# 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-

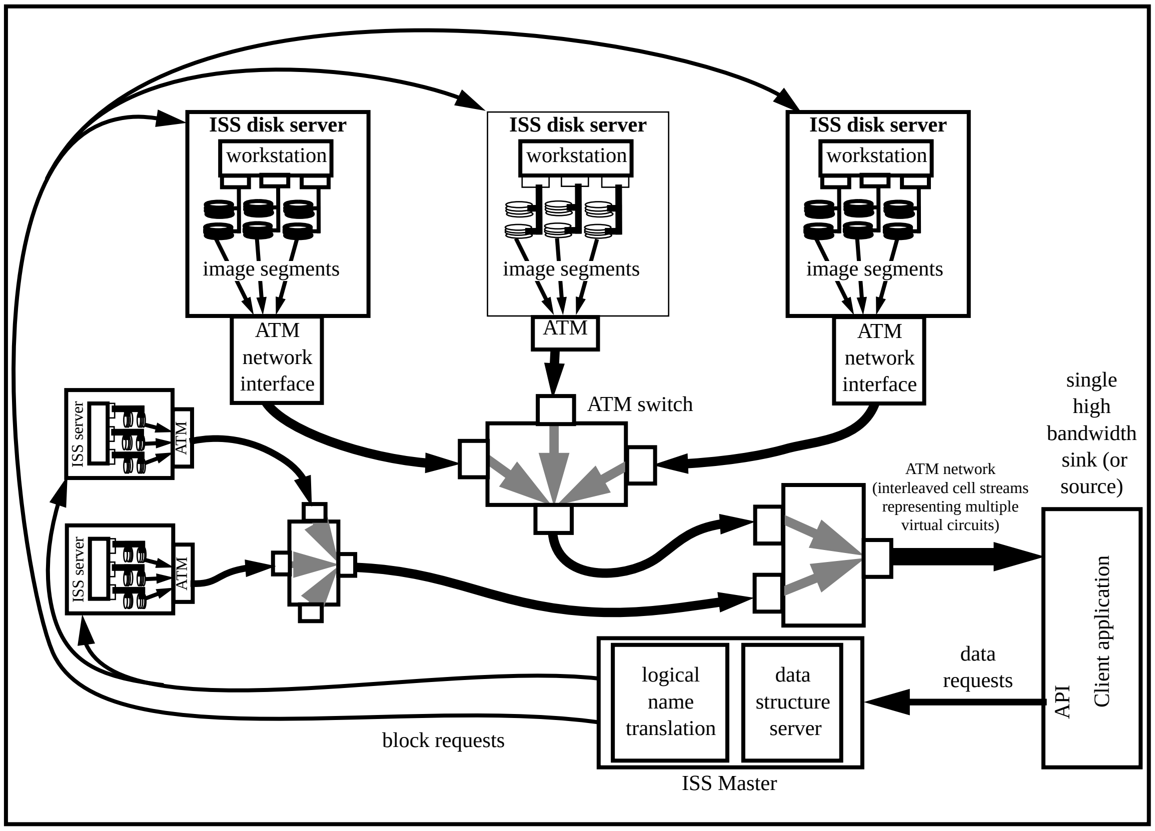

**Figure 1 Distributed-Parallel Server System Architecture**

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 throughput, 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

