Development of Secondary Archive System at Goddard Space Flight Center Version 0 Distributed Active Archive Center

Mark Sherman, John Kodis, Jean-Jacques Bedet, Chris Wacker
Hughes STX
7701 Greenbelt Road, Suite 400
Greenbelt, MD 20770
{sherman, kodis, bedet, wacker}@daac.gsfc.nasa.gov
+1-301-441-4285 Fax +1-301-441-2392

Joanne Woytek, Chris Lynnes
NASA/GSFC
Greenbelt Road
Greenbelt, MD 20771
{joanne,lynnes}@daac.gsfc.nasa.gov
+1-301-286-4418

Abstract

The Goddard Space Flight Center (GSFC) Version 0 (V0) Distributed Active Archive Center (DAAC) has been developed to support existing and pre Earth Observing System (EOS) Earth science datasets, facilitate the scientific research, and test Earth Observing System Data and Information System (EOSDIS) concepts. To ensure that no data is ever lost, each product received at GSFC DAAC is archived on two different media (VHS and Digital Linear Tape (DLT)). The first copy is made on VHS tape and is under the control of UniTree. The second and third copies are made to DLT and VHS media under a custom built software package named "Archer". While Archer provides only a subset of the functions available with commercial software like UniTree, it supports migration between near-line and off-line media and offers much greater performance and flexibility to satisfy the specific needs of a Data Center. Archer is specifically designed to maximize total system throughput, rather than focusing on the turn-around time for individual files. The Commercial Off the Shelf Software (COTS) Hierarchical Storage Management (HSM) products evaluated were mainly concerned with transparent, interactive, file access to the end-user, rather than as a batch-oriented, optimizable (based on known data file characteristics) data archive and retrieval system. This is critical to the distribution requirements of the GSFC DAAC where orders for 5000 or more files at a time are received. Archer has the ability to queue many thousands of file requests and to sort these requests into internal processing schedules that optimize overall throughput. Specifically, mount and dismount, tape load and unload cycles, and tape motion are minimized. This feature did not seem to be available in many COTS packages. Archer also utilizes a generic tar tape format that allows tapes to be read by many different systems rather than the proprietary format found in most COTS packages. This paper discusses some of the specific requirements at GSFC DAAC, the motivations for implementing the Archer system, and presents a discussion of the Archer design that resulted.

Introduction

One of the critical components within the DAAC's Data Archive and Distributed System (DADS) is the HSM system. Several years ago, UniTree was chosen as the best candidate

to satisfy the GSFC DAAC 's requirements providing both the basic HSM functions and the device drivers for the planned robotic devices. After months of integration and customization, UniTree reached some stability but it fell short of the GSFC DAAC throughput requirements [1], and was limited in the configurability of the archive, retrieval, and caching systems based on data-specific characteristics; e.g., size, volume, likely reuse, multiple versions, etc. It also became apparent that this product and other similar commercial products were not fully suited for this domain of application.

Archer is an in-house software package that was developed by the GSFC DAAC to provide management of secondary and tertiary backup copies of all datasets stored in the archive. Archer was developed to remedy some of the major drawbacks of HSMs, such as UniTree, in handling a data (vs. file) archival system. In particular its design was kept simple and tailored to handle data requests with large number of files and varying files characteristics. Performance was a key consideration in the design of the system and its highly parallel distributed architecture allows the system to be scaled to much larger archives. This paper starts by presenting an overview of the functionality needed for the GSFC DAAC to be a fully operational Data Center. The overall hardware architecture to meet the needs of the GSFC DAAC is described, followed by a discussion on what led the GSFC DAAC to the development of Archer. The architectural design of Archer is presented with its main features. Finally, the status, lessons learned, and future work are briefly described.

GSFC DAAC functions and architecture

The GSFC DAAC can be viewed as composed of three main components which are a Product Generation System (PGS), an Information Management System (IMS), and a Data Archive and Distribution System (DADS). The PGS and IMS are respectively associated with the production of higher level products and the catalog holdings searched and browsed by researchers. The DADS controls the overall processes of the ingestion of new data and the distribution of data requests. The migration between near-line and on-line devices is handled by both UniTree and Archer, however only Archer has the full capability to migrate media between near-line and off-line. For historical reasons, UniTree is currently responsible for the primary archive. Secondary and a tertiary archives, under the control of Archer, use respectively DLT and VHS as archive media. The Metrum RSS-600 Automated Tape Library (ATL) with 5 RSP-2150 drives and 600 VHS cassettes (for a total capacity of up to 8.7 TB) is shared by UniTree and the tertiary archive. Most tapes in the ATL and four of the five VHS drives are controlled by UniTree. The secondary archive is composed of three DLT 7 cartridge stackers. While UniTree and the tertiary archive are run on an SGI Challenge L, the secondary archive is executed on an SGI Challenge S.

Two SGI 4D/440 workstations are being used to test new version of the DADS, IMS, Archer software and new releases of UniTree. Having dedicated test machines is very important to avoid affecting the day to day operation at the GSFC DAAC. Several SGI machines are also used to process Pathfinder Advanced Very High Resolution Radiometer (AVHHR) land products and to perform Quality Assessment (QA) on new products generated. Figure 1 and 2 and Table 1 illustrate some of main platforms acquired by GSFC DAAC along with their specific functions.

