control Network
Message transmitted over host interface
Message transmitted over High Speed Network
4
7
2
3,7
5
6
File Server Mover
NA Disk Mover

Figure 4: **Message Exchange in a Read Operation Between the File Server and a NA Disk**

3. The NA Disk Mover prepares for the move, assigns a Transfer Identifier (TID) that the File Server Mover will use to identify the transfer, and sends a response back to the File Server over the SUCN containing the TID assigned.

4. The File Server sends a read request to the HA Disk Mover using interprocess communication mechanisms internal to the file server. This read request is an IPI third-party request which indicates the direction of data flow, the amount of data to be transferred, and the TID to be used to identify the transfer.

   Steps 5 and 6 are repeated for each block until all the data has been sent.

5. The HA Disk Mover prepares for the move and then sends an IPI-3 read command message over the HIPPI channel. The command is stored in the D_1 area of the HIPPI frame.

6. The NA Disk Mover when ready, sends an IPI-3 response within a HIPPI-FP frame. The D_1 area of the frame contains the transfer notification message and the D_2 area contains a data block.

7. Once the file has been transferred, both movers send completion messages back to the File Server over the SUCN.

Write from the File Server to an NA Disk When files are stored into the hierarchical mass storage system they are always stored in the disk cache first. Figure 5 is used to describe the sequence of messages that implement a file write from the File Server into a NA disk.

1. A user sends a write request to the file server over the SUCN.

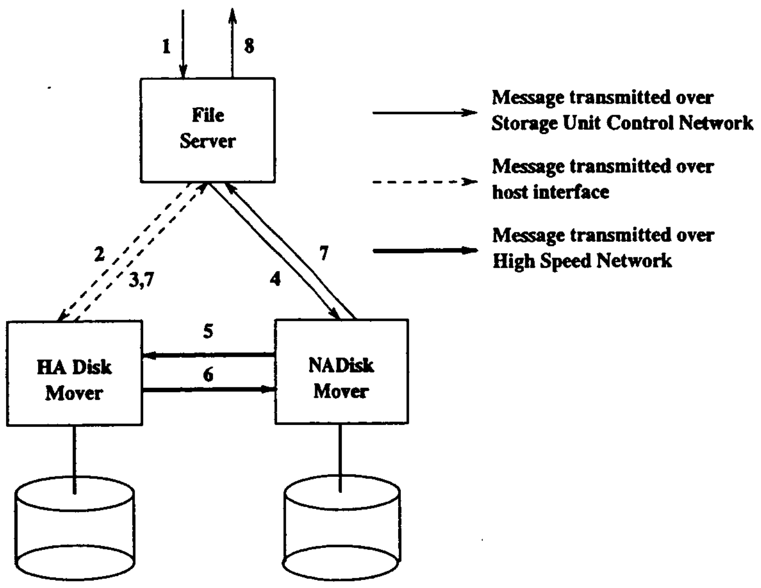
1
8
File Server
2
3,7
4
7
5
6
HA Disk Mover
NADisk Mover
Message transmitted over Storage Unit Control Network
Message transmitted over host interface
Message transmitted over High Speed Network

Figure 5: **Message Exchange in a Write Operation Between the File Server and an NA Disk**

2. The File Server processes and forwards the request to the HA Disk Mover using interprocess communication mechanisms internal to the file server. This request indicates the direction of data flow (from the HA Disk Mover to the NA Disk Mover), the amount of data to be transferred, the location of the NA disk, and that the file server will be the responder.
3. The HA Disk Mover prepares for the move, assigns a TID that the initiator will use to identify the transfer, and sends a response back to the File Server containing the TID.
4. The File Server sends a write request to the NA Disk Mover over the SUCN indicating the direction of data flow, the amount of data to be transferred, the TID to be used to identify the transfer, and that the NA Disk Mover is the initiator and that the HA Disk Mover is the responder.

   Steps 5 and 6 are repeated for each block until all the data has been sent.

5. The NA Disk Mover prepares for the move and sends an IPI-3 read command message over the HIPPI channel. The command is stored in the D_1 area of the HIPPI-FP frame.
6. The HA Disk Mover, when ready, sends an IPI-3 response within an HIPPI-FP frame. The D_1 area of the frame contains the transfer notification message and the D_2 area contains a data block.
7. Once the file has been transferred, both movers send completion messages back to the File Server over the SUCN.

