

Paradise Final Report

July 1, 1993 - June 30, 1996

Contract Title: Paradise - A Parallel Information System for EOSDIS
Contract Number: 57144
Reporting Period: 7/1/93-6/30/96
Names: David J. DeWitt
Phone Numbers: 608-263-5489
E-Mail Addresses: dewitt@cs.wisc.edu
Institution: University of Wisconsin
World-Wide Web Home Page: //www.cs.wisc.edu/paradise/

Project Background

Along with a donation from IBM, funding from this grant was used to begin the Paradise project. The goal of Paradise is to build a scaleable database system for storing, browsing and reprocessing EOSDIS data sets. The CESDIS grant was critical in getting the project and paved the way to securing other funds from NASA from the AISRP and MTPE programs. Recently ARPA has become interested in Paradise and will almost certainly continue funding the project in the future. While the total dollar amount of the CESDIS funding was relatively modest, it was absolutely critical in getting the project going.

Paradise Overview

We began the Paradise project in January 1993 to explore the application of parallel database system technology, such as that developed as part of the Gamma [DeWi90] project, to prototype a scalable database system for storing, browsing, and reprocessing EOSDIS data sets. When fully deployed, the EOS (Earth Observing System) will encompass 50 instruments on 10 satellites observing the Earth's environment. The expected data stream from these instruments will have a collective rate of about 4 Mbytes/second. While this rate, in itself, is not that high, it amounts to a couple of terabytes of new data every day and tens of petabytes over the 10 year lifetime of the instruments. To us, this effort was a natural candidate for parallel database system technology.

Paradise is based on an object-relational data model. In addition to the standard attribute types such as integers, floats, and strings, Paradise also provides a set of spatial data types including point, polygon, polyline, swiss-cheese polygon, and circle. All types in the system are implemented as abstract data types (ADTs) and the spatial data types provide a rich set of spatial operators (e.g. overlap) that can be accessed from an extended version of SQL.

In addition to a rich set of spatial data types, Paradise provides several additional data types that are designed to facilitate the storage and retrieval of images such as those that will be produced by the EOSDIS project. Three types of 2-D raster images are supported: 8 bit, 16 bit, and 24 bit. The 8 bit image type has an associated color palette. The final data type is a N-dimensional array data type in which one of the N dimensions can be varied.

The mixing of spatial, image, and conventional data types inside a single database system provides a number of unique capabilities. For example, one can use a query to select a set of tuples with certain attribute values (name= "Andrew" and date between 9/14 and 9/15), clipping the image attribute of each tuple by a bounding box corresponding to a circle centered on Miami with a radius of 100 miles.

The released version of Paradise employs a conventional client-server architecture. The Paradise front-end provides a graphical user interface that supports querying, browsing, and updating of Paradise objects through either its graphical or textual interfaces. The graphical front-end is implemented using Tcl/Tk [Oust91]. The Paradise server is implemented as a SHORE Value Added Server (VAS) directly on top of the SHORE Storage Manager [Care94]. To the basic SHORE server, Paradise adds a catalog manager, an extent manager, a tuple manager, a query optimizer and execution engine, and support for point, polyline, polygon, and raster ADTs. In addition, SHORE was extended to include R*-trees as a spatial access method (in addition to the existing B-tree mechanism already provided by SHORE).

To facilitate the transparent, but efficient, handling of collections of tuples containing large raster images or N-dimensional arrays, Paradise incorporates several performance optimizations. First, when a tuple containing a raster image (or array) is stored into the database, as illustrated by Figure 1, the image (or array) is decomposed into regular rectangular shaped regions called "tiles". The data in each tile is compressed using the basic LZW compression algorithm and then is stored as a separate SHORE object. A map table (one per raster image) is used to maintain the correspondence between each tile object and its region of the raster image. Decomposition of the raster image into tiles allows Paradise to fetch (from secondary or tertiary storage) only those tiles that are required to execute an operation; for example, when a raster image is clipped by a polygon.

