LBL--30410

DE91 012293

Building a Mass Storage System for Physics Applications

Harvard Holmes, Stewart Loken

Information and Computing Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

This work was supported by the Director, Office of Energy Research, Office of High Energy and Nuclear Physics, of the U. S. Department of Energy under Contract No. DE-AC03-76F00098.

Building a Mass Storage System for Physics Applications

Harvard Holmes, Stewart Loken

Information and Computing Sciences Division
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

INTRODUCTION

The IEEE Mass Storage Reference Model and forthcoming standards based on it provide a standardized architecture to facilitate designing and building mass storage systems, and standard interfaces so that hardware and software from different vendors can interoperate in providing mass storage capabilities. A key concept of this architecture is the separation of control and data flows. This separation allows a smaller machine to provide control functions, while the data can flow directly between high-performance channels. Another key concept is the layering of the file system and the storage functions. This layering allows the designers of the mass storage system to focus on storage functions, which can support a variety of file systems, such as the Network File System (NFS) [Sand85], the Andrew File System, and others. The mass storage system provides location-independent file naming, essential if files are to be migrated to different storage devices without requiring changes in application programs.

Physics data analysis applications are particularly challenging for mass storage systems because they stream vast amounts of data through analysis applications. Special mechanisms are required, to handle the high data rates and to avoid upsetting the caching mechanisms commonly used for smaller, repetitive-use files. High data rates are facilitated by direct channel connections, where, for example, a dual-ported drive will be positioned by the mass storage controller on one channel, then the data will flow on a second channel directly into the user machine, or directly to a high capacity network, greatly reducing the I/O capacity required in the mass storage control computer. Intelligent storage allocation can be used to bypass the cache devices entirely when large files are being moved.

KEY IDEAS OF THE IEEE MASS STORAGE REFERENCE MODEL

As described in the Mass Storage Reference Model: Version 4 [Cole90]:

> The central architectural features of the reference model and the motivation for them can be summarized as follows:
>
> - An object-oriented description allows the identification of a modular set of standard services and standardized client/server interfaces. The reference model servers are potentially viable commercial products and are building blocks for higher-level services and recursive integration in centralized, shared, or distributed hierarchical storage systems. This integration can be done within single-vendor systems, by third-party, value-added system integrators, or by end-user organizations. The object-oriented modularity hides implementation details, allowing many possible implementations in the support of the standard abstract objects and interfaces [Booc86].
>
> - For the storage system to be integrated with applications and operating systems supporting many different internal file structures, the abstract object visible to storage-system clients is an uninterpreted string of bits and a set of attributes.

- The separation of human-oriented naming from machine-oriented file identifiers allows integration with current and future operating systems and site-dependent naming systems. This implies separation of the name server as a separate module associated with the reference model [Wats81].

- The separation of access rights control as a site-specific module with a standard interface to the storage system accommodates the many operating systems and site-dependent access control mechanisms in existence.

- The separation of the request and data communication paths supports existing practices and the need for third-party control of transfers between two entities by direct data transfers from source to sink, as well as data transfer redirection and pipelining through such data transformations as encryption, compression, and check-summing.

- The separation of the site manager allows site-dependent policies and status to be managed. Provision is made for standard site-management interface functionality.

- Inclusion of a migration server within the file server allows each file server to be self-contained and file-migration policies for each server to be established separately. It also facilitates building a hierarchical storage system supporting automatic migration between servers. The general goal is to cache the most active data on the fastest storage servers and the least active on storage servers with the lowest cost-per-bit medium.

The primary reference model modules, shown in Figure 1, are:

- The *bitfile server*, which handles the logical aspects of bitfile storage and retrieval,

- The *storage server*, which handles the physical aspects of bitfile storage, and

- the *physical volume repository*, which provides manually or robotically retrievable shelf storage of physical media volumes.