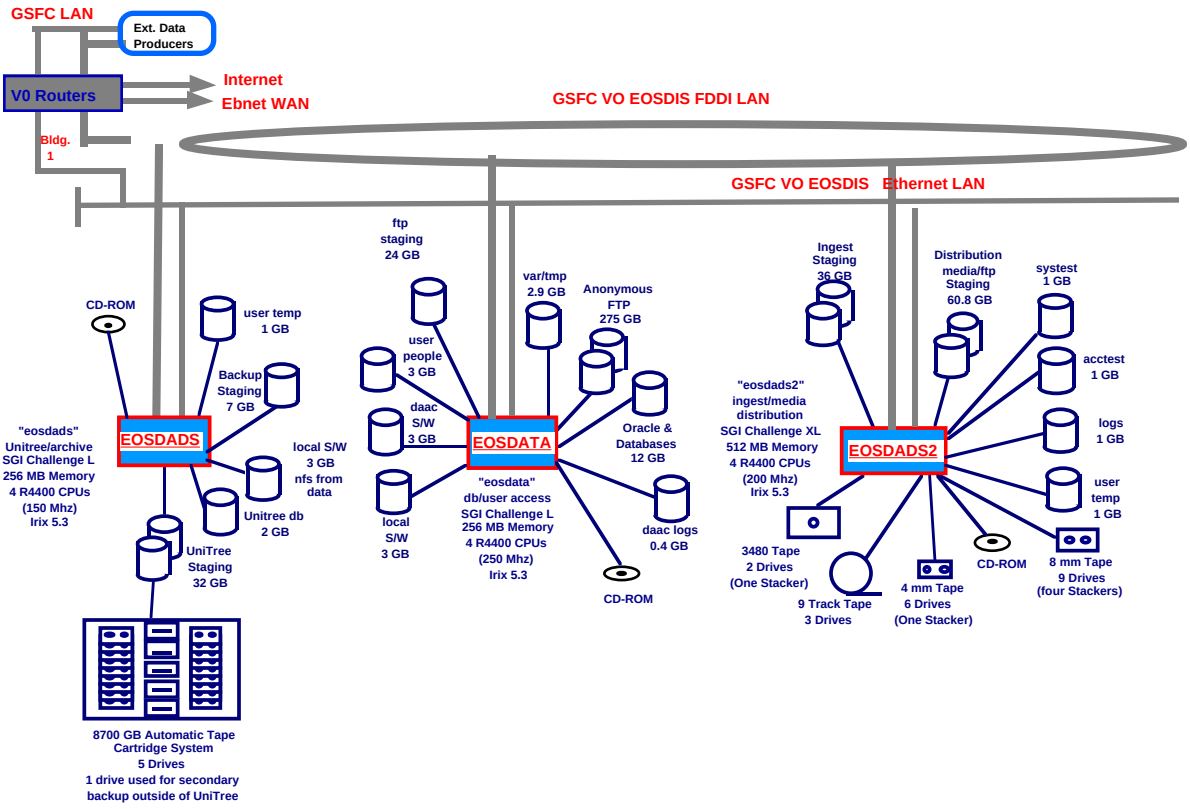

Figure 1 GSFC DAAC 1996 Configuration as of 2/28/96 (1 of 2)

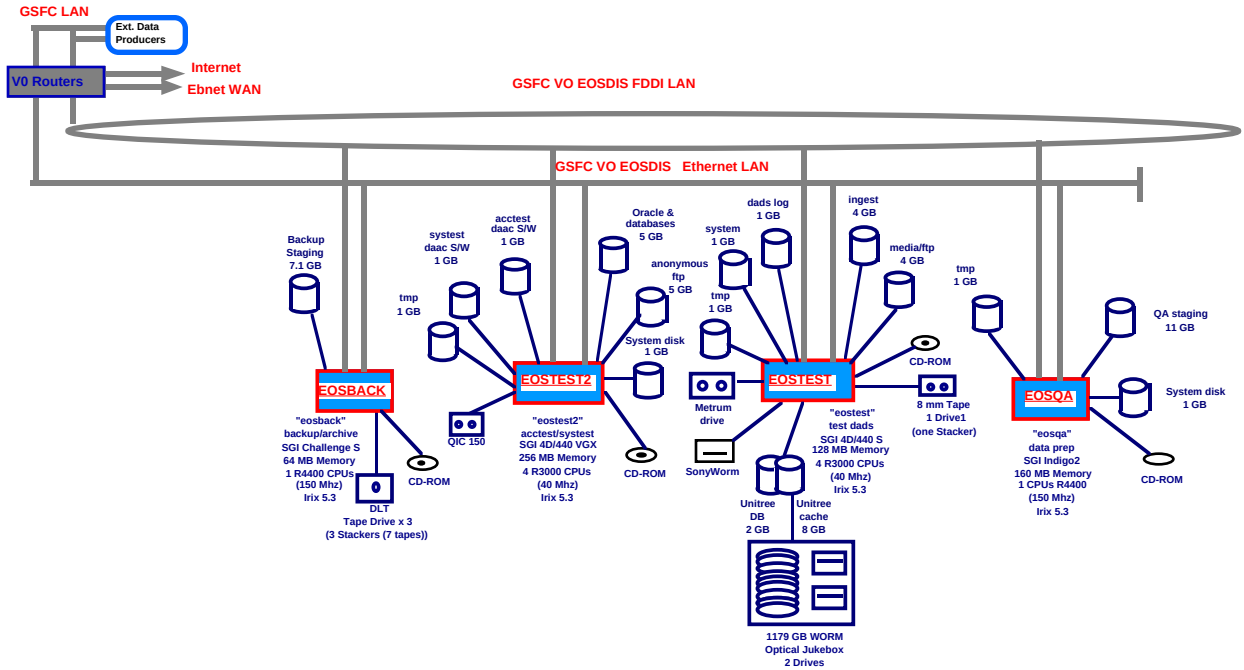

Figure 2 GSFC DAAC 1996 Configuration as of 2/28/96 (2 of 2)

