Storage Resource Management for Data Grid Applications

Program**: National Collaboratories**
Emphasis Area: **Middleware Technology**
(Areas of interest: **Data Management, Data Grids**)

Principle Investigator:
Arie Shoshani
National Energy Research Scientific Computing Division
Lawrence Berkeley National Laboratory
Mail Stop 50B-3238
Berkeley, CA 94720
Tel: (510) 486-5171
Fax: (510) 486 -4004
Email: shoshani@lbl.gov

Participants:

LBNL:
Arie Shoshani < shoshani@lbl.gov> PI
Alex Sim <asim@lbl.gov> Co-PI
Andreas Mueller < amueller@lbl.gov>

Fermilab:
Don Petravick < petravick@fnal.gov> Co-PI
Rich Wellner <wellner@fnal.gov>

Requested funding for each year; total request:

Year 1: $500K Year 2: $519K Year 3: $538K

Table of Contents

Abstract

Terascale computing often generate petascale data, where petabytes of data need to be collected and managed. Such data intensive applications already overwhelm scientists who spend much of their time managing data, rather than concentrate on scientific investigations. Because such data are now vital to large scientific collaborations dispersed over wide-area networks, such as the High Energy Physics experiments and Climate modeling simulations, there is a growing activity of developing a grid infrastructure to support such applications. Initially, the grid infrastructure mainly emphasized the *computational* aspect of supporting large distributed computational tasks, and optimizing the use of the *network* by using bandwidth reservation techniques. In this proposal we complement this with middleware components to manage the *storage* resources on the grid. Just as compute resources and network resources need to be carefully scheduled and managed, the management of storage resources is critical for data intensive applications since the access to data can become the main bottleneck. This situation stems from the fact that much of the data are stored on tertiary (tape) storage system, which are slow because of their mechanical nature. Data replication in multiple disk caches is also limited due to the volume of data. Finally, transfer of large data files over wide area networks also needs to be minimized. Based on previous experience with several prototype storage management systems, we propose to enhance the features of storage resource managers (SRMs), to develop common application interfaces to them, and to deploy them in real grid applications. We target both tape-based and disk-based systems, and design their interfaces so that they inter-operate.

1. Background and Significance

The amount of scientific data generated by simulations or collected from large scale experiments have reached levels that cannot be stored in the researcher's workstation or even in his/her local computer center. Such data are vital to large scientific collaborations dispersed over wide-area networks. In the past, the concept of a Grid infrastructure mainly emphasized the *computational* aspect of supporting large distributed computational tasks, and optimizing the use of the *network* by using bandwidth reservation techniques (called "quality of service") [1]. In this proposal we complement this with the support for the *storage* management of large distributed datasets. The access to data is becoming the main bottleneck in such "data intensive" applications because the data cannot be replicated in all sites. We describe briefly two scientific application areas to illustrate the need to manage storage resources. These areas are High Energy Physics (HEP) and Climate Modeling and Prediction (CMP).

In HEP experiments, elementary particles are accelerated to nearly the speed of light and made to collide. These collisions generate a large number of additional particles. For each collision, called an "event", about 1-10 MBs of raw data are collected. The rate of these collisions is 1-10 per second, corresponding to 30-300 million events per year. Thus, the total amount of raw data collected in a year amounts to 100s of terabytes to several petabytes. After the raw data are collected they undergo a "reconstruction" phase, where each event is analyzed to determine the particles it produced and to extract its summary properties (such as the total energy of the event, momentum, and number of particles of each type). The volume of data generated after the reconstruction phase ranges from a tenth of the raw data to about the same volume. Most of the time only the reconstructed data are needed for analysis, but the raw data must still be available. Another activity that produces similar amounts of data is the simulation of event data, and their reconstruction. Most of the data reside on tertiary storage (robotic tape systems), managed by a mass storage system, such as HPSS[♣]. The next phase of the scientific investigation is the analysis of the data, where 100s – 1000s of scientists all over the globe need to access subsets of the data. The access of data out of tape for analysis is usually the main bottleneck, and therefore repeated reading of the same files from tape should be avoided.

The simulation and reconstruction phases are obviously compute intensive, but they also produce huge amounts of data that have to be moved to archives. Data Grids are necessary not only to distribute the computation to various sites, but also to move the data to archives. The output of computations require temporary disk storage before the data can be moved to tape. This is where dynamic storage reservation management is needed. Storage management is even more critical in the analysis phase. While it is possible to predict the storage needs in the simulation and reconstruction phases, the analysis phase is more ad-hoc. Files have to move to multiple sites where the analysis takes place based on what the physicists want to examine. This ad-hoc process requires storage management to avoid unnecessary duplication of data in storage devices, as well as reading the same files repeatedly out of tape. We address in this proposal the management of storage for both tape and disk storage systems.