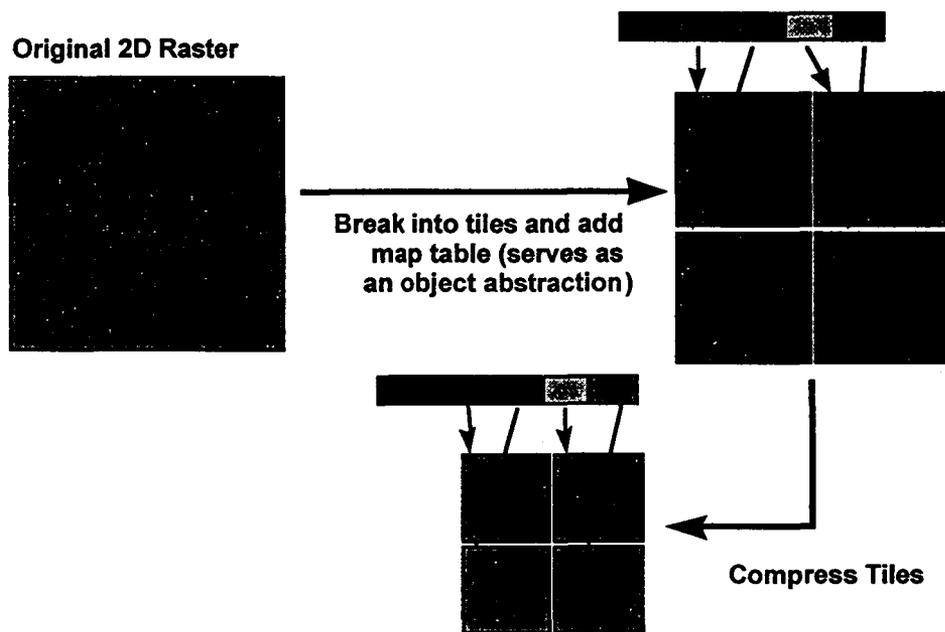


Figure 1: Tiling a Raster Image during the Ingest Process

An advantage of this approach is that the objects containing the actual raster images can transparently migrate between secondary and tertiary storage. In addition, by storing the raster images as large objects in a separate SHORE file, the tuples in the table remain physically clustered with one another, significantly improving the performance of sequential scan over the table. Finally, even for queries involving the raster attribute, the raster images need not always be brought into memory. For example, consider a clip operation between a polygon and a raster attribute. To determine whether an object satisfies the predicate we only need to check the bounding box information stored in the raster header part of the tuple. Even if the tuple does satisfy the clip predicate the raster images are fetched "lazily" only when the image actually needs to be manipulated or displayed.

Current Status

The client-server version of Paradise has been released to a number of sites. [DeWi94] contains more details about the project as well as a performance evaluation of the client-server version of the system on the Sequoia Benchmark [Ston93]. More information about the Paradise project can be found at URL <http://www.cs.wisc.edu/paradise/>.

We are currently working on a number of implementation and research tasks including parallelization, HDF support, and integrated support for tertiary storage.

Parallelization

Recently we completed the initial parallel implementation Paradise and got it working both on a 16 node SP2 that IBM donated to the project and a cluster of 20 dual processor PCs that Intel donated to the project. This cluster is interconnected using 100 Mbit/second Ethernet and a CISCO Catalyst 5000 switch. This effort involved a major rewrite of the client-server version of the system. First, each operator process was redesigned (and re-implemented) to take each of its inputs from an input stream and send its output to an output stream. This pipelined approach to query processing makes it simple to connect operators running on the same or different processors in a fashion that is transparent to the operator. Next, the overall software structure of the Paradise was re-architected. Instead of a single, multithreaded process that performs all functions associated with query execution, the new architecture consists of a master process that is responsible for optimizing and compiling queries plus a slave process on each processor of the cluster or multiprocessor. After a query has been optimized and compiled, the master process walks the execution plan, initiating operators on each of the slave processors. Third, we implemented a communications infrastructure that enables operators executing on different processors to communicate with one another.