Machine name	Function	Hardware description
EOSDADS	run UniTree & tertiary archive	SGI Challenge L, 256 MB memory 4 R4400 CPUs (150 Mhz) - Metrum RSS600 automatic library - 32 GB UniTree stage disks
EOSBACK	run secondary Archive	SGI Challenge S, 64 MB memory 1 R4400 CPU (150 Mhz) - DLT stackers
EOSDATA	run IMS and Oracle Database	SGI Challenge L, 256 MB memory 4 R4400 CPUs (250 Mhz) - 24 GB ftp stage disks - 275 GB anonymous ftp
EOSDADS2	run ingestion & distribution	SGI Challenge XL, 512 MB memory 4 R4400 CPUs (200 Mhz) - 36 GB ingest staging disks - 61 GB distribution staging disks - 8mm drives - 4mm drives - 3480 drives
EOSTEST2	test software in acctest & systest	SGI 4D/440 VGX, 256 MB memory 4 R3000 CPUs (40 Mhz)
EOSTEST	test dads software & new version of UniTree	SGI 4D/440, 128 MB memory 4 R3000 CPUs (40 Mhz) - 8 GB UniTree cache
EOSQA	run data product QA	SGI indigo 2, 160 MB memory 1 R4400 CPU (150 Mhz)

Table 1. Hardware at the GSFC DAAC

Criteria for the development of a secondary archive

This paper now focuses on issues faced by the GSFC DAAC during the last two years and some of the specific requirements that led to the development of a secondary archive system.

Over the years, the GSFC DAAC has faced problems with the HSM system UniTree and the archive media (VHS tapes and 12" WORM optical platters). In particular, UniTree did not work very well when 12" WORM optical drives were working concurrently with the VHS tape drives. Unitree also did not satisfy the general throughput requirements, and proved difficult to configure based on evolving data characteristics and data request profiles. While some issues have been resolved, others still remain open. Additionally, occasional loss of data due to media failure, UniTree software failures, along with a requirement from the Sea-viewing Wide Field of View Sensor (SeaWiFS) project necessitated the need to keep a second copy of all products. It became apparent that there was an urgent need for a secondary data archive system that would hold a backup copy of all data received at the GSFC DAAC, would take over in case the primary system failed, and if successful in increasing throughput, could be used as a primary retrieval system. At the time UniTree was not fully stable and the GSFC DAAC was under increasing need to provide better, more reliable data retrieval and a robust data recovery capability which did

not rely on the data provider to re-send lost data. The choices were either to purchase a second COTS product or to develop our own secondary data archival system. The data archive system was intended to mostly store data to archive tapes, track file location and tape utilization, and to handle both near-line and off-line tapes. Most COTS packages evaluated were deemed too sophisticated and expensive for the simple set of requirements that had been identified. Further, many of the COTS HSMs , which were oriented towards transparent, interactive file retrieval functionality, did not seem to fully meet these simple requirements. This was particularly true for automatic migration of media between near-line and off-line storage, and large, batch oriented file/data requests. Our experiences with the UniTree COTS package also pointed out other problems with commercial HSMs, such as performance bottlenecks and maintainability issues. For these reasons, the decision was made that the GSFC DAAC would gain by developing its own secondary data archive system. The remainder of this section focuses on some of the criteria that were factored into the secondary archive design.

As mentioned above, UniTree was designed around limited, interactive file access which imposed limitations that were undesirable for a large scale science data center. For instance, UniTree limits the number of concurrent stage operations (around 100) which causes major problems when large number of files are to be staged. Also, the order of requesting and staging data, along with adequate feedback on both successful and unsuccessful retrievals, are critical, both to achieve good performance, and to simplify the media distribution process. For example, a request may need a set of files staged and then copied to a number of 8mm tapes for distribution in the time order in which the data was initially produced. The request would best be handled by staging in the time order to be distributed, particularly if multiple distribution tapes will be needed. Additionally, in a production environment it is not unusual to have unexpected hardware and software problems or unexpected workloads that must be rectified manually. Therefore, it is important to have full control over the archive, letting the system run by itself, but allowing operators to take control of the system when needed. To provide flexibility and adaptability to facilities with the needed requirements and resources, HSMs should have an Application Program Interface (API), which many commercial products either do not provide or provide with very limited capabilities. It would be highly desirable to have standardized APIs to facilitate transition to a new HSM when needed.