A similar situation exists in CMP. Climate simulation programs generate very large datasets depending on the resolution of the grid meshes being simulated. The size of such a simulation is already measured in terabytes. This data generation rate is expected to increase as more ambitious models are simulated with finer resolution, better physics, and a more accurate representation of the land/ocean boundaries. The data is generated and stored in time order. Given a time point, the simulation programs generate some 30-40 measures, called "variables" (such as temperature or wind velocity) for each spatial point on the mesh. In the analysis phase, scientists wish to access subsets of the data, consisting of a selection over the time, space, and variables. This requires reading of many files form tape to extract a relatively much smaller amount of data. Here again, avoiding repeated reading of files from tape is crucial. Similarly, the sharing of files by multiple scientists requires storage management and coordination that is based on the access patterns of the data. Since scientists analyzing the simulation data are in multiple physical sites, grid-based storage management is an integral part of the analysis process.

In the past, storage management issues were avoided by pre-allocation of space, a great deal of data replication, and pre-partitioning of the data. For example, HEP data was pre-selected into subsets (referred to as "micro-DSTs" or "streams"), and those were stored in pre-selected sites where the scientists interested in this subset perform the analysis. Because the data are stored according to the access patterns anticipated when the experiment was implemented, analysis requiring new access patterns is made more difficult, since the time to get events other than

[♣] http://www.sdsc.edu/hpss/

the subset at hand was prohibitively long. This mode of operation is not preferred since scientists accessing the data can be in multiple remote institutions, and the amount of data is expected to grow. To improve access to the data in this mode, data grids are now being developed, and storage resource management is a needed integral part of the middleware of data grids. Examples of such grid middleware being developed are Globus [11], and SRB [12]. We note that our proposal to manage storage resources complements the other services provided by these projects, especially grid security, efficient file transfer, and replica catalogs. Examples of HEP data systems that would benefit from this research are Fermilab's Run II experiments [13] and their enabling middleware SAM [14] and Enstore [15]

2. Purpose and goals

The purpose of this proposal is to address the problem of managing the access to large amounts of data distributed over the sites of the network. It is necessary to have Storage Resource Managers (SRMs) associated with each storage resource on the grid in order to achieve the coordinated and optimized usage of these resources. The term "storage resource" refers to any storage system that can be shared by multiple clients. The goal is to use shared resources on the grid to minimize data staging from tape systems, as well as to minimize the amount of data transferred over the network. The main advantages of developing SRMs as part of the grid architecture are:

1) *Support for local policy*. Each storage resource can be managed independently of other storage resources. Thus, each site can have its own policy on which files to keep in its resource and for how long.
2) *Temporary locking.* Files residing in one storage system can be temporarily locked before being transferred to another system that needs them. This provides the flexibility to read frequently accessed files from disk caches on the grid, rather than reading files repeatedly from the archival tape system.
3) *Advance reservations.* SRMs are the components that manage the storage content dynamically. Therefore, they can be used to plan the storage system usage by making advanced reservations.
4) *Dynamic space management.* It is essential to have SRMs in order to provide the dynamic management of replicas according to the locations they are needed the most (based on access patterns).
5) *Estimates for planning.* SRMs are essential for planning the execution of a request. They can provide estimates on space availability and the time till a file will be accessed. These estimates are essential for planning, and for providing dynamic status information on the progress of multi-file requests.

3. Preliminary Studies

3.1 Types of storage resource managers

The term "storage resource" refers to any storage system that can be shared by multiple clients. We use the term "client" here to refer to a user or a software program that run on behalf of a user. Storage Resource Managers (SRMs) are middleware software modules whose purpose is to manage in a dynamic fashion what should reside on the storage resource at any one time. There are several types of SRMs: Disk Resource Managers (DRMs), Tape Resource Managers (TRMs), and Hierarchical Resource Managers (HRMs). We explain each next.

There are several types of SRMs: Disk Resource Managers (DRMs), Tape Resource Managers (TRMs), and Hierarchical Resource Managers (HRMs).

A Disk Resource Manager (DRM) manages a single shared disk cache. This disk cache can be a single disk, a collection of disks, or a RAID system. The assumption we make here is that the disk cache is made available to the client through some operating system that provides a file system view of the disk cache, with the usual capability to create directories, open, read, write, and close files. The function of a DRM is to manage this cache using some policy that can be set by the owner of the disk cache. The policy may restrict the number of simultaneous requests by users, or may give preferential access to clients based on their assigned priority.