We are currently adding support for the parallel manipulation of rasters/arrays, all of Paradise's geospatial types including polygons, polylines, and points, plus video. Users of a spatial database system frequently need to combine two inputs based on some spatial relationship - for example, a user might want to find all rivers that overlap with some landuse polygons. This operation, called a spatial join, can be very expensive and efficient algorithms for evaluating it are required. As part of handling spatial data in Paradise, we have developed a new spatial join algorithm called PBSM (Partition Based Spatial-Merge), which partitions large inputs into manageable chunks, and joins them using a computational geometry based plane-sweeping technique. A novel spatial partitioning function has been developed as part of this algorithm. [Pate96] contains a performance comparison of PBSM with existing spatial join algorithms demonstrates the advantages of PBSM, especially in cases when neither of the inputs to the join has an index on the joining attribute. A copy of the paper can be found on the Paradise web site as well as in the proceedings of the 1996 SIGMOD conference.

For the upcoming SIGMOD conference we are planning on benchmarking the parallel version of Paradise using 10 years of AVHRR data acquired from NASA Goddard.

HDF Support

To simplify the task of using Paradise for those who do not "speak" SQL, we implemented an HDF-compatible, call-level interface to Paradise. Two major extensions were needed. First, we extended Paradise's type system by adding support for 8 and 24 bit raster images as well as multidimensional arrays. Each of these three types are a standard Paradise base type and thus can be used like any other type (int, float, string, polygon, etc.) when defining a relation. Since HDF's concept of V-data is analogous to the concept of a relation in a relational database, nothing new was necessary to support this construct.

The second extension involved modifying the HDF library to replace that portion that deals with reading and writing HDF files with calls to Paradise instead. To use the Paradise version of HDF, all one has to do is to relink an HDF application with the Paradise version of the HDF library. At run-time when the application makes an HDF call, the call is converted to a database query that gets shipped to the Paradise server for execution. As tuples are returned by the Paradise server to the application process, they are converted by the Paradise HDF Library to the format expected by the HDF API Library. This process is illustrated in Figure 2.