A key element of a typical data retrieval request submitted at the GSFC DAAC is the need to stage, in one request, a large number of small files. Some HSMs tend to perform poorly when several hundred or thousand of files need to be staged, even if the files reside on few tapes. Other products put a limit (e.g. 100) on the number of stages that can be submitted at once, reducing overall performance, requiring substantial software design to properly handle the staging, and having a large impact on the day to day operations. On average, most of the files currently archived at the GSFC DAAC are small (around 1 MB) while data requests range from a single file to several thousand files at a time, resulting in a high penalty when retrieved from tapes. The overhead of the pick, mount, load, search and rewind operations is high compared to the read/write operation which may take only a few seconds for these small files. Consequently, it is critical to minimize the number of mounts and maximize, whenever possible, the amount of files read/written per mount. It is therefore desirable to sort the order in which files are transferred to and from tapes by which tape they are on and their position on the tape. This may be achieved by knowing the physical location of the files on tapes and then writing software to request the files in that order. Unfortunately, this information is not easily available in HSMs such as UniTree. To maximize system throughput, it is also necessary to keep data transfer rates to/from the storage devices at nearly the limits imposed by the hardware. Detailed analyses

were done on the performance of the VHS drives under UniTree, and it was shown that data transfer rates were substantially less inside UniTree than those measured outside UniTree, even with just a single drive operating [1].

Performance is a key issue in an archive, but other considerations such as interoperability are equally important. HSM vendors with their own proprietary formats make the transition to another HSM very difficult and expensive. This can have disastrous consequences if a vendor decided to stop marketing their products or to stop support of a given hardware device, as was the case for UniTree and the Cygnet jukeboxes at the GSFC DAAC. The situation worsens as the size of archives increases dramatically (Petabytes). The GSFC DAAC also has a requirement to migrate all of its archived data under the control of the Version 0 system to the next generation system. By storing the data in a non-proprietary, generally used format such as tar, migration can be more easily and quickly accomplished, since all that is required is to physically move the tapes to the new system. The interoperability of the tapes can be resolved by having one or several standardized tape format(s). This is difficult to achieve when vendors disagree on the merits of the formats and have already invested large amount of money in them. Another approach may be to provide a mechanism for HSMs to recognize and read formats from various vendors and do this without sacrificing performance. An important feature that is not always available is the ability to reconstruct the data base from the data itself. For instance, UniTree data is useless without the UniTree data base. These problems have been recognized and an Information and Image Management International (AIIM) File Level Metadata for Portability of Sequential Storage Media group has been formed to address some of these issues. This group met for the first time in April 1996, in Chicago, Illinois.

Faced with storage requirements growing exponentially and limited budget, it may be necessary to store data off-line. This solution is even more attractive in a data center where many tapes are seldom requested. This feature seems to be ignored or is limited at best with some HSMs. It is not sufficient to indicate that the tape is off-line. At a minimum the physical location of each off-line media should be known by the HSM and operators should be prompted to transfer media between near-line and off-line in an efficient manner. This should be viewed as another level of hierarchy with full functionality, and statistics should be made available.

A key issue in any Data Center is the data integrity and the data preservation. To ensure the highest quality for all data ingested and distributed to the users, it is important to capture, report, and react to errors in a usable way. These errors could occur with the media, the drives, the disks, or be related to some software problems. Even soft media errors may need to be monitored to identify archive media degradation. Data corruption needs to be automatically detectable through methods such as computation and comparison of file checksums upon all archival and retrieval requests. In spite of being critical, errors are not always provided with enough information, are often listed in a cryptic form, are difficult to locate in log files, or are simply not reported. Programs requesting the data are often not provided with adequate feedback to respond to both critical (e.g., hard media) failure and non-critical (e.g. soft media) failures. This creates confusion, requires a high level of expertise, and can have a detrimental impact on day to day operations. Error detection is not sufficient in itself and "smart" algorithms should be in place to take appropriate actions after errors are discovered. For example, a configurable limit should be set pertaining to the number of retries to read or search for a file. Another example may be to not automatically mount new media when an unrecoverable write error is detected, since the problem could be due to a bad drive and could result in numerous new tapes being discarded. Similar problems can occur with WORM optical media where a failure due to a bad drive is incorrectly interpreted to be a media failure and a new media is requested.

When the request again fails, another new media is requested, and so on, until operators notice the problem and shut-down the operation. This can cause the loss of many platters and requires extensive manual intervention to rectify this situation . These examples illustrate that the hard-coded error handling policies implemented for general, success-oriented operations do not always function well within a large, operational system. These policies are easily correctable and changeable. Changing policies and requirements may be a trivial task to implement with an in-house system, but may be much more difficult to integrate with a commercial package.

When dealing with very large science data centers (Petabytes), scalability is a major issue. An HSM should be designed to scale not only with the volume but also with the number of files being archived. This may require distribution of the software as well as the hardware. Implementation of a Unix file system or a virtual disk system is not regarded as a viable solution because of its limitations. There is a limit in the operating system on the number of concurrent open calls. The name server in an HSM can also become a bottleneck with very large number of files and some of the modules composing a data archive system may have to be distributed over several machines to spread the load more evenly.