SRMs are essential for the dynamic replication of files based on actual access requirements. By actively managing what resides on each storage resource based on access patterns, files that are accessed more frequently (so called "hot files") stay longer in disk caches. This maximizes file sharing to ease file data movement on the grid. The management of storage resources is also essential to the cooperative scheduling of compute, storage, and network resources. Furthermore, this work would lead to a clear definition of the Application Programming Interfaces (APIs) and the functionality of storage resource management, so that existing software can be adapted to the same framework.

The general architecture for using SRMs in a grid is shown in Figure 1 below. The top part of the diagram shows clients in some site submitting "requests" for job processing to a "request manager". The request manager consults various catalogs and other information sources to determine which files it needs to get and where to get them. In the figure, we show a "metadata catalog" used to transform a user request (such as "space, time, variables" for a climate request) into the set of desired file; a "replica catalog" to determine where replicas of the files exist; and a "network weather service" to help determine the best location to get each file replica. Once the location of replicas was determined, the request manager submits requests for file locking and space reservations to the various SRMs on the grid. It uses the file transfer grid service (e.g. the Globus gridFTP) to actually move the files. The SRMs shown in the bottom of the figure can be anywhere on the grid, and can be DRMs, TRMs, or HRMs. We also show the use of a DRM as managing the local disk cache.

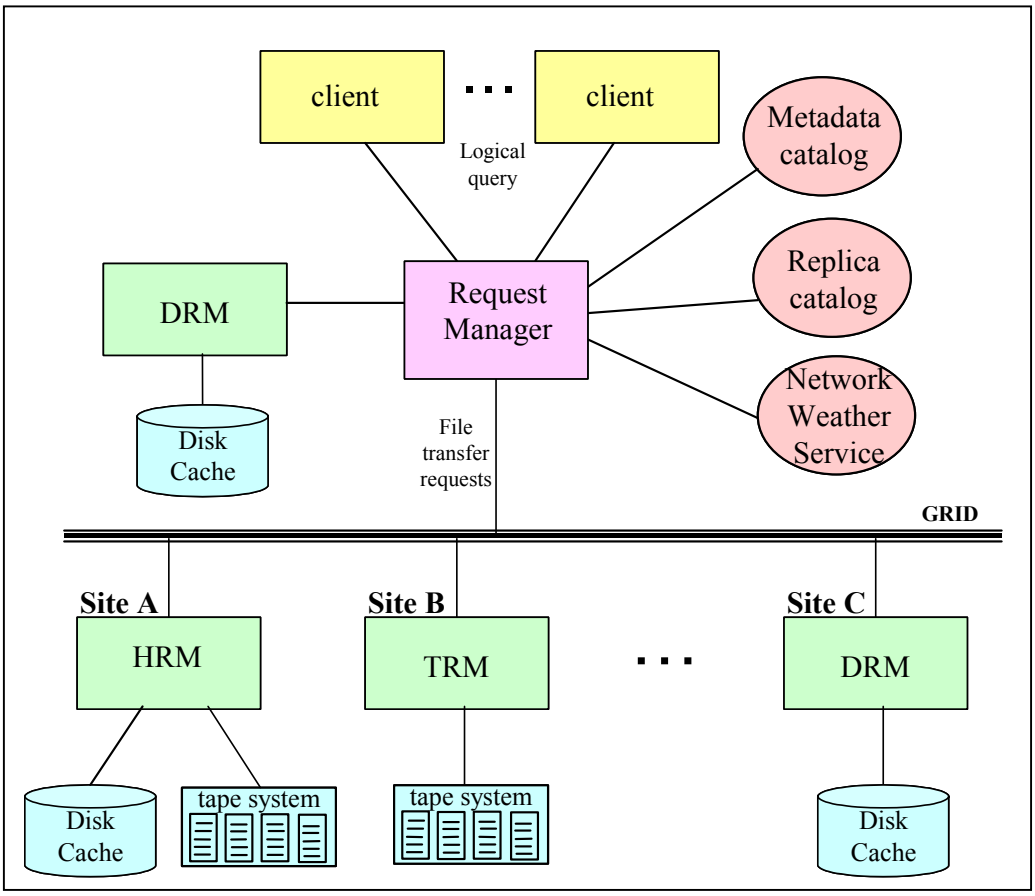

Figure 1: General architecture of grid services and SRMs role

3.2 Previous work on SRMs

Our approach is to make Storage Resource Managers (SRMs) as part of the grid middleware architecture. Each SRM is responsible for the management and policy enforcement of a Storage Resource (such as a shared disk cache, or a shared robotic tape system). The applications or program invoked by applications make requests to such SRMs for space reservations, for temporary locking of files, and for file transfer requests. Consequently, the applications need only express application-specific logical file requests, and the Grid infrastructure can take care of interacting with the necessary SRMs to get the data the application needs in the most efficient way. This work is based on our experience with managing large datasets for High Energy Physics applications as well as Climate simulation applications. Furthermore, we have developed and deployed early versions of SRMs in the NGI related projects.