Read from a NA Tape to a NA Disk When a file is not found in the disk cache, it has to be moved from the tape level to the disk cache before it can be retrived by a user. Figure 6 is used to describe the message exchange that takes place when a file is read from a NA Tape to a NA Disk.

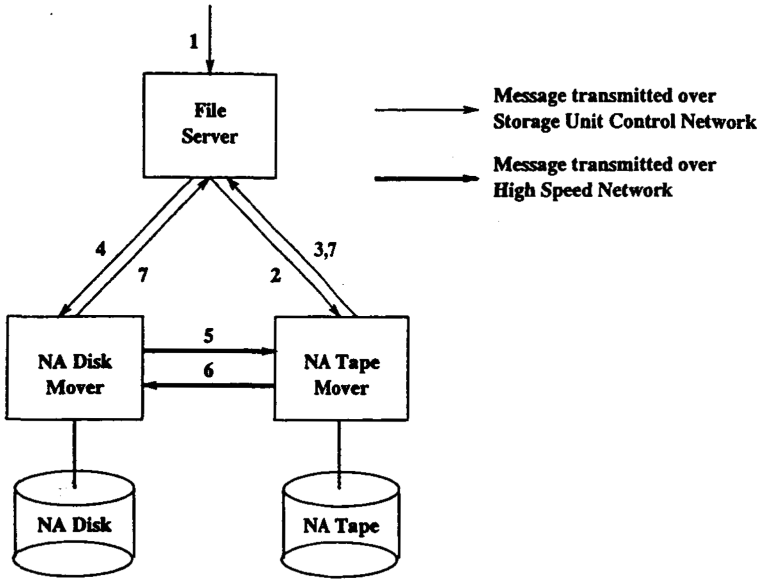
1
File Server
Message transmitted over Storage Unit Control Network
Message transmitted over High Speed Network
4
7
2
3,7
NA Disk Mover
5
6
NA Tape Mover
NA Disk
NA Tape

Figure 6: **Message Exchange in Read from a NA Tape Device to a NA Disk Device**

1. A user sends a read request to the File Server over the SUCN for a file stored in a tape device.
2. The File Server processes and forwards the request to the NA Tape Mover over the SUCN. This request indicates the direction of data flow (from the NA Tape Mover to the NA Disk Mover), the name of the file to be transferred, the location of the NA Disk Mover, and that the NA Tape Mover will be the initiator.
3. The NA Tape Mover prepares for the move, assigns a TID that the NA Disk Mover will use to identify the transfer, and sends a response back to the File Server over the SUCN containing the TID assigned.
4. The File Server sends a request to the NA Disk Mover over the SUCN. This request indicates the direction of data flow (from the NA Tape Mover to the NA Disk Mover), the TID assigned, and that the NA Disk Mover will be the initiator.

   Steps 5 and 6 above are repeated for each block until all the data for the requested file has been sent.

5. The NA Disk Mover sends an IPI-3 read command to the NA Tape Mover over the HIPPI channel. The IPI-3 command resides within the D_1 area of the HIPPI-FP frame and the D_2 area is empty.
6. The NA Tape Mover when ready to transmit a block, sends the block of data in an IPI-3 response. The D_1 area contains the transfer notification message and the D_2 area contains the data block.
7. Both movers send completion messages back to the file server over the SUCN.

3 A Performance Model for NADs

This section describes the performance model used to model hierarchical mass storage systems that use network attached storage devices. We first present the characteristics of the workload used to drive the model, then we discuss the actual queuing network model, and the equations used to compute the service demand parameters for the model.