Purchasing a commercial product such as an HSM provides many advantages. On the other hand, there may be major drawbacks that should be diligently evaluated before making any decision regarding the need for a COTS product versus an in-house product. One major problem experienced at the GSFC DAAC was the integration of UniTree with custom archive and distribution software. The task was difficult, time consuming, expensive to implement, and caused long delays in the delivery of the whole system. One solution was to request the vendor to incorporate the desired functionality in a new release. However, these functions may be too specific to have market value; or when there is interest to other users, it usually takes months, if not years, before design, integration and release. Another approach is to contract the integrator to develop specific functions that are not part of the core commercial product. Besides the length of time to set-up the contract, provide the requirements, and then design, write, test and integrate the functions, there is a high risk involved in tailoring a commercial product to meet specific needs, as each new release of the product may require new customized development resulting in a high cost. All together, the process can be extremely lengthy in time and frustrating in having to write work-around software or procedures to try and handle the situation while waiting for the vendor to react. HSMs are rather complex systems, built for specific, well-defined systems, and are not without flaws. Some of these bugs may seriously limit how the system can be used and it may take weeks or months to obtain a patch to fix the problem. While requiring in-house resources and expertise, there is more control with programs developed in-house. Bugs can usually be rectified more quickly and decisions can be made internally to prioritize them. Moreover, the experience we had with UniTree and the discussion we had with other colleagues tend to confirm that HSMs have not yet reached the stage of maturity found in products such as data base management systems.

Part of the original charter of the GSFC Version 0 DAAC was to test EOSDIS concepts and standards. Experimentation with various HSM strategies and the development of Archer as a possible alternative to commercial HSM products fit within that charter. Having an in-house product would also increase the ability to add new media types, which usually takes place on a longer time scale with COTS. The high cost of commercial HSMs is another consideration that cannot be ignored and contributed heavily in the decision to develop Archer. This is even more important in a distributed environment where a home-grown HSM can be freely redistributed whereas a COTS has to be licensed for multiple platforms and sites. In addition to the expensive purchase price, there is usually a high maintenance cost and some integration development costs that makes commercial HSM

solution less attractive. While the preference is to use a commercial product, in some cases no commercial product can satisfy specific and unique needs, and the developer must rely too much on companies whose goals are oriented towards slightly different requirements or functions. A key to the usability of a COTS product is whether its main functionality matches or just resembles one's needs. If just resembling one's needs, as was the case of COTS HSM packages and the science data needs of the GSFC DAAC, then attempting to either fit the COTS package into a slightly different functionality or assuming new releases to include the required functionality can be costly in time, resources, maintainability, and usability . These are some of the arguments and justifications that led to the design and development of a secondary archive system at the GSFC DAAC. One can hope that HSMs will become, in the near future, mature and flexible products that satisfy a vast and varied quantity of customers at a reasonable price.

Design of the secondary archive

Archer is a hierarchical storage management system that was designed to satisfy the requirements specified in the previous section. Files can reside in a cache, be robotically accessible, or be on a tape off-line. Users do not need to know the physical location of the files (data transparency), however, this information is easily and rapidly accessible through an API or by querying the Oracle data base which is used to keep track of file locations. The use of a relational data base facilitated and expedited the development of the system and provided a journal file to insure integrity of the archive database. Migration between cache and tape is automated and data can be stored and organized by families. For instance, a family can represent all files that belong to a specific product and level. The Archer file names are similar to the ones used in Unix, yet there is no implementation of a Unix file system. Consequently, commands such as open/close are not available and others, such as ls must be simulated through database SQL commands (e.g., and "als" command is provided to simulate ls). Files are simply requested to be stored or retrieved to/from the archive via PUT and GET operations. Multiple users can be serviced simultaneously and the client/server architecture has been designed to permit a distribution of the various servers among different machines to make the system scalable.

Archer file names have two parts. The first part identifies the directory to which a file belongs. The second part identifies the file. Both the directory and the file part can be any arbitrary string of characters (e.g. "/" are not required) but by convention, the names have been chosen to be consistent with Unix. Each directory is assigned to a family when created and is stored in an Oracle database table. The first part of a file name must completely match one the Archer directories, the part remaining is considered the file name.

The architecture of Archer is illustrated in Fig. 3. The main components of the system are defined as:

client interface (API): This is a series of C-callable entry points through which requests are originated. A request can be made to archive files, retrieve files, list files, delete files, list directories, list families, add tapes, list tapes, delete tapes, and flush families. All client interfaces communicate with a single archive server process.

Files can archived and retrieved in any size batch using either a synchronous or asynchronous method. The client is responsible for copying files out of cache during a file retrieval request. Command-line wrappers exist around all API functions so that the Archer internals can be accessed from the shell.

archive server: Only one archive server exists per archive. The archive server supports multiple file servers, and is responsible for directing message traffic between client processes and file servers or rejecting any requests which contain invalid information. The archive server can run on any machine in the archive.

file servers: Each file server is responsible for managing requests and file tables for a set of families in the archive. The file server manages cache space for all requests and verifies that the requests are satisfied. Each file server can manage multiple cache directories. Each file server supports multiple storage managers. For performance reasons, file servers may run on different machines in the archive.

copy server: A copy server is a small process which receives requests from the file servers to copy files into cache for archive requests. The copy server can copy a configurable number of files into cache in parallel. The copy server exists to minimize the overhead involved with forking processes to copy files in parallel. One copy server runs on each machine in the archive.

storage managers: Each storage manager is assigned a subset of the file server's families. Each may manage a different media type. The storage manager is responsible for managing and ordering the storage/retrieval of requests to/from tape. Each storage manager supports multiple storage servers, all of which must contain the same media type.

storage servers: Each storage server controls an individual storage device whether it is a single drive, a stacker, or a more complex multiple drive robotic system. The storage server is responsible for all activities involved in the storage/retrieval of files to/from tape. These activities include the loading/unloading of tapes to/from drives, tape positioning, tape verification, and the reading/writing of files to/from tape. Each type of storage server has its own type of ACE control display.