We describe below the work performed so far in the development of prototypes of an HPSS-HRM, and an early version of a DRM.

Specifically, our development work of SRMs is based on experience we had with a system designed to support multiple High Energy Physics applications sharing data that is stored on a tertiary storage system. The system, called STACS (for Storage Access Coordination System) [2-6], was developed under the Grand Challenge program and is now deployed and used by the STAR project [7]. One of the components developed under this program was the Storage Manager, which is responsible for queuing requests and file transfer monitoring from tape to disk (requesting file transfers from HPSS) [5]. We further developed this component, called the HRM, so that it can be applied to a grid architecture under the NGI program. It accepts URLs of the files requested, accessing HPSS to stage the files to its local disk, and calling back the requesting component when a file is staged. In addition to pre-staging and call back capabilities, HRM provides the client with status capability estimating the time till staging will be done.

In collaboration with the Fermi National Laboratory (Fermilab), we developed a common interface to that system. This common interface was used by Fermilab to link to their SAM data access and Enstore network attached storage system. We used the same interface for our HRM to interface to HPSS in Particle Physics Data Grid (PPDG) experiments. Thus, we have demonstrated the feasibility of having *a single interface to completely different systems*. This is often the case in large collaborations. In addition to its use in PPDG, we applied the same HRM to the Earth Science Data Grid (ESG), demonstrating its usefulness across multiple application areas. The HRM was part of the ESG prototype demonstrated at the SC 2000 conference (it received the "hottest infrastructure award").

The HPSS-HRM was enhanced recently to provide a more general grid interface. The capabilities of the HPSS-HRM were also enhanced for its use in the CMS HEP project [9] project. Specifically, in the past HRM relied on a "file catalog" that contained information about the tape ID that HPSS assigned to a file as well as the file size. The new enhancements now use a newly developed HPSS access module called HSI [10] to extract this information dynamically for requested files. This made HRM more general and applicable to multiple experiments that use HPSS.

A version of an "on-demand" DRM was developed as part of the STACS system mentioned above. From this experience we gained insight on the caching policies required to manage the cache. We developed an algorithm for deciding what should be removed from the disk cache when space is needed based on the anticipated needs of the requests made to the system. This algorithm also managed coordinated access to multiple files that are needed *at the same time* by the client. These were referred to as "file bundles". This algorithm was published in [4]. However, this cache management was not developed as a separate module, but rather it was developed as an integral part of the job scheduler. Recently, we started to design the functionality and the interfaces to an independent DRM as part of the current PPDG project. An early version of this DRM has been developed. We plan to apply this DRM to a real experiment and use this experience in future developments.

Finally, in order to demonstrate the generality of SRMs, we have interfaced our HRM to the Storage Resource Broker (SRB) [12], so that it can be accessed through an SRB client. Developed at the San Diego Supercomputing Center (SDSC), the SRB is client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. We have demonstrated in a prototype system the use of the HRM with SRB. An interface was developed from the SRB server to the HRM system to perform file staging requests. When a file is cached by the HRM, the SRB server is notified, and it takes care of transferring the file to its destination upon the client's request. Consequently, two new functions were added to SRB, a "stage" call, and a "status" call. The "stage" call allows the user to request pre-staging of files from HPSS tape to the HRM disk, and requesting the file at a later time. The "status" call is used to return to the SRB user the information provided by HRM on the status of the file, such as the time to completion of the staging request. The combined system was tested with the SRB client at the University of Wisconsin, and the SRB-HRM server at LBNL. This experience helped us in generalizing the functionality of the HRM.

This experience provided us with the insight for the development of general purpose SRMs as proposed below.

4. Research Design and Methods

4.1 Methodology

There are three facts of data intensive scientific applications that this proposal addresses: 1) Very large datasets (hundreds of terabytes to several petabytes) are stored primarily on tertiary mass storage systems (such as HPSS), usually not at a site local to the application. 2) Some of the files that are more commonly used may be replicated on disk caches in various sites of the Grid. 3) Applications often do not need all the files at once to proceed with the analysis (nor do the have disk space for them).