3.1 Workload Characterization

The workload defined by the authors in a previous study [8, 9] for a host-attached based hierarchical mass storage system was used to drive the queueing network model developed here. The data source for workload characterization was the log of ftp *get* and *put* requests for a period of ten days. First, histograms were generated depicting the number of *get* and *put* requests for each hour of the day. Using these histograms we determined that the interactive load on the system achieved its peak between 9:00 am and 6:00 pm for all histograms considered. This focused our workload characterization to the requests arriving during that period of time. Next, a k-means clustering algorithm was used to determine the appropriate number of classes for each type of request and the file size for each class [18, 19]. The algorithm was initialized using k values uniformly distributed over the range of file sizes and iterated until there was no more shifting of points between clusters. The data for ten days of get and put requests was used for the characterization. Separate data sets were generated for each day and for each of the two types of requests. The algorithm converged after five repetitions on the average for each data set used in our experiments.

As the number of clusters k increases, a better clustering of the points can be determined. On the other hand, as the number of clusters increases the computation time for solving the queueing network also increases since the number of clusters determines the number of classes in the queueing network. In order to find a value of k which both attains a good clustering of the points and at the same time allows for efficient solution of the queueing network, a tightness measure was used. The tightness measure used is described in detail in [8, 9]. Smaller values for the tightness imply a better clustering of the data since the centroids are chosen to be closer to all the points within their corresponding cluster. Even though, in the limit as k approaches N, the tightness goes to zero, the decrease in the tightness is not monotonic. Thus, a value of $k \ll N$ was chosen to achieve a locally minimal value of the tightness. The centroids provide the file sizes which determine the workload classes and the membership of points within a cluster determines the fraction of all requests for a type of that specific size. Table 1 describes the workload selected to drive the model, showing the file size of each class selected and the frequency of occurrence of each class out of a total of 3691 requests during the measurement period. Four classes of get requests and four classes of put requests were chosen. The notation used to refer to the classes for the rest of the paper is g_1 through g_4 for the four get request classes and p_1 through p_4 for the four put request classes.

Table 1: **Workload Characteristics**

Class	File Size in MB	Frequency of Occurrence
g_1	1.2	33.8%
g_2	19.6	9.9%
g_3	78.9	4.2%
g_4	220.6	1.4%
p_1	1.7	42.3%
p_2	34.8	3.3%
p_3	77.7	3.9%
p_4	144.1	1.2%

Table 2: **Per Class Think Times (in seconds)**

	g_1	g_2	g_3	g_4	p_1	p_2	p_3	p_4
Z_r	30.0	96.1	246.7	735.7	23.3	308.5	246.7	881.0

3.2 The Model

Figure 7 shows the queueing network model developed. The model is a closed multiple class queuing network with load dependent devices. Each class in the model is associated with one of the get or put classes described in subsection 3.1. Each dotted block in the figure represents a component of the system. The "Workstations" component represents the user workstations where the requests for file retrieval and storage are generated. They are represented by a delay device which models the think time at user workstations. The "LAN" component represents the Storage Access Control Network (SACN) and the Storage Unit Control Network (SUCN) which are assumed to be implemented by the same Ethernet LAN. The Ethernet is modeled as a load-dependent device since its service rate depends on the load imposed on it [18]. The "File Server" component is modeled by a CPU device and by a host-attached disk device. The "HIPPI" component represents the HIPPI switch which connects the network attached devices with the File Server. The queue in the "HIPPI" component models the contention for the switch which allows a single point-to-point connection between any of the three ports shown in Fig. 1. The last two components, the "NA Disk" and "NA Tape", model the network-attached disk and tape devices respectively. Each NA device is represented by two queuing devices: one represents the CPU where the mover operating system and local file system runs and the other represents the actual NA device.

3.3 Input Parameters

Using the centroids for each class determined in section 3.1, the think time Z_c of class c was measured from the FTP log file and the inter-arrival time was computed using the average between arrivals of the same class for each of the classes considered. Table 2 lists the computed think times for each class of requests. Table 3 describes the input parameters to the model with their values.