Closely related to these modules are:

- The *bitfile client*, which is the programmatic agent of the user required to convert user desires into bitfile requests to the bitfile server and data transfer commands to the bitfile mover,

- the *bitfile mover*, which provides the components and protocols for high-speed data transfer,

- the *name server*, which provides the retention of bitfile IDs and the conversion of human-oriented names to bitfile IDs, and

- the *site manager*, who monitors operations, collects statistics, and establishes policy and exerts control over policy parameters and site operation.

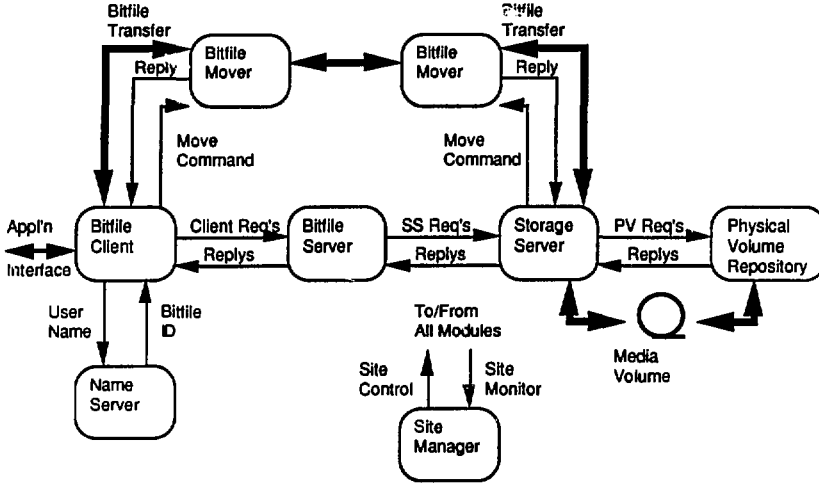

Figure 1. The Storage System Reference Model

This model can be integrated with local file systems via specialized bitfile clients. Both NFS and FTP access are commonly provided; they are called brokers, since they are both servers and clients; servers to the workstations accessing files, and clients to the mass storage system.

Integration with local file systems is a key element in the success of a mass storage system. If the mass storage system appears in the ordinary file space of the user, then he can manipulate files in the mass store using the usual commands; he does not have to learn new commands. More importantly, applications do not have to learn new ways to access files. The key ingredient for this integration is an access point where file access can be diverted from the local file system to the mass store. NFS,the Network File System, provides just this hook into the file system. NFS will divert file access to a server across the ethernet and the mass storage system can respond to these requests via its NFS broker. NFS client software is available for most UNIX systems and also for VMS.

PHYSICS MASS STORAGE APPLICATIONS

Current detectors in High Energy Physics experiments generate enormous amounts of data. A principal challenge of the data collection strategy is to quickly recognize useful data and discard irrelevant data, so that the volume of data collected and the effort expended to analyze it is minimized. Contemporary data collection methods may be viewed as a pipeline of filters, where at each stage the object is to recognize the relevant data and discard the irrelevant data. The first stage operates as part of the detector itself, using triggers and gates to recognize potentially interesting events. Interesting events are then digitized, under the control of multiple microprocessors. The next filter is the microprocessors themselves; they look at the event and make sure that it passes more stringent tests. If the microprocessors select the event, then it is stored in fast memory. At this stage, a conventional workstation or other processor looks at the data in fast memory and does

some preliminary analysis, to decide if the event is sufficiently interesting to be written to tape. If so, it will be written to tape, and then be moved to an off-line phase of event reconstruction and analysis. The off-line phase will result in Data Summary Tapes (DSTs), which are then reproduced and distributed to collaborators at their home institutions. At the home institutions of the collaborators, the DSTs are subjected to multiple rounds of filtering, statistical analysis, modeling and analysis in support of various hypotheses of the individual collaborators. Many collaborations are now using or considering the use of "processor farms," multiple scientific workstations working in parallel, to perform the steps of off-line event reconstruction and analysis.