We claim in this proposal that similar to network and computational resource reservation mechanisms, storage resources need to be reserved, coordinated, and shared according to a fair policy. However, one advantage of a storage resource over a network or a compute resource, is *that multiple users can share a storage resource at the same time* if they happen to need the same data. So, we have an opportunity here to try and optimize what should reside on which storage resource at any one time. Similarly, we can permit partial replication of popular data in order to avoid re-transfer of data over the network. The backbone for providing such capabilities is the concept of SRMs.

The main concepts that guide the design of SRMs are the replica type and the ability to "pin" a file. Since they form the basis of our proposed future work, we discuss these in some detail in the next sections.

4.1.1 Replica types

Replicas in a data grid have three possible status types depending on their expected usage and function. The status could be used to simplify the interaction with Storage Resource Managers (SRMs), in that locking of files can be avoided.

a) A "permanent" type
This type of replica refers to a file stored in a location that is intended to be "permanent", and is usually a location on some tape system. Typically, it is the archive used to store the files. Archives are usually tape system, but certain files can be designated as "permanent" and stored on some disk cache because they are expected to be accessed often. The concept of a "permanent" can be used to support a "primary copy" concept, in that the only entity that can remove this permanent file is the owner of that file.

b) A "quasi-permanent" status
A "replica administrator" who decides where replicas should reside creates this type of replica based on his/her knowledge of the expected use of a file. This file is "quasi-permanent" in that it is likely to stay in the assigned location for long periods of time, but can be removed by the administrator if expected use diminishes. In contrast to a "permanent file" a "quasi-permanent file" can be created and removed by an entity other than the owner of a file, such as the data administrator of a disk cache on some site.

c) A "volatile" status
This type of replica is created dynamically because of users' requests to process the file. It will stay in place if the demand for it is high, and otherwise will be removed when space is needed. Volatile files can be created or removed by the SRM that manages the storage resource without explicit permission from an owner or administrator (but based on some local policy).

The above replica types are so fundamental to a data grid design that they need to be supported by all generic SRMs. Exceptions may be permitted for special types of SRMs, such as archives that are used to store permanent files only. SRMs should also be able to support the dynamic change of a file type by an owner or administrator action. For example a volatile file can be changed to a quasi-permanent file so that it cannot be removed.

4.1.2 "Pinning" and "two-phase pinning"

The concept of *pinning* is similar to locking; it is a temporary lock that has a time stamp associated with it. The action of "pinning a file" results in a "soft guarantee" that the file will stay in a disk cache for a pre-specified length

of time. The length of the "pinning time" is a policy determined by the disk cache manager. The need for pinning stems from unpredictable delays on the grid as well as unreliable behavior of clients. Suppose that a client at site x finds out that a certain file exists at site y, and wishes this file to be replicated at its site. By the time the file transfer request is issued the file may be removed from site y (because space is needed). This can occur because of network delays, or the system at site y being temporarily busy. To avoid this situation, we propose to use a technique called "two-phase pinning". First, the client requests that the file will be pinned. After the file is pinned by the storage manager, it is transferred to its destination, and then released by the client. Given that a client can be unreliable (it may be irresponsible, it may crash, or the network connection may fail) the time out is used to avoid permanent locking of the file. After the time out period elapses there is no guarantee that the file will be kept in the disk cache, although it may be kept longer if another client is using it, or space is not immediately needed.

Two-phase pinning is akin to the well known "Two-phase locking" technique used extensively in database systems, except that the lock is temporary. While two-phase locking is used very successfully to synchronize writing of files and avoiding deadlocks, two-phase pinning is used to prevent files from being removed from a disk cache prematurely. Pinning can also be used to synchronize the requests for multiple files *concurrently,* but similar to locking this requires algorithms to avoid deadlocks.

4.1.3 Why is "pinning" useful?

The distinction of a replica type can help avoid pinning of "permanent" files, because it is not supposed to be removed. A "volatile" replica is likely to be removed if access demand to it is low. Thus, there is a possibility that the replica is removed in between the time that the client finds about its existence from the replica catalog, and the time that the replica needs to be transferred or accessed. Further, there is some chance that the replica will be removed in the middle of its transfer to another site. Pinning can be used to avoid pre-mature removal of files.

Regardless of which type of replica is accessed, and whether it is pinned there is a need for the requester to detect and recover from failures. This is because there is no absolute guarantee that a pinned replica will be available at the time of transfer - the source system may be down, or the local administrator claimed the storage space. Also, there is no absolute guarantee that the transfer will complete properly because of network failures. However, there is a higher probability that a pinned file will be available at the time it is needed. Letting an SRM know that a file will be needed increases the probability that the SRM will keep the file until it is transferred. Letting the SRM know that a file was released, helps the SRM manage its storage resource more effectively.