Table 3: **Model Input Parameters**

Parameter	Description	Value
$FrameSize$	size of an Ethernet frame	1518 bytes
$TRate_e$	Ethernet transfer rate	10 Mbps
$BlockSize_d$	NA and HA disk block size	8Kbytes
$BlockSize_t$	NA tape block size	16Kbytes
$CtrlSize$	size of a HIPPI-FP frame with an emtpy D_2 area	1024 bytes
$DataSize_d$	size of the D_2 area of a HIPPI-FP frame with a block of data from the disk	$BlockSize_d$
$DataSize_t$	size of the D_2 area of a HIPPI-FP frame with a block of data from the tape	$BlockSize_t$
$TRate_h$	transfer rate of the HIPPI channel	100 Mbytes/sec
$t_{cpu,ftpo}$	average CPU time spent by the ftp daemon in establishing the connection	0.007229 sec
$t_{cpu,fs}$	average CPU time to resolve the file name into a bitfile id and to locate the bitfile id in the file server's table	0.8746 sec
$t_{cpu,ha}$	fixed part of the CPU time at the FS per file transfer	0.00036 sec
$t_{cpu,hb}$	CPU time at the FS per each 64Kbytes transferred	0.0000618 sec
$Seek$	average seek time for the NA and HA disks	12 msec
$Latency$	average latency for the NA and HA disks	6 msec
$TRate_d$	transfer rate for the NA and HA disks	8MBytes/sec
t_{mover}	time needed to process a transfer request by the mover	0.8846 sec
$Mount$	average time for a tape dismount, followed by a mount	71.9 sec
$Seek_t$	average seek time to locate a file in the tape	20 sec
$TRate_t$	transfer rate of the NA tape device	6.0 MB/sec

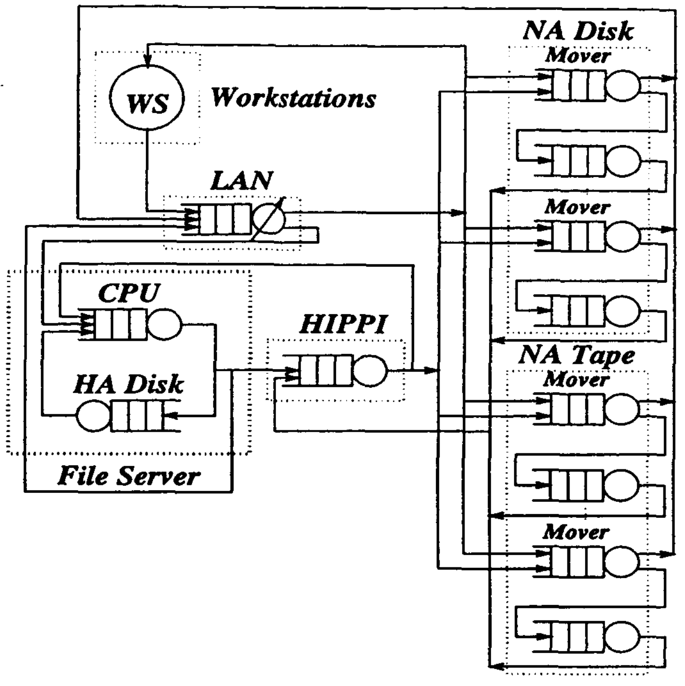

Figure 7: **Queueing Network Model**

3.4 Computation of Service Demands

This section describes the equations used to compute the service demands for each device of the queueing network model using the parameters described in the previous section. We use the notation $D_{i,c}$ to represent the service demand of class c requests at device i.

The service demand at the delay device labeled "Workstations" for class c is equal to the think time Z_c for requests of that class as shown in Table 2.

The service demand, $D_{LAN,c}$, for class c at the Ethernet LAN depends on the class and on the hit ratio, p_h, at the disk cache. This happens because the number of messages transmitted over the LAN depends on which of the procotols described in section 2.3 is used to service the request. Thus, the service demand at the LAN for class $c \in \{1, 2, 3, 4\}$ is:

$$D_{LAN,c} = \{p_h\, NFrames_d + (1 - p_h)\, NFrames_t\} \frac{FrameSize}{TRate_e} \qquad (1)$$

where $NFrames_d$ is the number of frames transmitted when the request is served by an NA disk (disk $NFrames_t$ is the number of frames transmitted when the request has to be served from the NA tape (disk cache miss). Using the protocols described in section 2.3 we get $NFrames_d = 4$ and $NFrames_t = 9$. For a request served by an NA disk, one frame contains the request (Fig. 4 message 1), two frames setup the transfer (Fig. 4 messages 2,3), and one frame indicates completion of the transfer. Notice that messages transmitted internally to the File Server are ignored since the time needed to send them is negligible if compared with the other times involved. For a request served by an NA tape, one frame contains the request (Fig. 6 message 1), three frames setup the transfer (Fig. 6 messages 2,3,4), two frames indicate completion of the NA tape to NA disk transfer