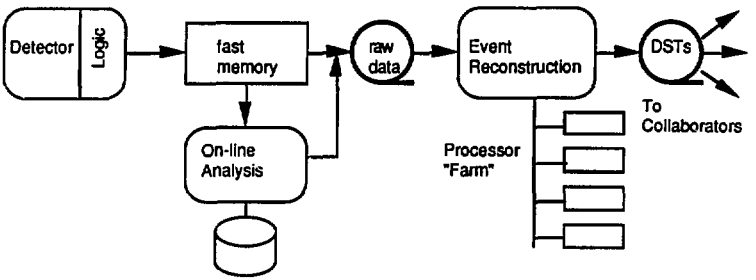

Figure 2. Model of Physics Data Acquisition

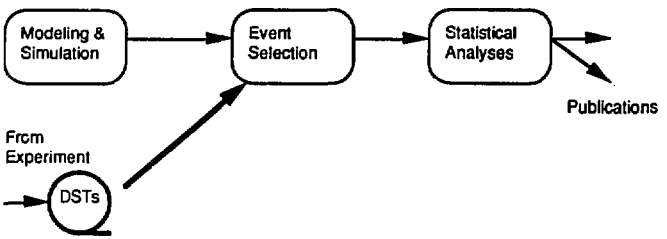

Figure 3. Model of Physics Analyses

The two types of files put on magnetic tape, "raw" data and DSTs, benefit most from a mass storage system. In each case, the major benefits are the automation of tape handling, and the automation of record keeping about the tapes.

In the first case, the off-line reconstruction of events, the requirements of a mass storage system are:

- high bandwidth to absorb events from the detector,
- good performance to large numbers of clients, and
- convenient administrative mechanisms to keep track of the large number of files and users (client machines).

It is at the latter stage, at the home institutions of the collaborators, that a mass storage system can contribute most to the reduction in tape handling, since the DSTs are scanned many times. Here the requirements are similar:

- good performance to large numbers of clients, and
- management of distributed, replicated files across multiple users.

APPLYING THE MASS STORAGE REFERENCE MODEL TO PHYSICS APPLICATIONS

The Mass Storage Reference Model has several characteristics which address these needs of physics applications. To meet the need for high bandwidth, the Reference Model provides distributed operation and separate control and data paths. These features allow the on-line analysis and the event reconstruction modules to run portions of the mass storage system locally, and to support direct channel connections between the fast memory, the tape drives, and the event reconstruction modules. By doing this, the fast memory cannot be blocked or slowed down by any other machines outside of the on-line control and analysis systems. Even if there is no direct, high speed connection between the experimental/data collection areas and other computing facilities, a low speed connection (Ethernet or equivalent) can be used to exchange control information with other components of the mass storage system; this allows names, volume identifiers and other control information to be consistent across all mass storage components. Thus tapes written at the experimental data collection areas will be recognized and correctly treated by other components of the mass storage system.

For physics analysis, the need for high bandwidth arise from multiple passes over DSTs to filter out interesting events for later, more intensive analysis. To meet this need, it may be best to run the filtering module on the same host which supports the mass storage hardware. This prevents the very high I/O of the filtering process from swamping the networks to the clients. Then only the small fraction of events which are interesting are sent over the networks to the clients. Of course, this requires a non-dedicated host for the mass storage system, and it exposes the mass storage system to the usual risks of running user processes on the same processor. This configuration is illustrated in Figure 4.