Archive Control Environment (ACE):

This is a GUI interface through which the operator and the archive interact. The ACE interface displays the status of the storage server and the device it is monitoring. This status includes whether the device is on-line, off-line, reading, writing, or idle, and the names of the tapes in the slots of the device, if applicable.

Through this interface, an operator may be notified of various events (e.g. system restarts, tape write errors), some of which may require a response. An operator may be prompted to mount a series of tapes in various slots of the device, or they may issue a request to load tapes manually.

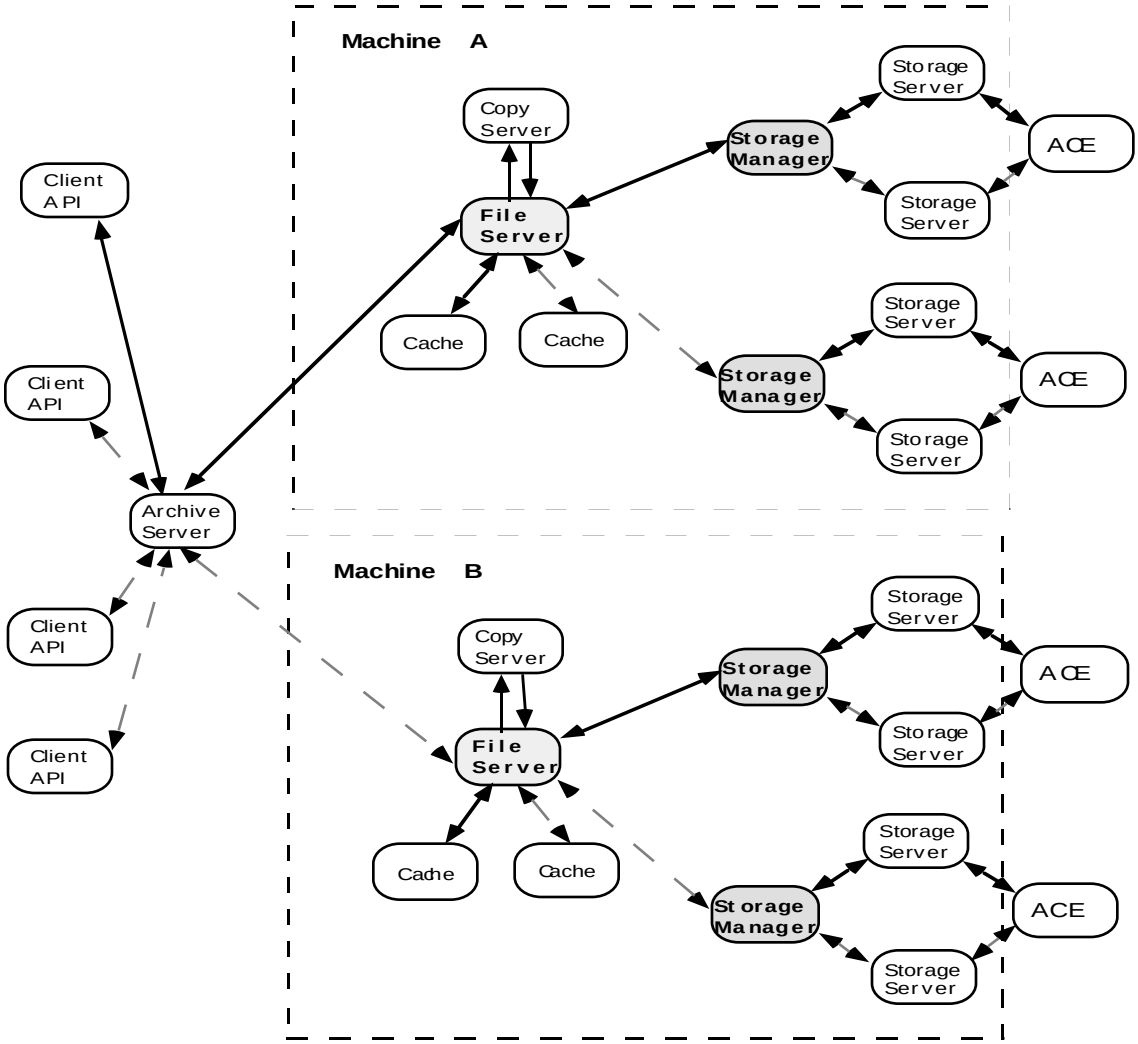

Fig 3 Archer Architecture

PUT and GET scenarios

In a typical PUT scenario, the client sends a request to the archive server to archive a file(s) to a specific family. The archive server directs the request to the appropriate file server. The file server allocates disk space in the cache and sends a message to the copy server to transfer the file(s) into the cache. After the file is copied to cache, a message is sent back through the system, informing the client of the cache transfer status. In a successful cache transfer, a message is sent to the appropriate storage manager. The storage manager receives and queues requests of successful cache transfers and waits for a pre-defined number of files (by family) to be staged in the cache before submitting a request to the storage server to copy the files to tapes. Finally, the storage server mounts the right tape and writes the data to it.