When the replica is quasi-permanent, the likelihood of it being removed at the time it is needed is small. However, we need to allow a the type of a quasi-permanent file to be changed dynamically to a volatile file by the administrator and vice versa. To accommodate this the quasi-permanent file can be treated as a volatile file but have a "quasi-permanent flag" associated with it. This issue is important for the design of "pinning strategies" as discussed below.

Volatile replicas are essential in order to manage dynamic storage allocation of replicas; that is, the ability for SRMs to determine dynamically which files to keep in their system at any one time. The goal is to dynamically and automatically migrate replicas to the site where they are used the most. This requires the registration of volatile replicas in the replica catalog soon after they are replicated, so that they are globally known. Similarly, if a file is removed by an SRM, the replica catalog entry will have to be removed as well before the file is physically removed.

Another reason for pinning is for advanced reservation and planning, what is referred to as the "quality-of-service" (QOS). In such a scenario, one may want to reserve space, pin several files, reserve network bandwidth, and transfer the files during this QOS window. For this to work, it is necessary that the source replicas are either permanent, or they are pinned.

4.1.4 Pinning strategies

As a practical matter, SRM implementation is simpler if the SRM does not have to keep track of pinning. However, an SRM can still perform "simple pinning" strategies. We'll call the pinning strategies from simple to sophisticated as level-0-pinning, level-1-pinning, etc. We identify 4 such levels, and discuss them briefly next. The discussion below refers to pinning of files in disk caches. This applies to both DRMs and HRMs. In the case of an HRM it refers to the disk system that the HRM manages.

a) Level-0-pinning
This is the case that pinning requests are not made at all. SRMs do not mark files as pinned even when a file is being transferred. Rather the SRM finds out if a file is in use or was recently in use to determine which files to remove when space is needed. Using this strategy, the SRM looks for the "oldest" file (or files), usually by a call to the underlying file system to check for files "last touched". A time-out may be associated with this, where a file can be removed if it was not touched for a certain time period. In this case, a release of a file is meaningless. This strategy is sufficient to increase the likelihood that files that are accessed often remain in the disk cache.

b) Level-1-pinning
In this case, pinning requests are made, but the SRM does not keep track which client made the pinning request. It has a single time-stamp associated with each file. Originally, the file is time stamped when it is first brought into the disk space. This time stamp is updated every time some client requests a pin. When space is needed the SRM checks for the oldest time stamp, and removes the associated file. Similar to level-0-pinning, level-1-pinning ensures that files accessed often remain in the cache. But it has the advantage that it is not necessary to check or update "last touched" for all the files, an expensive operating system operations if we manage thousands of files. It can avoid operating system calls altogether.

c) Level-2-pinning
This strategy keeps track of which client requested the pin, and when each client requested the pin. The main reason for keeping track of pin-per-client is to prevent clients from issuing repeated pins to the same file, thus keeping the file in disk cache indefinitely. Another advantage is that it makes it possible for the SRM to queue transfer requests, rather than refusing them when it is overloaded. With level-2 pinning, additional policies for treating users fairly or according to a priority assignment is possible, since the identity of the requester is known. In addition, queuing of requests is a very important feature for HRMs since they transfer files from tape. If the request is queued, an estimate can be provided as to the length of time before the file is staged to the disk cache and ready for transfer.

d) Level-3-pinning
This is a request to pin a file for a certain time and duration. As mentioned above, this is the case where a client wishes to pin a collection of files for a certain time period, when it has also reserved network bandwidth (QOS), and space in a target site where the files will be moved to. This level of pinning requires negotiations between the requestor and the SRM, as well as a cost model (or client priorities) assigned and managed.

Using any of the above strategies, there is no requirement that files must be pinned before they are transferred. It is still possible to transfer a file without pinning. The advantage to the requestor to perform a pin is that it is treated like an advanced reservation, requesting that the replica will be kept in cache for the length of the time-out policy. The advantage to the system as a whole is that it can manage storage resources effectively, avoiding re-staging of files from tape, and the ability to plan complex requests. This provides more efficient file delivery to the user, and effective use of the grid resources.

We propose to implement increasingly more sophisticated caching strategies. The HRM that use simple policies and low-level pinning should be relatively easy to implement and deploy early.