(

(Fig. 6 messages 7), 2 frames initiate the NA disk to HA disk transfer (Fig 4, messages 2,3), and one frame indicates completion of this transfer (Fig. 4 message 7). The service demand $D_{LAN,c}$ at the LAN device for class $c \in \{5,6,7,8\}$ is

$$D_{LAN,c} = NFrames_w \frac{FrameSize}{TRate_e} \tag{2}$$

For a put request, $NFrames_w = 3$: one frame contains the request (Fig. 5 message 1), one frame initiates the transfer (Fig. 5 message 4), and one frame completes the transfer (Fig. 5 message 7).

The service demands of the HIPPI device also depend on the class and the hit-ratio of the NA disk device. This is because the number of messages transmitted depends on the file size of the class making the request and on whether the file needs to be staged to the NA disk before being transferred to the HA disk. The service demand $D_{HIPPI,c}$ at device HIPPI for class $c \in \{1,2,3,4\}$ is

$$\begin{aligned} D_{HIPPI,c} \;=\; & p_h \lceil \frac{FileSize_c}{BlockSize_d} \rceil \frac{2\,CtrlSize + DataSize_d}{TRate_h} + && (3)\\ & (1-p_h)\,[\lceil \frac{FileSize_c}{BlockSize_t} \rceil \frac{2\,CtrlSize + DataSize_t}{TRate_h} + && (4)\\ & \lceil \frac{FileSize_c}{BlockSize_d} \rceil \frac{2\,CtrlSize + DataSize_d}{TRate_h}] && (5) \end{aligned}$$

Part 3 of the service demand equation is the time for an NA disk to HA disk transfer which occurs when there is a disk cache hit. Parts 4 and 5 are the times for an NA tape to NA disk transfer and an NA disk to HA disk transfer respectively, which occur when there is an disk cache miss. The service demand $D_{HIPPI,c}$ at device HIPPI for class $c \in \{5,6,7,8\}$ is

$$D_{HIPPI,c} = \lceil \frac{FileSize_c}{BlockSize_d} \rceil \frac{2\,CtrlSize + DataSize_d}{TRate_h} \tag{6}$$

Note that the service demand equation for put requests does not depend on the hit ratio of the NA disk since files are always written to the disk cache.

The service demand equation $D_{CPU,c}$ at the CPU device for all classes c is

$$D_{CPU,c} = t_{cpu,ftpo} + t_{cpu,fs} + t_{cpu,ha} + t_{cpu,hb} \lceil \frac{FileSize_c}{BlockSize_d} \rceil \, DataSize_d \tag{7}$$

Notice that the hit ratio does not affect the part of the equation that depends on the file size. This is the case since in the case of a miss the NA tape to NA disk transfer is a device to device transfer and does not impose any load on the CPU of the File Server.

The service demand equation for the HA disk device also does not depend on the hit ratio of the NA disk for the same reason described in the paragraph above. The service demand $D_{HAD,c}$ at HA disk device HAD for all classes c is

$$D_{HAD,c} = (Seek + Latency + \frac{BlockSize_d}{TRate_d}) \lceil \frac{FileSize_c}{BlockSize_d} \rceil \tag{8}$$

We make the assumption that one seek and one rotational delay are needed for each block transferred.

The service demand equation for the NA disk mover device (NADM) depends only on the class of the request. The service demand $D_{NADM,c}$ at device $NADM$ for class $c \in \{1,2,3,4\}$ is

$$D_{NADM,c} = p_h\, t_{mover} + 2(1 - p_h)\, t_{mover} = (2 - p_h)\, t_{mover} \tag{9}$$

If the read request is a hit at the NA disk then only one transfer takes place whereas on a miss the file goes from the NA tape to the NA disk, and from the NA disk to the HA disk. The service demand $D_{NADM,c}$ at device $NADM$ for class $c \in \{5,6,7,8\}$ is

$$D_{NADM,c} = t_{mover} \tag{10}$$

since a put request involves a single transfer by the NADM.