In a typical GET scenario, the client sends the request to the archive server which forwards it to the appropriate file server. The file server identifies whether the file(s) resides in the cache. When the file is not in the cache, the file server allocates disk space in the cache and sends a message to the storage manager to retrieve the file. When the storage manager

determines the time is right to fetch the file, a message is sent to the appropriate storage server, the right tape is mounted, and the file is read from tape into the cache. A message is then transmitted back through the system informing the client of this transfer. To avoid authorization problems the client is responsible for copying data from the cache to its location.

Archer storage format

In designing the Archer storage format, the option of using a proprietary format such as the one implemented in UniTree was rejected due to concerns with portability, and flexibility. Another important consideration was the ability to reconstruct the metadata directly from tape without the need of the database. This feature can be useful in the event of a disaster and can also facilitate the migration to another archive system which may not have access to the database system. There is no official standard archive format available but tar is a de-facto standard with Unix and other platforms, and for this reason was selected as the best candidate to satisfy our requirements. As mentioned above, the GSFC DAAC average file size (at the current time) is relatively small (1 MB) and, therefore, saving each file in a separate tar format would result in a heavy performance and space penalty. To alleviate this problem, groups of files are saved in a tar file called a "save set" prior to being migrated to tape. The number of files to tar together is usually selected so that a "save set" is around 50-100 MB for a 1-2 MB/s tape drive. The size of the save set is configurable for different media and data types (i.e., families) in order to best utilize the performance characteristic of the tape drives based on the file characteristics of the data. When a file is requested from an Archer tape, the whole save set where the file resides is read from tape and untared on the fly. Reading a save set takes longer than reading a single file but this penalty is small compared to the high overhead associated with the mount/load/search times. In addition, since the data requests are based on high quantity, batch file retrievals, neither single file access (such as provided by UniTree) or, the even more granular, block oriented access (such as provided in the AMASS HSM system) provide any benefit, and can, in fact ,hurt overall performance for this type of system. The Archer storage format is illustrated in Fig 4.

Error detection and recovery

From the beginning of the design of Archer, special care was given to error detection and recovery. This is critical not only to minimize impact on day to day operations but also to insure the integrity of the data archived and distributed at the GSFC DAAC. The first type of errors to examine is media failure. When a tape write error is detected, several pre-assigned and operator configurable number of attempts are executed. Continued failure will cause an operator prompt to occur with the option to continue retrying the operation, to ignore the requested operation, or to retry the operation on a different tape in the case of a hard write error. If the operator chooses to ignore the requested operation, he/she can then take the suspected drive off-line to avoid continuous operator prompts resulting from this write error. With a tape read failure, the read operation is retried for an operator configurable number of times, then marked as failed. Operators are notified on their terminals of the media problems.