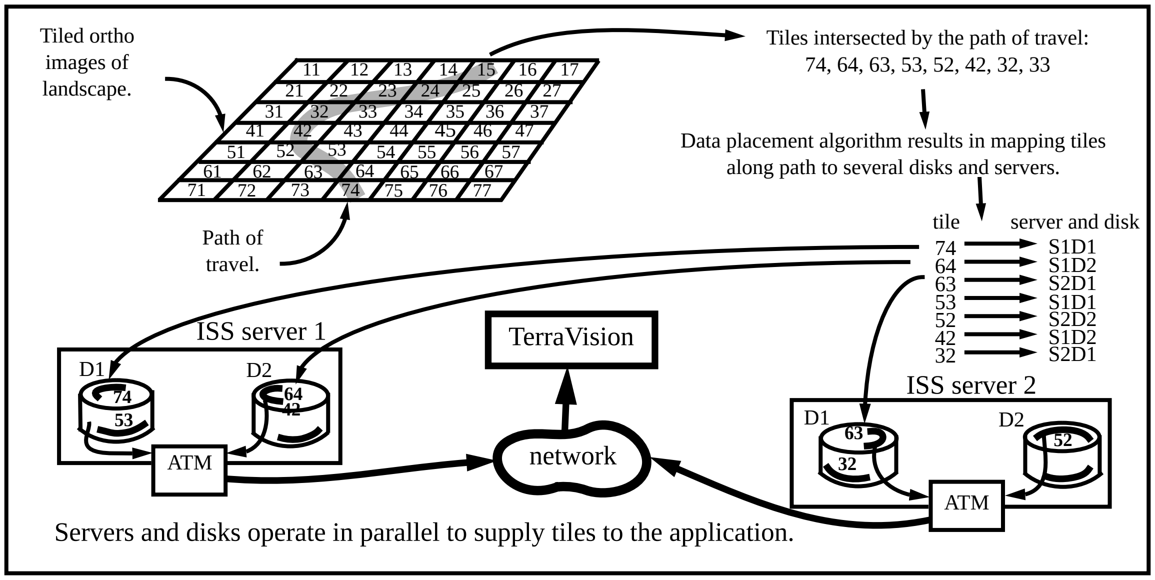

**Figure 2 ISS Parallel Data Access Strategy as Illustrated by the TerraVision Application**

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

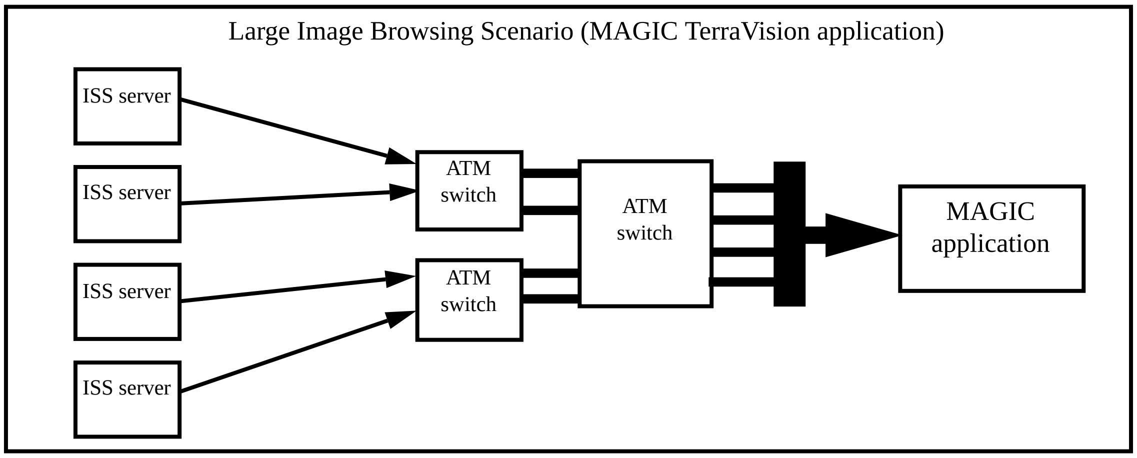

**Figure 3 Use of the ISS for Single High-Bandwidth Application**

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).