4.2 work plan & schedule

We plan to develop initially two types of SRMs in a generic way, so that they can be applied to various scientific applications: an HRM that will initially work with HPSS and a general purpose DRM. The generality will be achieved through the use of standard Application Programming Interfaces (APIs). Our plan is to develop progressively SRMs with more features, based on experience with previous less powerful versions. For example, the initial version of a DRM will not have support for queuing file requests. This feature will be added with a later version after some practical experience with the DRM is gained.

The proposed work includes several aspects:

1) Deployment, scalability, and robustness.
We will deploy the current prototypes in real application environments. LBNL already developed an initial version of an HRM, and expects to have an initial prototype of a DRM ready by the time this proposal is scheduled to start.

The DRM prototype will support only "level-1 pinning". As explained above, this version does not support request queuing or policy enforcement, but is sufficient to deploy in real applications. This task involves checking the scalability of the HRM and DRM, as well as its robustness. Our Fermilab collaborators will perform the deployment, first by adapting the APIs to their SAM-Enstore system, and then by using LBNL's DRM in their system. This will be followed by deployment in the PPDG collaboratory Pilot as well as the ESG collaboratory Pilot (both proposed to SciDAC).

2) Design of concepts, functionality, and APIs of advanced features.
Currently, the HRM and DRM are designed to support only "read" requests for analysis, assuming that data was previously placed in storage resources. We plan to extend their capabilities support both "read" requests and "write" requests. They will accept requests from any site, reserve space, temporarily pin files in place when necessary, and initiate and monitor the transfer of files as necessary. LBNL and the Fermi collaborators will perform this design work jointly. The design will also be coordinated with the PPDG and ESG collaboratory Pilots.

3) The implementation of the SRM modules with additional features.
The LBNL team will implement the new features. The Fermilab collaborators will adapt the interfaces to and testing the software with their mass storage and data access system. They will also help test and deploy the more advanced DRMs to physics and astrophysics projects. Specifically, the following tasks will be performed:

a) The DRM will be designed to include level-2 and level-3 pinning strategies. This includes the implementation and testing of various caching policies. In the first year only simple policies (based on most recently used information) will be implemented. After deployment and experience with simple policies, more complex policies (based on usage history, user hints, and access patterns) will be incorporated.
b) The DRM will be developed to include the ability of the application to write files into it. File could be written as temporary or as permanent.
c) The HRM will be enhanced to include internally a DRM. This will provide with the HRM the same generic capabilities that independent DRMs provide.
d) Our HPSS-based HRM will be extended to include a write capability. This will require storing of file to be archived in temporary disk storage, and then scheduled for archiving in tape storage systems.
e) File migration from DRMs to archival storage will be supported.

4.3 Deliverables

Year 1

- Perform scalability tests of existing prototype versions of HRM and DRM.
- Design and test for fail-safe and robustness of existing prototype versions of HRM and DRM.
- Deployment of HRM in real experiments. We plan to target at first CMS and Atlas.
- Perform scalability tests and robustness of the HRM interface to SAM-Enstore at Fermilab.
- Design advanced features of HRM and DRM, especially "write" features.
- Develop a direct interface from the Globus gridFTP to HRM. This will provide a viable alternative to direct access of the MSS by gridFTP, by providing an intermediate disk cache for buffering and pre-staging of files from the MSS.

Year 2

- Implement a DRM with level-2 pinning strategy, including support for queuing requests and per user policy enforcement. Deploy this advanced DRM at Fermilab and in at least one of the PPDG pilot experiments as well as in the ESG pilot.
- Implement "write" capabilities for HPSS-HRM. This includes the ability to temporarily store files in the HRM's disk cahce, and migrate to tape in the background.
- Implement "write" capability at Fermilab on top of the SAM-Enstore system
- Design the ability to support advance reservations to the SRMs. This is a complex task that will require coordination with the PPDG pilot project.

Year 3

- Deploy level-2 DRMs in additional PPDG and ESG sites. Design and perform scalability and robustness tests
- Implement a reservation-based DRM and HRM. We plan to use optimization policies for such reservations developed by others, as proposed in the PPDG pilot.

- Develop an option to use HRM as a TRM only. This means that the HRM will not support its own disk cache, but rather request direct transfers of files from the MSS to the network. A direct transfer from an MSS is planned as part of Globus middleware services, and we expect by that time we could rely on it.
- Deploy HRM and DRM in multiple cooperating application areas. This task will involve actively seeking applications that can benefit from SRM grid services, especially in Astrophysics and Earth science.