The service demand for the NA disk device depends both on the class of the request and on the hit ratio of the device. The service demand $D_{NAD,c}$ at device NA disk for class $c \in \{1,2,3,4\}$ is

$$\begin{aligned} D_{NAD,c} &= p_h\,(Seek + Latency + \frac{BlockSize_d}{TRate_d})\,\lceil \frac{FileSize_c}{BlockSize_d} \rceil + & (11)\\ & \quad 2\,(1 - p_h)\,(Seek + Latency + \frac{BlockSize_d}{TRate_d})\,\lceil \frac{FileSize_c}{BlockSize_d} \rceil & (12) \end{aligned}$$

We assume, as for the HA disk, that for each block transferred we incur one seek and one rotational delay. Part 11 of the equation considers the service demand for a hit, whereas part 12 considers the service demand for a miss which requires both an NA tape to NA disk transfer and an NA disk to HA disk transfer. The service demand $D_{NAD,c}$ at device NA disk for class $c \in \{5,6,7,8\}$ is

$$D_{NAD,c} = (Seek + Latency + \frac{BlockSize_d}{TRate_d})\,\lceil \frac{FileSize_c}{BlockSize_d} \rceil \tag{13}$$

since a write always requires a transfer from the HA disk to the NA disk.

The service demand equation for the NA tape mover device (NATM) depends both on the class of the request and on the hit ratio. The service demand $D_{NATM,c}$ at device $NATM$ for class $c \in \{1,2,3,4\}$ is:

$$D_{NATM,c} = (1 - p_h)\, t_{mover} \tag{14}$$

A request arrives at the NA tape mover only if the file is not located at the NA disk device. The service demand $D_{NATM,c}$ at device $NATM$ for class $c \in \{5,6,7,8\}$ is zero since writes never go explicitly to the NA tape device.

The service demand for the NA tape (NAT) device depends on the hit ratio, the class of the request and on the probability p_{mount} that a request will require a different tape to be mounted from the one currently mounted on the tape device. The service demand $D_{NAT,c}$ at device NAT for class $c \in \{1,2,3,4\}$ is

$$D_{NAT,c} = (1 - p_h)\,[Mount + Seek_t + \frac{BlockSize_t}{TRate_t}\lceil \frac{FileSize_c}{BlockSize_t} \rceil] \tag{15}$$

(

4 Validation

A process oriented simulation model was developed to assess the accuracy of the analytic model using CSim.

more to be added here

5 Numerical Results

more to be added here

5.1 Scenario Description

more to be added here

5.2 Performance Results

more to be added here

6 Concluding Remarks

more to be added here

References

[1] Samuel S. Coleman, Richard W. Watson, Robert A. Coyne, and Harry Hulen, "The Emerging Storage Management Paradigm", in *12th IEEE Symposium on Mass Storage Systems*, Monterey, California, 1993, pp. 101–110.

[2] William Collins, James Brewton, Danny Cook, Lynn Jones, Kathleen Kelly, Lynn Kluegel, Dan Krantz, and Cheryl Ramsey, "Los Alamos HPDS: High-Speed Data Transfer", in *12th IEEE Symposium on Mass Storage Systems*, Monterey, California, 1993, pp. 111–118.

[3] Robert Hyer, Rich Ruef, and Richard W. Watson, "High-Performance Data Transfers Using Network-Attached Peripherals at the National Storage Laboratory", in *12th IEEE Symposium on Mass Storage Systems*, Monterey, California, 1993, pp. 275–284.

[4] Randy H. Katz, "High-Performance Network and Channel Based Storage", *Proceedings of the IEEE*, vol. 80, no. 8, pp. 1238–1261, August 1992.

[5] Convex Computer Corporation, *Unitree++ System Administration Guide, First Edition*, Convex Press, Richardson, Texas, 1993.

[6] Danny Teaff, Dick Watson, and Bob Coyne, "The Architecture of the High Performance Storage System (HPSS)", in *Proceedings of the Goddard Conference on Mass Storage & Technologies*, College Park, Maryland, March 1995.