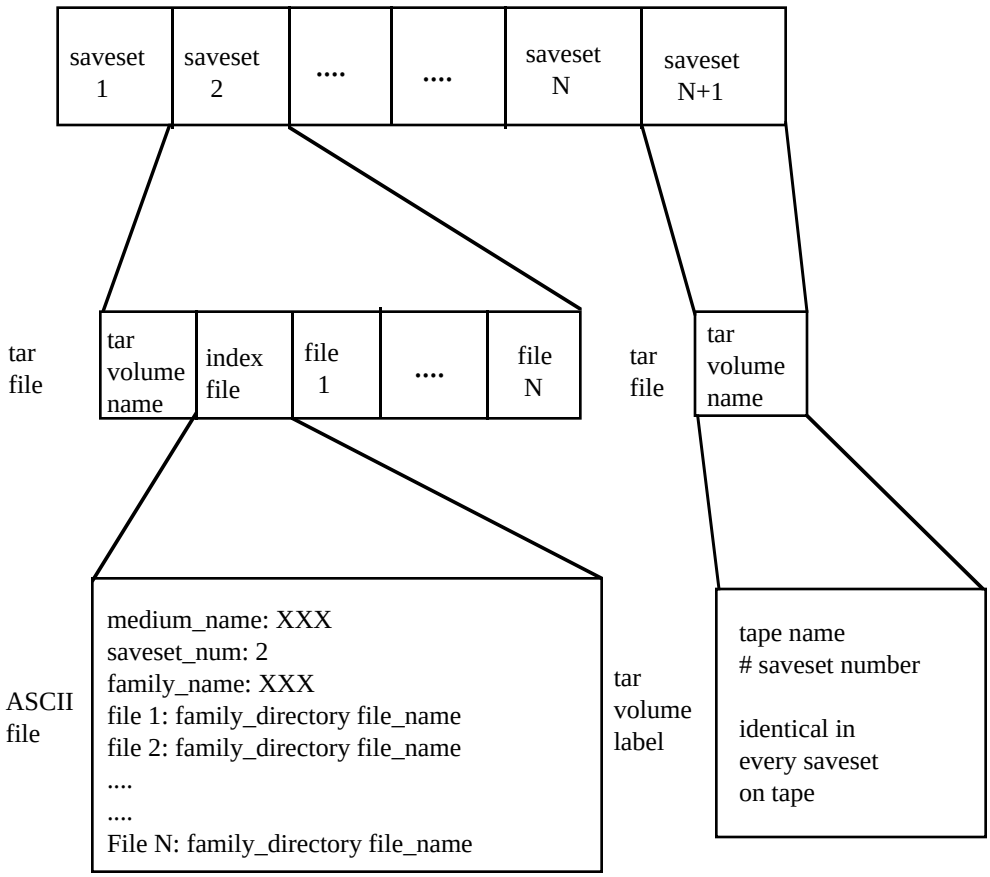

Fig 4 Archer tape format

Performance

One of the main considerations in the design of Archer was to develop a system with good performance. The emphasis was on the gross throughput of groups of related files as opposed to single-file turn around time. In order to achieve this objective several key features have been implemented. As mentioned above, files are grouped in save sets, improving the performance of a system with small files. To increase the hit cache ratio, a cache management algorithm has been developed on the file server with the capability to easily include new scheduling algorithms if desired. Improved log messages have also been designed to track the status of each file (examples: staged and purged) in the system and to monitor and generate performance statistics. New files ingested in the system are queued in the cache and copied to tape only after a pre-assigned volume of data is reached. This allows a large volume of data to be copied with a single tape mount. Files requested are first searched for in the cache. When the files are not located in the cache, Archer will sort files in the order they are physically stored on tapes, to minimize the overhead due to file positioning on the tape and the mounting and dismounting of tapes. Archer was developed with a multi-threaded client/server architecture and multi-threaded tape I/O architecture that provides efficient streaming of tape drives. The DLT tape drives have been tested to read/write close to the peak transfer rates advertised by vendors. Having a large database that contains the logical to physical relationship provides easy to utilize

information but, due to the size of the files (millions) and the need to continuously access the table, performance is adversely affected. To partly alleviate this problem, the first part of the file name maps to the family name, which allows a quick identification of the table to which the file belongs. As mentioned in the Status and Future Work Section, future versions of Archer will be independent of a relational database system.

Operational concepts

One of the goals of Archer was to facilitate the operational activities at the GSFC DAAC as well as the jobs performed by operators. One of the features of ACE (utilizing a graphical Tcl/Tk interface) is to provide a message button that highlights problems encountered. For example ACE (see Fig. 3) may list a tape write error . Archer processes are carefully monitored by an overseer process and if a problem arises, a message is displayed to indicate if the processes exited normally, abnormally, or failed due to a signal. In the event of failure, the archive is automatically restarted and the operator is notified.

Table 2 summarizes the issues discussed above.

Table 2: Summary of Archer Features and Functions

Issues	Features
Good performance	-low overhead to sustain operation at near tape speed - minimize number of mounts - maximize number of files requested from tapes - multi-threaded tape I/O - multi-threaded client services - hierarchical storage (disk cache, magnetic tape, off-line) - sort file read order by tape - allow large batch reads for improved sorting
Interoperability	- no proprietary tape format (use tar) - open system - self contained (contains data & metadata) (HDF) - recreate metadata dbms from reading tapes
Large requests of small files	- save set
Archive management	- support on-line, near-line, and off-line media
Flexible	- API -configurable parameters (based on data type or families, media, system, etc.).
Capture and monitor errors	- tape drive - media - disk cache failure - ACE display/monitor system
Error recovery	- before file is cached - before migration - during migration
Scalable system	- distributed H/W - distributed S/W - distributed storage devices
Administration	- reliable - archive multiple copies - collect statistics - errors - performance - facilitate migration from V0 to V1 - reduce dependencies on vendors - minimum coupling with DADS software - simplify integration - simplify exportation - integrity - journal file - support operator assisted off-line tape access
Kept simple	- does not implement a Unix file system - file name similar to Unix file system - simple synchronous and asynchronous put/get user interface - retrieval is by family and file identifier - COTS software to handle archive database
Hierarchical storage management	- files can be in cache, on tape, or off-line - identical storage and retrieval operations - automatic migration from cache to tape

Status and Future work

Since its delivery in Fall 1995, Archer Version 4.4 has been used on several occasions to recover lost files. Based on random audits, no file loss from Archer has yet been detected and Archer outperforms UniTree in archive operations, especially with large batches. There have been some operational problems. For example, some unexpected tape errors have occasionally caused the Archer system to hang. Also, only one single cache disk is currently supported and file and tape status is available only through SQL database queries.

The next build of Archer, scheduled to be operational in August 1996, should improve the overall performance through better internal scheduling of database operations. Multiple cache support has been added. Error recovery has been modified to prompt operators when several tape retries failed and to provide a choice of options. A global process monitors Archer and alerts operators to any problem detected.

Several other NASA groups have expressed an interest in Archer and there are plans to enhance Archer to be more like a COTS package with full documentation and its own configuration management (independent of the DADS development). The two main features envisioned are to remove Archer dependency on Oracle by maintaining the needed information internally and in disk files, and to improve the storage manager and storage server to better support new robotic devices and drives.

Conclusion

The GSFC DAAC has successfully designed and implemented a secondary archive system with a staff of one to three programmers over a fifteen month period. The initial release was operating after only seven months of design, development and testing. Though still in its infancy, Archer is satisfying the most pressing needs of the GSFC DAAC.

While Archer provides only a subset of the functions available with COTS software like UniTree, it supports migration between near-line and off-line media and offers good performance and flexibility. By selecting tar as tape format, Archer makes data more portable between Unix systems.

References

[1] Architecture and Evolution of Goddard Space Flight Center Distributed Active Archive Center, Jean-Jacques Bedet, Wayne Rosen, Mark Sherman, Hughes STX; Phil Pease, NASA/Goddard Space Flight Center, NASA Conference Publication 3295, March 28-30, 1995.