LBNL-48998

Storage Resource Managers: Middleware Components for Grid Storage

Arie Shoshani, Alex Sim, Junmin Gu
Lawrence Berkeley National Laboratory

Abstract

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 [1] 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 paper we discuss the concept of Storage Resource Managers (SRMs) as components that 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. SRMs can be used to dynamically optimize the use of storage resource to help unclog this bottleneck.

1. What are 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 runs on behalf of a user. Storage Resource Managers (SRMs) are middelware 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.

A Disk Resource Manager (DRM) manages dynamically a single shared disk cache. This disk cache can be a single disk, a collection of disks, or a RAID system. The disk cache is available to the client through the 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. However, space is not pre-allocated to clients. Rather, the amount of space allocated to each client is managed dynamically by the DRM based on requests by the client. The function of a DRM is to manage this cache using some policy that can be set by the administrator 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. In addition, a DRM may perform operations to get files from other SRMs on the grid. This capability will become clear later when we describe how DRMs are used in a data grid.

A Tape Resource Manager (TRM) is a middleware layer that interfaces to systems that manage robotic tapes. The tapes are accessible to a client through fairly sophisticated Mass Storage Systems (MSSs) such as HPSS, Unitree, Enstore, etc. Such systems usually have a disk cache that is used to stage files temporarily before transferring them to clients. MSSs

typically provide a client with a file system view and a directory structure, but do not allow dynamic open, read, write, and close to files. Instead they provide some way to transfer files to the client space, using transfer protocols such as FTP, and various variants of FTP (e.g. Parallel FTP, called PFTP, in HPSS). The TRM's function is to accept requests for file transfers from clients, queue such requests in case the MSS is busy or temporarily down, and apply a policy on the use of the MSS resource. As in the case of a DRM, the policy may restrict the number of simultaneous transfer requests by users, or may give preferential access to clients based on their assigned priority.

A Hierarchical Storage Manager (HRM) is a TRM that has a staging disk cache for its use. It can use the disk cache for pre-staging files for clients, and for sharing files between clients. This functionality can be very useful in a data grid, since a request from a client may be for multiple files. Even if the client can only process one file at a time, the HRM can use its cache to pre-stage the next files. Furthermore, the transfer of large files on a shared wide area network may be sufficiently slow, that while a file is being transferred, another can be staged from tape. Because robotic tape systems are mechanical in nature, they have a latency of mounting a tape and seeking to the location of a file. Pre-staging can help eliminate this latency. Another advantage of using a staging disk in an HRM is that it can be used for file sharing. Given that multiple clients can make a request for files to an HRM, the HRM can choose to leave a file longer in cache so that it can be shared with other clients based on use history or anticipated requests. The HRM design is based in part on experience in a previous project reported in [2].

In general, it is best if SRMs are shared by a community of users that are likely to access the same files. They can be designed to monitor file access history and maximize sharing of files by keeping the most popular files in the disk cache longer.

2. The role of SRMs in a Data Grid

Suppose that a client runs at some site and wishes to analyze data stored in files located in various sites on the grid. First, the client must have some way of determining which files it needs to access. Checking a file catalog, using some index, or even using a database system can accomplish this step. We refer to this step as "request interpretation". The information used in this step is commonly referred to as metadata, since the result of this step will be a set of logical file names that need to be accessed. The second step is to find out for each logical file where it physically resides or replicated. Note that a single logical file can be replicated in multiple sites if it was accessed previously by clients at these sites. Files can also be pre-replicated into sites based on expected use. In a grid environment, this information exists in a "replica catalog", a catalog that maps a single logical file name to multiple physical files names located in various sites. The physical file name includes a name of the site, the directory path on that system, and the file name.