[7] Richard W. Watson and Robert A. Coyne, "The Parallel I/O Architecture of the High Performance Storage System (HPSS)", in *14th IEEE Symposium on Mass Storage Systems*, Monterey, California, September 1995, pp. 27–44.

[8] Odysseas I. Pentakalos, Daniel A. Menascé, Milt Halem, and Yelena Yesha, "An Approximate Performance Model of a Unitree Mass Storage System", in *14th IEEE Symposium on Mass Storage Systems*, Monterey, California, September 1995, pp. 210–224.

[9] Odysseas I. Pentakalos, Daniel A. Menascé, Milt Halem, and Yelena Yesha, "Analytical Performance Modeling of Hierarchical Mass Storage Systems", Tech. Rep. TR-CS-95-XX, University of Maryland Baltimore County, 1995.

[10] Bob Ciotti, "RAID Integration on Model-E IOS", Work Done at Numerical Aerodynamic Simulation, NASA Ames Research Center, 1994.

[11] Chris Wood, "Client/Server Data Serving for High-Performance Computing", in *14th IEEE Symposium on Mass Storage Systems*, Monterey, California, September 1995, pp. 107–119.

[12] ANSI X3.210-1992, *High-Performance Parallel Interface-Framing Protocol (HIPPI-FP).*

[13] ANSI X3.218-1993, *High-Performance Parallel Interface-Encapsulation of ISO 8802-2 (IEEE Std 802.2) Logical Control Protocol Data Units (HIPPI-LE).*

[14] ANSI X3.183-1991, *High-Performance Parallel Interface-Mechanical, Electrical, and Signalling Protocol Specification (HIPPI-PH).*

[15] ANSI X3.222-1993, *High-Performance Parallel Interface-Physical Switch Control (HIPPI-SC).*

[16] Don E. Tolmie, "High-Performance Parallel Interface (HIPPI)", in *High Performance Networks: Technology and Protocols*, Ahmed N. Tantawy, Ed., pp. 131–156. Kluwer Academic Publishers, Boston, 1994.

[17] ISO/IEC 9318-3, *Information Technology-Intelligent Peripheral Interface Part 3: Device Generic Command Set for Magnetic and Optical Disk Drives.*

[18] Daniel A. Menascé, Virgilio A. F. Almeida, and Larry W. Dowdy, *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.

[19] Leonard Kaufman and Peter J. Rousseeuw, *Finding Groups in Data*, John Wiley & Sons, Inc., New York, NY, 1990.

NASA

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Performance Modeling of Network-Attached Storage Device Based Hierarchical Mass Storage Systems		5. Report Date 6. Performing Organization Code
7. Author(s) Daniel A. Menasce and Odysseas I. Pentakalos		8. Performing Organization Report No. 10. Work Unit No.
9. Performing Organization Name and Address University of Maryland Baltimore County 302 Administrative Building, 1000 Hilltop Circle Baltimore, Maryland 21250-5394		11. Contract or Grant No. NAS5-32337 USRA subcontract No. 5555-40
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001 NASA Goddard Space Flight Center Greenbelt, MD 20771		13. Type of Report and Period Covered Final October 1994 - September 1995 14. Sponsoring Agency Code

15. Supplementary Notes

This work was performed under a subcontract issued by
Universities Space Research Association
10227 Wincopin Circle, Suite 212
Columbia, MD 21044 Task 52

16. Abstract

Network attached storage devices improve I/O performance by separating control and data paths and eliminating host intervention during the data transfer phase. Devices are attached to both a high speed network for data transfer and to a slower network for control messages. Hierarchical mass storage systems use disks to cache the most recently used files and a combination of robotic and manually mounted tapes to store the bulk of the files in the file system.

The paper shows how queuing network models can be used to assess the performance of hierarchical mass storage systems that use network attached storage devices as opposed to host attached storage devices. Simulation was used to validate the model. The analytic model presented here can be used, among other things, to evaluate the protocols involved in I/O over network attached devices.

17. Key Words (Suggested by Author(s)) hierarchical mass storage systems		18. Distribution Statement Unclassified--Unlimited	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 1	22. Price

NASA Form 1626 Oct 86