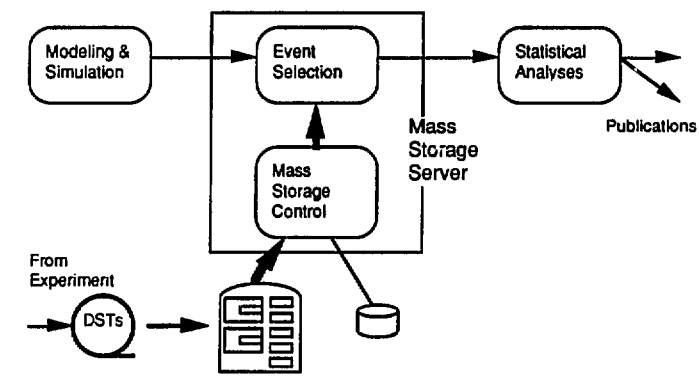

Figure 4. Mass Storage System With Integrated Event Selection

To provide good performance for large numbers of clients, the Reference Model provides distributed operation, which can be used to increase the capacity of the mass storage system in convenient increments. An important performance enhancement is the ability to "pre-stage" files, i.e., ask the storage system to cache a file to disk, in anticipation of its use. This is already

provided by currently available systems. However, another issue which can be critical to the efficient operation of multiple clients is the management of replicated files at multiple locations. This topic is discussed in the section on Unsolved Problems and Future Developments.

To ease the administrative requirements associated with large numbers of files and clients, the Reference Model provides location-independent file naming. Since file names are no longer tied to the physical locations of files, an arbitrarily large name space can be provided. This gives each application the freedom to set up file-naming strategies which are most advantageous to it. At the same time, an application can run on a broad range of clients, since all clients uniformly see the name space of the mass storage system. A second benefit of the location-independence of file names is the ability to move files to their most efficient storage locations without disturbing the operation of applications. This can eliminate an enormous amount of administrative bookkeeping.

The requirement to support distributed, replicated files across multiple users is an issue which is not directly addressed by the Reference Model. The Reference Model supports distributed storage of distinct files, but not replicated files. The issue of replicated files is discussed in the section on Unsolved Problems and Future Developments.

MASS STORAGE CONFIGURATION AT LBL

LBL is currently implementing a mass storage system based on the IEEE Mass Storage System Reference Model. We are using commercial software, the UniTree product from DISCOS, a division of General Atomics. Other products are also commercially available, e.g., Epoch. These products integrate with other systems through the use of NFS and File Transfer Protocol (FTP). We are using a SUN system as the host for the mass storage system. Our user community prefers low price for archival storage over rapid performance; accordingly we intend to use 8 mm helical scan tape drives with a robot loader for our first archival storage unit. Additional units will be selected after we gain experience with the initial configuration. Our initial system will be connected to the Laboratory-wide ethernet; as performance needs increase, the mass storage system will connect to LBL's FDDI backbone, and to FDDI connections to the major machines at LBL.

We intend to charge our users real dollars for storage space, and for access to their files. If we do not charge at least a minimum price, our users will stop keeping track of their files and chaos will result. Charging for space will maintain accountability and force managers to make the proper dispositions of files when projects end or employees leave. Our pricing strategy is intended to encourage archival storage, but to discourage repetitive access. Because the mass storage system has limited i/o and network capacity, we want to encourage users to keep their high use files on disks local to their workstation or file server, rather than the mass storage system. Our current plans are to charge $0.15 per megabyte per day for access and $0.001 per megabyte per month for archival storage. Our access charge is computed by noting that $0.15 per day is about $5.00 per month per megabyte; this is slightly higher than what we charge users for space on our timesharing systems with backup service. Any lower rates would give users an economic incentive to keep files on the mass storage system rather than moving them to their workstations or file servers. Since files that are accessed are cached to disk before being sent to users, we just look at the file space on disk when calculating access charges. We will charge for deleted files that have not been purged from the system, since they are using real resources. If a user wants two archival copies of files, we will charge him for both copies.

UNSOLVED PROBLEMS / FUTURE DEVELOPMENTS

We need to address a number of other issues to make the mass storage system as useful as we think that it should be. These include:

- the support of multiple copies of files for efficiency, with the appropriate locking,

- the ability to protect users against inadvertent deletion of files, and the ability to restore previous versions of files,
- the integration of existing file servers into a comprehensive storage management scheme, and
- the use of the mass storage system to provide secure backup capability, replacing current backup systems.