In many grid environments today, the burden for the above work is being thrust on the clients. Therefore, it is now recognized that such tasks can be delegated to middleware components to provide these services. A "request manager" is the term used to refer to such services. The request manager performs "request planning" based on some strategy, and then a "request

execution" of the plan. There are three options to consider: either move the application program to the site that has the file, move the file to the client's site, or move both the program and the data to another site for processing. All three possibilities are valid, and much of the middleware development addresses this issue. In all these cases, SRMs play an important role. In the case that the program moves to the site where the file exists, it is necessary to "pin" the file in that site; that is, to request that the file remains in that site, so that when the program is initialized the file in found in the cache. When the program completes, the file can be "released". In the case that the file needs to be transferred from a source site to target site (either to the client's site, or to another site), it is necessary to "pin" the file in the source site, to reserve the space in the target site, and maintain this state till the transfer to the target site is complete. Then the "pin" can be released. Here, the SRM at the source site has the role of managing the "pinning", and the SRM at the target site has the role of allocating space (i.e. making space by removing other files if necessary), and reserving the space till the transfer completes.

SRMs need to deal also with system failures, so that space reservations do not persist forever, and "pins" do not persist in case that a "release" is not performed. The concept of "pinning a file" is central to SRMs and will be discussed further later.

3. Advantages of using SRMs

The main advantage of an SRM is that it provides smooth synchronization between shared resources by pinning files, releasing files, and allocating space dynamically on an "as-needed" basis. But, in addition, they can eliminate unnecessary burden from the client. First, if the storage system is busy, SRMs can queue requests, rather than refuse a request. Instead of the client trying over and over again, till the request is accepted, an SRM can instead queue the request, and provide the client with a time estimate based on the length of the queue. This is especially useful when the latency is large such as for a reading a file from tape. If the wait is too long, the client can choose to access the file from another site, or wait for its turn. Similarly, a shared disk resource can be temporarily full, waiting for users to finish processing files, and queuing requests is a better alternative to simply refuse the request.

A second advantage to the client is that SRMs can insulate them from storage systems failure. This is an important capability that is especially useful for HRMs since MSSs are complex systems that fail from time to time, and may become temporarily unavailable. For long lasting jobs accessing many files, which is typical of scientific applications, it is prohibitive to abort and restart a job. Typically, the burden of dealing with an MSS's temporary failure falls on the client. Instead, an HRM can insulate clients from such failures, by monitoring the transfer to the HRM's disk, and if failures occur, the HRM can wait for the MSS to recover, and re-stage the file. All that the client perceives is a slower response. Experience with this capability was shown to be quite useful in real usage [2].

4. "Pinning" and "two-phase pinning"

The concept of *pinning* is similar to locking. However, while locking is associated with the *content* of a file to coordinate reading and writing, pinning is associated with the *location* of

the file to insure that a file stays in that location. Unlike a lock, which has to be released, a "pin" is temporary, in that it has a time-out period associated with it, and the "pin" is automatically released at the end of that time-out period. 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 the inherently unreliable behavior of the data grid (because of system failures, network failures, or irresponsible clients). Since we cannot count on pins to be released, we use the pinning time-out as a way to avoid pinning of files forever.

Two-phase pinning is akin to the well known “two-phase locking” technique used extensively in database systems. While two-phase locking is used very successfully to synchronize writing of files and to avoid deadlocks, two-phase pinning is used to synchronize requests for multiple files *concurrently*; that is, if the client needs several files at the same time, it can first pin these files, and only then execute the transfers for all files, then releasing them as soon as each is transferred. We note, that even if file replicas are read-only, a deadlock as a result of pinned files can occur if we allow requests for multiple files concurrently. However, if we assume that file requests are asynchronous and that time-outs that release files are enforced, pin-locks are eventually resolved because pinned files are released after they time-out. Nevertheless, two-phase pinning is a useful technique to avoid system thrashing by repeatedly pinning and pre-emptying pins. It requires coordination between the SRMs.

5. Design of “Read” and “Write” functionality of SRMs

When a request to read a file is made to an SRM, the SRM may already have the file in its cache. In this case it returns the address of the file in its cache. The client can then read the file directly from the disk cache (if it has access permission), or can copy or transfer the file into its local disk. In either case, the SRM will be expected to pin the file in cache for the client for a period of time. A well-behaved client will be expected to “release” the file when it is done with it. This case applies to both DRMs and HRMs.