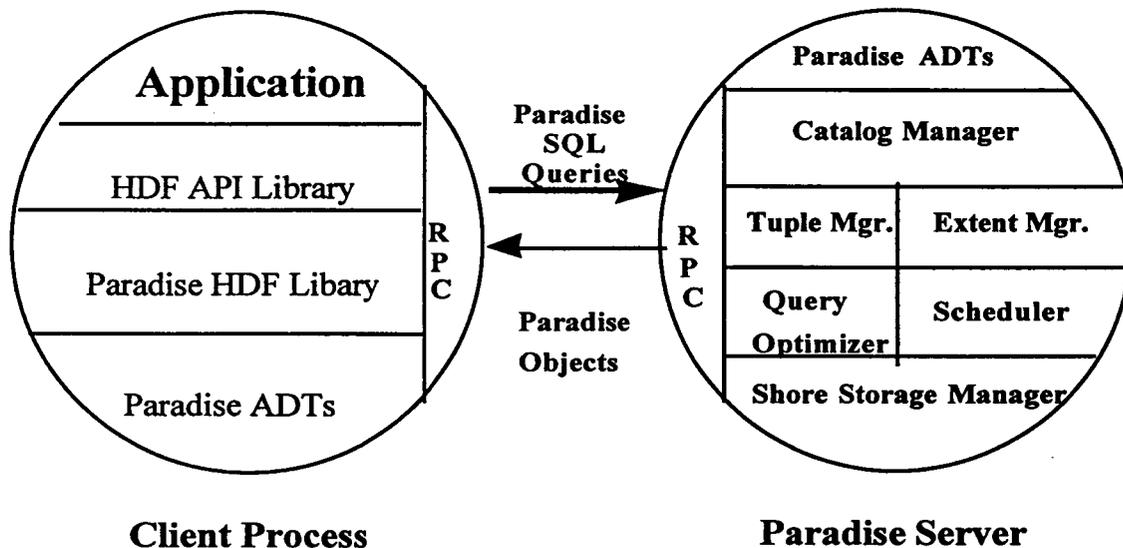


Figure 2: Operation of the Paradise HDF Library Interface

The compatibility of this mechanism was tested by taking a copy of NCSA Collage and relinking it with the Paradise HDF library. As we had hoped, this was done without recompiling or modifying Collage. At the February 1996 NASA meeting for MTPE grant awardees, we demonstrated Collage running on top of Paradise. We are in the process of conducting a through benchmark of HDF on top of Paradise, comparing it with the standard version of HDF as distributed by NCSA.

Integrated Support for Tertiary Storage

A key benefit of taking a DB-centric approach to EOSDIS is that the database system can manage migration of data from tertiary storage to secondary storage as an extension of its existing mechanisms for migrating data from secondary storage to primary storage. To support tape-based tertiary storage we extended the Shore Storage Manager to support the DLT 4700 drive. Since DLT tapes cannot be updated in place, SHORE builds a log-structured file system on the tape. When a block on tape is first referenced it is initially cached in a memory resident buffer pool. This buffer pool is managed on an LRU basis. Tape blocks that are removed from the buffer pool by the LRU policy are buffered in a disk cache in case they are subsequently re-referenced. When a block is updated, the updated block is appended to the logical 'end' of the tape followed by a new directory block.

While modern tape technology such as the Quantum DLT 4700 is dense and relatively fast, a typical tape seek still takes almost a minute! Our solution is two pronged. First, we employ a novel query execution paradigm that we term *query pre-execution*. The idea of pre-execution grew from the experimental observation that queries which accessed data on tape were so slow that we could actually afford to execute the query twice! During the pre-execution phase, Paradise

executes the query normally except when a reference is made to a block of data residing on tape. When such a reference occurs, Paradise simply collects the reference without fetching the data and proceeds with the execution of the query. Once the entire query has been "pre-executed", Paradise has a very accurate reference string of the tape blocks that the query needs. Then, using a *cache-conscious tape scheduling algorithm*, which reorders the tape references to minimize the number of seeks performed, the query is executed normally. While the idea of query pre-execution sounds impractical for a disk-based system, we demonstrate that it actually works very effectively when dealing with large raster images on tape.

The second major technique that we employ to make query processing on tape efficient is termed *query batching*. Query batching is a variant of traditional tape-based batch processing from the 1970s and what Gray refers to as a *data pump*. The idea of query batching is simple: dynamically collect a set of queries from users, group them into batches such that each batch uses the same set of tapes, pre-execute each query in the batch to obtain its reference string, merge the reference strings, and then execute the queries in the batch together. The processing of a batch is done essentially in a "multiple instruction stream, single data stream" (MISD) mode. The ultimate goal is to scan each tape once sequentially, "pumping" tape blocks through the queries that constitute the batch as the blocks are read from tape.

We have completed an implementation of these mechanisms in Paradise and have conducted a detailed performance evaluation. A copy of the paper is available from the Paradise web site. It will be submitted to the 1997 SIGMOD Conference.

Grant Publications

[DeWi93] DeWitt, D., J. Luo, J. Patel, and Jie-Bing Yu, "Paradise - A Parallel Geographic Information System," Proceedings of the ACM Workshop on Advances in Geographic Information Systems, Nov. 1993.

[DeWi94] DeWitt, D., N. Kabra, J. Luo, J. Patel, and Jie-Bing Yu, "Client Server Paradise," Proceedings of the 1994 VLDB Conference, Chile, Aug. 1994.

[Pate96] "Partition Based Spatial Merge Join", (Jignesh Patel and D. DeWitt), to appear, Proceedings of the 1996 SIGMOD Conference, Montreal, CA, June, 1996.