Many of these problems are interrelated, and over the long term, we expect that they will be solved by the use of specialized file systems running in place of the NFS and FTP brokers. The Andrew File System (AFS), developed at CMU, is a leading candidate for this role. AFS, for example, would use the Reference Model for its raw storage needs, and provide a replicated, robust file system to its clients. Its file system would manage multiple copies of a file, and provide the appropriate locking for its clients. AFS contains its own name server, and would interface to the mass storage system via the machine-oriented bitfile IDs.

Over the shorter term, we intend to use simple extensions to the mass storage system to implement some of these features. One problem, the inadvertent deletion of files, has already been solved by the UniTree system in a fashion similar to the Macintosh file system. This system provides a "trash can" directory, where deleted files are stored for a limited time, allowing the user to simply move his files from the "trash can" back to their correct location. Once they are gone, however, they are truly gone! We intend to charge very little for long term storage, to encourage users to keep everything that they think that they might need in the future.

We plan to provide storage management to file servers and workstations via simple scripts that will run periodically. These scripts will search the local disks of the clients for files which have not been referenced recently. These files will be copied to the mass storage system and the local file deleted. A symbolic link will be made from the local file name to the corresponding file in the mass storage system. This symbolic link will allow applications to continue to execute without change. To undo this process, we intend to examine the log of files requested from the mass storage system, and send a message to the client to retrieve the file and replace it on the local disk. This approach causes the first access to the file to proceed directly from the mass storage system, and subsequent accesses to be satisfied from the local disk. As we gain experience with the mass storage software, we may use the distributed capability of the mass storage system to directly manage storage on the client machine. Of course, we will need to be quite sure of the reliability and performance of the system before we do this.

Replacing the current backup procedures with the mass storage system is not a simple process. There are certain system-level files that we will continue to backup as we do presently. Apart from that, current backup procedures provide protection against failure of a tape by keeping several previous backup versions, and these backup versions are stored remotely from the machine room. To provide equivalent protection in the mass storage system, we may have to record duplicate copies of tapes. If one set of tapes is stored remotely from the machine room, as backup tapes are, then we'll need to keep the local and remote tapes in synch.

CONCLUSION

It is now possible to build an economical mass storage system. In the last year or so, there have been key developments in both the software and hardware areas. The Mass Storage System Reference Model, developed by the IEEE Technical Committee on Mass Storage Systems and Technology, has reached the point where vendors are taking it seriously and promising compatibility. Most vendors are attending the committee meetings. At least two commercial systems are now available. It is no longer necessary to "roll your own" system. On the hardware side, the recent flurry of robotic systems for optical disks and small form factor tapes has made it possible to put together in inexpensive system. Software flexibility and new hardware devices provide convenient, incremental growth paths.

These systems can provide significant benefits to physics applications, in the form of higher performance for applications and reduced administrative work managing the large volumes of storage now characteristic of experimental high-energy physics.

ACKNOWLEDGEMENT

This work was supported by the Director, Office of Energy Research, Office of High Energy and Nuclear Physics, of the U. S. Department of Energy under Contract No. DE-AC03-76F00098.

REFERENCES

[Booc86] Booch, G., "Object-oriented Development," *IEEE Transactions on Software Engineering, SE-12,* 1986, p. 211- 221.

[Cole90] Coleman, S., and Miller, S., eds., Mass Storage System Reference Model: Version 4, *IEEE Technical committee on Mass Storage Systems and Technology,* 1990

[Sand85] Sandburg, R., "Design and Implementation of the SUN network file system," *Proceedings of the tenth Usenix Conference,* 1985, p. 119-130

[Wats81] Watson, R. W., Identifiers (naming) in distributed systems, *Distributed Systems--Architecture and Implementation,* Springer-Verlag, Berlin, 1981, p. 191-210