Storage Resource Managers:
Why They Are Important to Data Grid Architecture

Arie Shoshani

March 2001

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 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 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 run 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 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 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 in front of a robotic tape system. Such tape systems are accessible to a client through fairly sophisticated Mass Storage Systems (MSSs) such as HPSS, Enstore, etc. Such systems usually have some 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 MSS resource. As in the case of a DRM, the policy may restrict the number of simultaneous 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 multiple files to an HRM, the HRM can choose to leave a file longer in cache so that it can be shared with other client based on use history or anticipated requests.

2. A typical scenario

Suppose that a client runs at some site and wishes to analyze data located on various files 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 will 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. 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 such services. "Request interpreters" and "request managers" are terms that refer to such services. The request manager is a generic term that can further be subdivided to the function of "request planning" and "request execution".

The choice of which file replica to use depends on "request planning". There are three options: either move the analysis 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 can play an important role. However, for the sake of continuing with the scenario, suppose that the choice is made that all files needed by the client move to the client's site, site A. Suppose further that some of the requested files are located on its site's shared local disk, and the rest are in other sites. The request manager can then contact its local DRM at site A for all the files it needs. The DRM, in turn, will immediately pin all files it has in its cache, and let the request manager know that these files are in local cache. For the remote files, the DRM will first allocate space on its disk cache, and then contact remote SRMs and request pinning of files. It is not necessary for the DRM to get all the files at once. To get a file from a remote site the DRM will initiate the contact with the remote site. For example, suppose the client is at site A, and it needs file X which is managed by some HRM at site B. The DRM at site A first allocates space on its disk cache, and then contact the HRM at site B asking to pin file X. If the HRM has that file in its disk cache, it will pin it and notify the DRM at site A that the file is now available for transfer. If the file is not in the HRM's disk cache, it will stage it from tape to its disk cache and then notify the DRM at site A that the file is now available for transfer (an example of such a service is the Globus [4] gridFTP). Then the DRM can call a file transfer service to move the file. After the file is successfully transferred, the DRM at site A notifies the HRM at site B that it releases that file.

3. Advantages of the use of an SRM

1) Local policy administration

One of the main reasons to use an SRM is to allow the owner of the storage resource to apply a local policy for the use of that resource. In a data grid there may be some storage resources that belong to "the grid" as a whole, or to a community of users on the grid. However, most of the resources are usually owned by the site that generates the data or by a site that makes data available to others. For example, several communities usually share an MSS, and the owner of that resource may want to restrict the use of the system of each community. In addition, the site adminstrator may want to give preference to the site's users, and only then provide access to users on the grid.

2) Limiting access to the shared resource

As mentioned above, a storage resource may be "owned" by a site, and only partially or conditionally available to a grid community. In such a case an SRM can be used to limit access to the resource. For example, a shared disk resource may allocate a certain amount of disk to the grid community, and the DRM can enforce that its community clients do not exceed that use. In the case of a shared MSS, an HRM or TRM can be used to limit the number of simultaneous requests for file transfers (e.g. number of PFTPs in HPSS). Similarly, if there are several communities sharing the MSS, each can be assigned a quota of simultaneous requests for file transfers. More sophisticated policies can also be developed, where the quotas are based on how busy the system is at the time that requests are made.

3) Queuing of requests

Every system has a limit as to how many clients it can serve concurrently. This stems partially from the fact that each file transfer requires a fairly large amount of memory to perform the block transfers. In such cases, the request for a file is simply refused by the system. The client then has to try over and over again, till the request is accepted. This is not only a burden to the client, but also overloads the system that keeps refusing requests. An SRM can instead queue the requests, 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 MSS. 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 the file, or until a time out is applied. A queue can also be used in DRMs rather than refusing a request. DRMs may also be busy transferring files from other sites to their disk cache. Here again, a queue can be used when the DRM is temporarily overloaded. The combination of queuing requests with setting limits of use (as discussed in the previous point) has the effect of throttling the use of the storage resource.

4) Insulate clients from failure

This is an important capability that is especially useful for HRMs to provide because they control their own disk. Since an MSS that an HRM interfaces to can be very complex, they may fail from time to time, and 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. Thus, the burden of dealing with an MSS temporary failure falls on the client. 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. This capability was implemented in the STACS system [2, 3] and found to be very useful.

4. Replica types

Replicas in a data grid have three possible status types depending on their expected usage and function. This status could be used to simplify the interaction with Storage Resource Managers (SRMs).

1. 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 archive. Typically, it is the location that files are stored immediately after they are created. This usually corresponds to a file stored in "tier 0" in tier architecture. A "permanent file is a physical file that can only be created and removed by the owner of the dataset (or the data collection).

2. A "durable" type

An administrator who can make decisions on where replicas should reside creates this type of replica based on his/her knowledge of the expected use of a file. This file is "durable" in that it is likely to stay in the assigned location for a long periods of time, but can be removed by the administrator if expected use diminishes. This type of replica will usually resides in "tier 1" in a tier architecture, but can be stored in a "tier 2" level as well. A "durable" file is a physical replica of a file that can only be created and removed by the administrator of the disk cache.

3. A "volatile" type

"Volatile" replicas are created dynamically because of users' requests to process the files. A volatile file stays in the cache if the demand for it is high, and otherwise will be removed when space is needed. This type of replica is primarily stored in "tier 2" in a tier architecture, but can also be residing in "tier 1" according to the dynamic policy of replica management. A volatile file is a physical replica of a file that is subject to removal by the DRM according to preset policies.

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

The concept of *pinning* is similar to locking. 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 as a way to avoid pinning a file forever.

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. Two-phase pinning can also be used to synchronize the requests for multiple files *concurrently*; that is, if the client needs several files at the same time, it can first these files, and only then executing the transfers for all files, releasing them as soon as each is transferred.

We note, that under the widely accepted assumption that file replicas are only read, there is no need for detecting, or preventing deadlocks between the pinned files.

The usefulness of the distinction of a replica type is in that if a replica is "permanent" or "durable", it is highly likely to be available for file transfer (i.e. copy from one site to another). A "volatile" replica is more 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 is transferred or accessed. Further, there is some chance that the replica will be removed in the middle of its transfer to another site.

6. 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 resource 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. This allows the operation of transferring files to be performed as a "2-phase pinning" process: pin, transfer, and release pin. 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 [5,6]. The emerging concepts and interfaces seem to nicely complement other grid middleware components being developed by various grid projects.

References

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

[2] Storage Access Coordination System (STACS), http://gizmo.lbl.gov/stacs

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

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

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

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