[Yu96] "Processing Raster Images on Tertiary Storage: A Study of the Impact of Tile Size on Performance," (JieBing Yu and D. DeWitt), Proceedings of the 1996 NASA Mass Storage Conference, Sept. 1996.

[Yu97] "Query Pre-Execution and Batching in Paradise: A Two-Pronged Approach to the Efficient Processing of Queries in Tape-Resident Data Sets", (JieBing Yu and D. DeWitt), to be submitted to the 1997 SIGMOD Conference, Tucson, Arizona.

Copies of all Paradise publications can be found on the Paradise web site <http://www.cs.wisc.edu/paradise/>

References

[Care94] "Shoring Up Persistent Applications," Carey, M., D. DeWitt, J. Naughton, M. Solomon, et. al., Proceedings of the 1994 SIGMOD Conference, May 1994.

[Carl92] Carlone, R., "NASA's EOSDIS Development Approach," United States General Accounting Office, Feb. 1992.

[DeWi90] D. DeWitt, S. Ghandeharizadeh, D. Schneider, H. Hsiao, A. Bricker, R. Rasmussen, "The GAMMA Database Machine Project," IEEE Transactions on Knowledge and Data Engineering, Vol. 2, No. 1, March, 1990.

[DeWi92] D. DeWitt and J. Gray, "Parallel Database Systems: The Future of Database Processing or a Passing Fad?," Communications of the ACM, June, 1992.

[Oust91] Ousterhout, J., "An X11 toolkit based on the Tcl language," Proceedings of the 1991 Winter USENIX Conference, 1991.

[Ston86] Stonebraker, M., "The Case for Shared Nothing," Database Engineering, Vol. 9, No. 1, 1986.

[Ston93] Stonebraker, M., J. Frew, K. Gardels, and J. Meredith, "The SEQUOIA 2000 Storage Benchmark," Proceedings of the 1993 SIGMOD conference, Washington, D.C. May, 1993.

09-17-1996 10:56

301 286 1777

SODD 933 B2B/W230

P.02

		Report Documentation Page	
1. Report No. FINAL	2. Government Accession No.	3. Recipient's Catalog No.	
4. Title and Subtitle PARADISE--A PARALLEL INFORMATION SYSTEM FOR EOSDIS		5. Report Date	
		6. Performing Organization Code	
7. Author(s) DAVID DEWITT		8. Performing Organization Report No.	
		10. Work Unit No.	
9. Performing Organization Name and Address COMPUTER SCEINCE DEPARTMENT UNIVERSITY OF WISCONSIN-MADISON 1210 W. DAYTON ST. MADISON, WI 53706		11. Contractor Grant No. 57144	
		13. Type of Report and Period Covered 7/1/93-6/30/96	
12. Sponsoring Agency Name and Address CESDIS		14. Sponsoring Agency Code	
		15. Supplementary Notes	
16. Abstract <p>The Paradise project was begun in 1993 in order to explore the application of the parallel and object-oriented database system technology developed as part of the Gamma, Exodus, and Shore projects to the design and development of a scaleable, geo-spatial database system for storing both massive spatial and satellite image data sets. Paradise is based on an object-relational data model. In addition to the standard attribute types such as integers, floats, strings and time, Paradise also provides a set of and multimedia data types, designed to facilitate the storage and querying of complex spatial and multimedia data sets. An individual tuple can contain any combination of this rich set of data types. For example, in the EOSDIS context, a tuple might mix terrain and map data for an area along with the latest satellite weather photo of the area. The use of a geo-spatial metaphor simplifies the task of fusing disparate forms of data from multiple data sources including text, image, map, and video data sets.</p>			
17. Key Words (suggested by Author(s)) EOSDIS, PARALLEL DATABASE SYSTEMS, TERTIARY STORAGE		18. Distribution Statement	
19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price