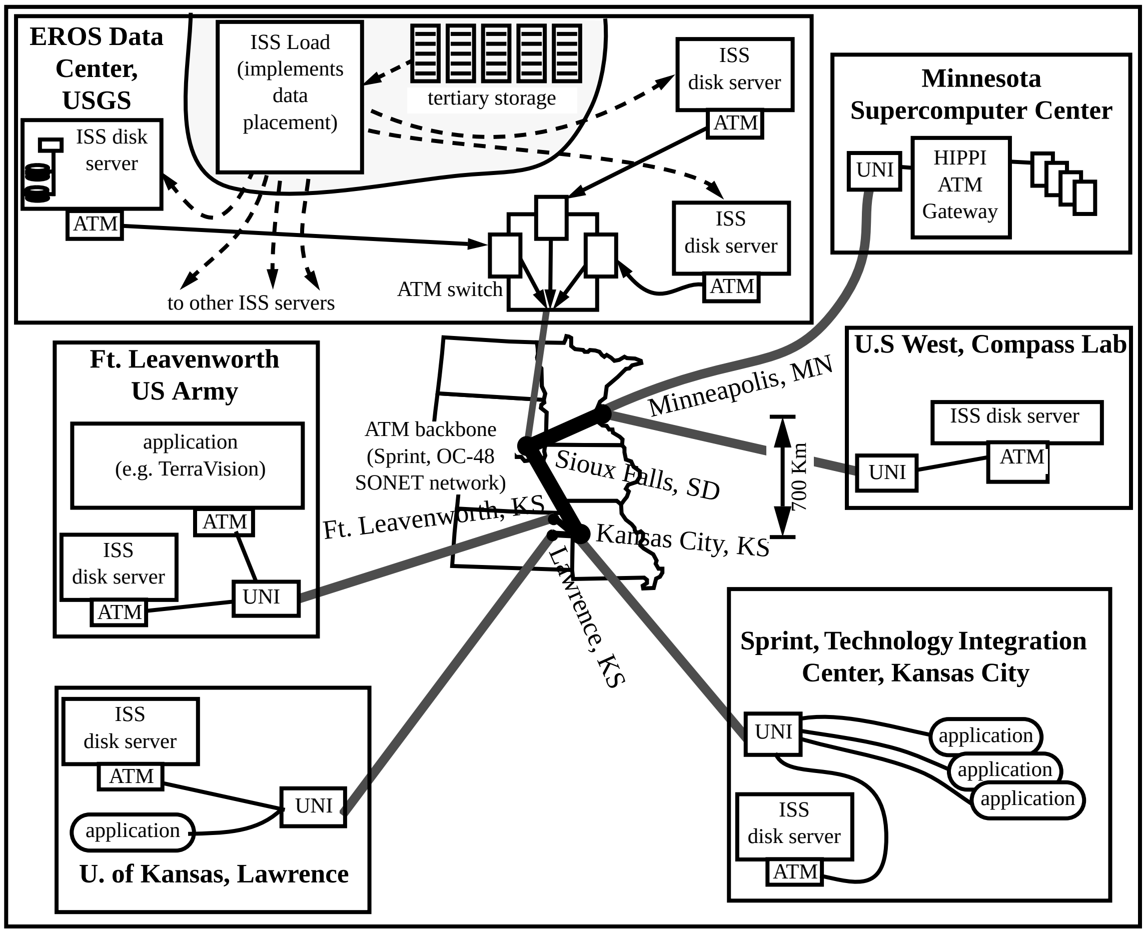

**Figure 4 MAGIC Testbed Application and Storage System Architecture**

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-

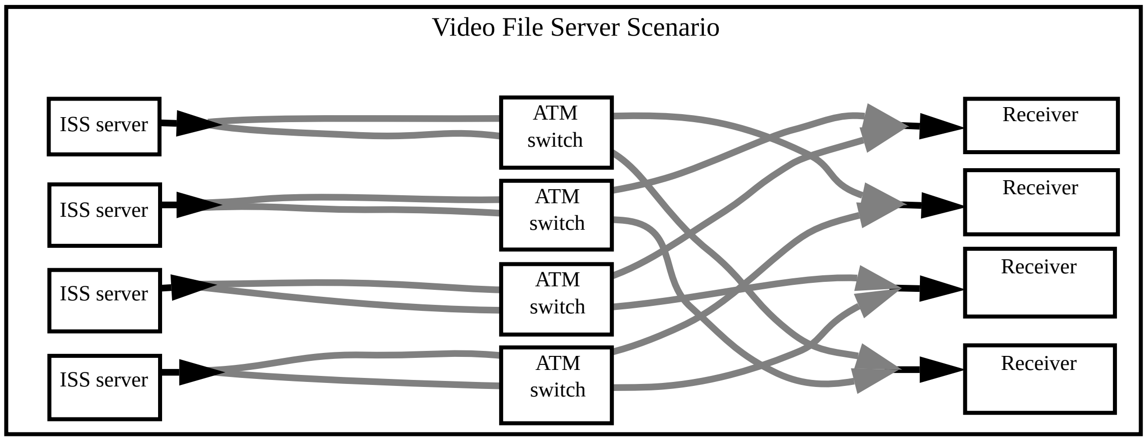

**Figure 5   Use of the ISS to Supply Many Low-Bandwidth Streams**

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

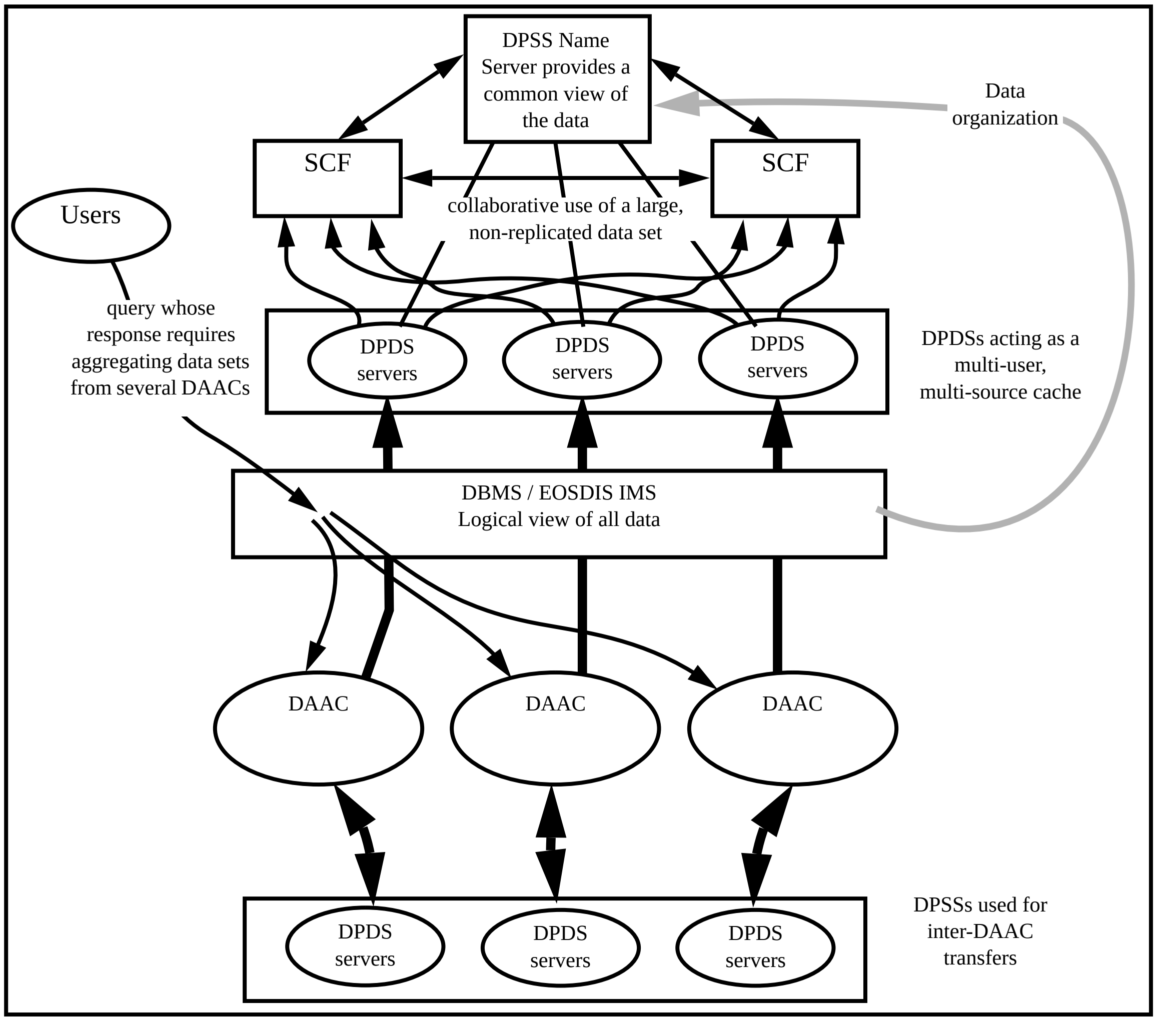
DPSS Name Server provides a common view of the data
Data organization
SCF
SCF
Users
collaborative use of a large, non-replicated data set
query whose response requires aggregating data sets from several DAACs
DPDS servers
DPDS servers
DPDS servers
DPDSs acting as a multi-user, multi-source cache
DBMS / EOSDIS IMS Logical view of all data
DAAC
DAAC
DAAC
DPDS servers
DPDS servers
DPDS servers
DPSSs used for inter-DAAC transfers