4.4 Coordination with other datagrid projects

The SRMs we develop will be applied to specific application areas proposed in several Collaboratory Pilots. In particular, we will apply (and adapt) them to work with multiple High Energy Physics experiments through the proposed SciDAC Particle Physics Data Grid Collaboratory Pilot project - CMS, STAR, ATLAS and D0 (see letter of support from Richard Mount – physics PI). In addition, we plan deploy our SRMs in the proposed SciDAC "Earth Science Grid Collaboratory Pilot" project (see letter of support from one of the PIs, Dean Williams – LLNL). Another potential SciDAC collaboratory pilot that intend to use our SRMs is "Diagnostic Portal for Comparative Analysis of Observed and Simulated Climate Data" (PI: Bahram Parvin - LBNL). Fermilab will also work with the EU DataGrid project Storage Management on areas associated with the LHC experiments.

In addition, we will apply then to grid applications in climate modeling storage systems. This work will be coordinated with the SciDAC proposed middleware project: "A High Performance Data Grid Toolkit: Enabling Technology for Wide Area Data-Intensive Applications" (PIs: Ian Foster - ANL, Carl Kesselman - ISI). In particular, we plan to update the replica catalog automatically for files being cached dynamically. We also plan to rely on any advancement in higher bandwidth transfers developed by the Globus project. Another aspect of our collaboration is to take advantage of advances developed as part of the Probe project at ORNL and NERSC (LBNL). In particular, the development of HSI will provide a more efficient access to HPSS tapes than parallel FTP, as well as providing information on the location of files, so we can order staging requests to minimize tape mounts. We plan to interface to HSI as the front end to HPSS.

5. References Cited

[1] The Grid: Blueprint for a New Computing Infrastructure Edited by Ian Foster and Carl Kesselman, Morgan Kaufmann Publishers, July 1998.

[2] New Capabilities in the HENP Grand Challenge Storage Access System and its Application at RHIC, L. Bernardo, B. Gibbard, D. Malon, H. Nordberg, D. Olson, R. Porter, A. Shoshani, A. Sim, A. Vaniachine, T. Wenaus, K. Wu, D. Zimmerman, to be published in the Journal of Computer Physics Communications, 2001 (CPC'01).

[3] Experience with Using CORBA to Implement a File Caching Coordination System, Alex Sim, Henrik Nordberg, Luis Bernardo, Arie Shoshani, Doron Rotem, to appear in Journal of Concurrency: Practice and Experience, 2001; 13:1-15.

[4] Coordinating Simultaneous Caching of File Bundles from Tertiary Storage, A. Shoshani, A. Sim, L. M. Bernardo, H. Nordberg, 12th International Conference on Scientific and Statistical Database Management (SSDBM'00).

[5] Access Coordination of Tertiary Storage for High Energy Physics Application, L. M. Bernardo, A. Shoshani, A. Sim, H. Nordberg (MSS 2000).

[6] Storage Access Coordination Using CORBA, A. Sim, H. Nordberg, L. M. Bernardo, A. Shoshani, D. Rotem, 1999 International Symposium on Distributed Objects and Applications (DOA'99).

[7] Multidimensional Indexing and Query Coordination for Tertiary Storage Management, A. Shoshani, L. M. Bernardo, H. Nordberg, D. Rotem, and A. Sim, (SSDBM 1999).

[8] Grand Challenge Application on HENP Data Focused on Efficient Data Access for RHIC, http://www-rnc.lbl.gov/GC.

[9] The Compact Muon Solenoid experiment at CERN's Large Hadron Collider,
http://www.cithep.caltech.edu/cms/gem.html and http://cmsinfo.cern.ch/Welcome.html.

[10] Third-party HPSS-HPSS transfers via Hierarchical Storage Interface (HSI),
http://www.csm.ornl.gov/PROBE/hsi.html.

[11] The Globus project: developing fundamental technologies needed to build computational grids,
http://www.globus.org.

[12] The Storage Resource Broker (SRB), The San Diego Supercomputing Center (SDSC),
http://www.npaci.edu/DICE/SRB.

[13] CDF/CD/D0 Joint Offline Computing Project http://RunIIComputing.fnal.gov

[14] D0 Sequential Acccess using Meta-Data http://d0db.fnal.gov/sam/

[15]Enstore Mass Storage System http://www-isd.fnal.gov/enstore/

6. Budget summary

This project is planned for a 3-year period. The total budget plan of $500K is based on about 2 FTE for design, development, and implementation tasks in LBNL ($400K), and 1/2 FTE at Fermilab ($100K) for their participation in joint design and definition of application interfaces, as well as adapting the interfaces to and testing the software with their mass storage and data access system. Fermilab will also help test and deploy DRMs to physics and astrophysics projects.