If the file is not in the disk cache, the SRM will be expected to get the file from its source location. For a DRM this means getting the file from some remote location. For an HRM, this means getting the file from the MSS. This is another capability that SRMs provide to alleviate this task from the client. Rather than return to the client with “file not found”, the SRM provides the service of getting the file from its source location. Since getting a file from a remote location or a tape system may take a relatively long time, it should be possible for the client to make a non-blocking request. To accommodate this possibility the SRMs provide a call-back function that notify the client when a requested file arrives in its disk cache and the location of that file. In case that the client cannot be called back since it does not have a server, SRMs also provide a “status” function call that the client can use to find out when the file arrives. The status function can return estimates on the file arrival time if the file has not arrived yet.

HRMs can also maintain a queue for scheduling the file staging from tape to disk by the MSS. This is especially needed if the MSS is temporarily busy. If a queue exists, then the HRM puts the request at the end of the queue. Otherwise, it schedules its staging

immediately. Like a DRM, the HRM needs to notify the client that the file was staged by issuing a call_back, or the client can find that out by using “status”.

A request to “write” a file requires a different functionality. In the case of a DRM, if a file size is provided, then that space is allocated, and the client can write the file to it. Otherwise, a default size is assumed, and the available space is adjusted after the file is written. In the case of an HRM, the file is first written to its disk cache in exactly the same way as the DRM description above. The HRM then notifies the client that the file has arrived to its disk using a call_back, then it schedules it to be archived to tape by the MSS. After the file is archived by the MSS, the SRM notifies the client again using a call_back. Thus, the HRM’s disk cache is serving as a temporary buffer for files being written to tape. The advantage of this functionality by HRM is that writing files to a MSS can be performed quickly to the HRMs disk cache, and then archived as a background job to tape. In this way the HRM can eliminate the burden from the client to deal with a busy MSS as well as dealing with temporary failures of the MSS system.

6. Status

We have implemented several versions of a DRMas well as a HRM that interfaces to HPSS. The HRM is basically a TRM that deals with reading/writing files from/to HPSS, which embeds a DRM in it for managing its disk cache. These components have been incorporated into the Particle Physics Data Grid (PPDG) project to perform grid replication functions [3]. The HRM was also used in a demo for SuperComputing 2000 as part of an infrastructure to get files from multiple locations for an Earth Science Grid application (ESG) [4]. The SRMs use grid-enabled secure file transfer services provided by the Globus project [5], called gridFTP. We are now evaluating several “cache replacement policies” to be used by DRMs, by both conducting simulations and setting up real testbeds.

7. Conclusion

We discussed in this paper the concept of Storage Resource Managers (SRMs), and argued that they have an important role in streamlining grid functionality and making it possible for storage resources to be managed *dynamically*. While static management of resources is possible, it requires continuous human intervention to determine where and when file replicas should reside. SRMs make it possible to manage the grid storage resources based on the actual access patterns. In addition, SRMs can be used to impose local policies as to who can use the resources, and how to allocated the resources to the grid users. We also introduced the concept of "pinning" as the mechanism of requesting that files stay in the storage resource until a file transfer or a computation takes place. Pinning allows the operation of the coordinated transfer of multiple files to be performed as a "2-phase pinning" process: pin the files, transfer, and release pins. Several versions of prototype SRMs have been built and used in test cases as part of the Particle Physics Data Grid (PPDG) and Earth Science Data Grid (ESG) projects. The emerging concepts and interfaces seem to nicely complement other grid middleware services being developed by various grid projects, such as providing efficient and secure file transfer, replica catalogs, and allocating compute resources.

References

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

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

[3] Particle Physics Data Grid (PPDG), http://www.ppdg.net/

[4] Earth Science Grid (ESG), http://gizmo.lbl.gov/esg

[5] The Globus Project, http://www.globus.org