**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

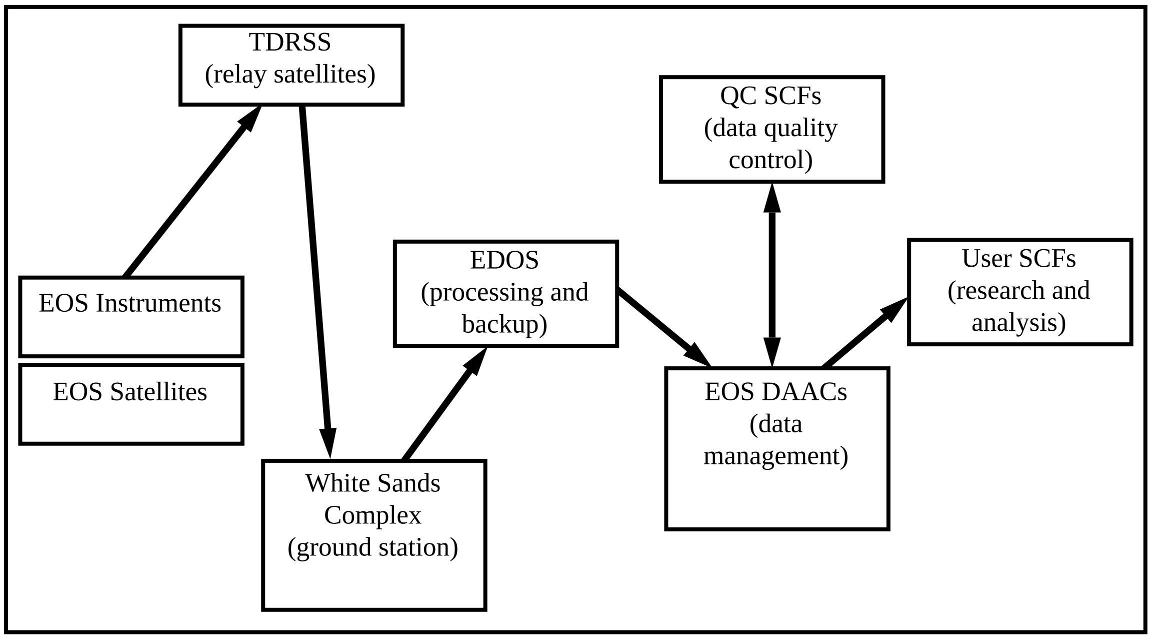
TDRSS (relay satellites)
QC SCFs (data quality control)
EDOS (processing and backup)
User SCFs (research and analysis)
EOS Instruments
EOS Satellites
EOS DAACs (data management)
White Sands Complex (ground station)

**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 .)