# A Simple Mass Storage System for the SRB Data Grid

Michael Wan, Arcot Rajasekar, Reagan Moore, Phil Andrews
*San Diego Supercomputer Center*
*University of California, San Diego*
*(mwan, sekar, moore, andrews)@sdsc.edu*

## *Abstract*

*The functionality that is provided by Mass Storage Systems can be implemented using data grid technology. Data grids already provide many of the required features, including a logical name space and a storage repository abstraction. We demonstrate how management of tape resources can be integrated into data grids. The resulting infrastructure has the ability to manage archival storage of digital entities on tape or other media, while maintaining copies on distributed, remote disk caches that can be accessed through advanced discovery mechanisms. Data grids provide additional levels of data management including the ability to aggregate data into containers before storage on tape, and the ability to migrate collections across a hierarchy of storage devices.*

## 1. Introduction

The SDSC Storage Resource Broker (SRB) [1, 2, 3, 4] is data grid middleware that provides a storage repository abstraction for transparent access to multiple types of storage resources. The SRB has been used to implement data grids (to integrate access to data distributed across multiple resources), digital libraries (to support collection-based management of distributed data), and persistent archives (to manage technology evolution). The Storage Resource Broker is in widespread use, supporting collections that have five million images replicated across multiple HPSS archives, and data grids that span the continent. Persistent archives are now being implemented using the Storage Resource Broker in support of the National Archives and Records Administration, and the National Science Foundation National Science Digital Library.

The capabilities provided by the SRB represent a unique integration of data grids, digital libraries, and persistent archives. The mechanisms that were required to integrate these three types of data handling systems turn out to be uniquely suited to the implementation of a mass storage system. The name space is managed by the digital library technology, distributed physical storage resources are managed by the data grid technology, and technology evolution is managed through a persistent archive storage repository abstraction.

The SRB is implemented as a federated client-server system, with each server managing/brokering a set of storage resources. Storage resources that are brokered by the SRB include Mass Storage Systems (MSS) such as HPSS [5], UniTree [6], DMF [7] and ADSM [8], as well as file systems. What is of great interest is that the SRB data grid can be used to implement all of the capabilities of a distributed Mass Storage System, while providing access to data stored in file systems, databases, object ring buffers, databases, and other types of storage systems. A Mass Storage System based upon data grid technology can be implemented using virtually any type of storage device.

The motivations for implementing a MSS in a data grid are:

- Cost of licensing - Not all of our users can afford the licensing fees of commercial MSS systems. By implementing the Mass Storage System functionality within a data grid, a common software system can be used for resource federation and data sharing, as well as for data archiving.
- Efficiency and performance - A MSS system that is tightly integrated with the infrastructure of the SRB data grid can take full advantage of SRB features such as file replication, server directed parallel I/O, latency management, data based access controls, and collection based data management.
- Elimination of duplication of features – The SRB and MSS systems such as HPSS duplicate some features. For example, the HPSS has its own disk cache that is used as a front end to a tape system. The SRB can be configured to provide a cache that serves as a front end to a HPSS resource. Since the SRB has no knowledge of the operational characteristics of the HPSS cache, it may not be able to effectively manage its own cache utilization in conjunction with the HPSS cache management. With the integration of Mass

Storage System capabilities into the SRB, a single large cache pool can be used. Another example is that most MSS systems have their own authentication schemes in addition to the SRB authentication system. A single authentication system can be used if their capabilities are integrated. The resulting system is easier to administer.

- SRB already has many of the required capabilities – The features needed for a simple MSS include a name space, a storage repository abstraction, storage resource naming and data management tools. For example, the SRB Metadata Catalog (MCAT) [9] maintains a POSIX-like logical name space that eliminates the need to design a name server from scratch. Since the MCAT namespace is based on commercial database technology, the transaction performance is substantially better than current archives. Other reusable features will be discussed later. Leveraging these reusable features greatly reduces the effort required to implement a simple MSS in a data grid.
- Finally, an MSS based on data grids can provide a storage system that spans remote caches and distributed archival devices interconnected by a Wide-Area-Network. Such a logical linking of distributed devices will provide new ways for data sharing and fault tolerance not currently provided by site-located mass storage systems.

## 2. SRB Architecture and Features

The Storage Resource Broker (SRB) is middleware that uses distributed clients to provide uniform access to diverse storage resources. It consists of three components: the metadata catalog (MCAT) service, SRB servers for access to storage repositories and SRB clients, connected to each other via a network.

The MCAT is implemented using a relational DBMS such as Oracle, DB2, SQLServer, PostgresSQL, or Sybase. It stores metadata associated with data sets, users and resources managed by the SRB. It maintains a POSIX-like logical name space (file names, directories and subdirectories) and provides a mapping of each logical file name to a set of physical attributes and a physical handle for data access. The physical attributes include the host name and the type of resource (UNIX file system, HPSS archive, object ring buffer, database). The physical handle for data access is the file path for UNIX file system type resources. The MCAT server handles requests from the SRB servers. These requests include information queries as well as instructions for metadata creation and update.

The MCAT imposes additional mappings on the logical name space to support replication (one logical name mapped to multiple physical file names), soft links (a logical name mapped to another logical name), aggregation (structural mapping of a file to a location in a container), segmentation (structural mapping of a file across multiple tape media), file-based access control (users mapped to permissions on roles for each digital entity). These mappings make it possible to organize digital entities independently of their actual storage location.

The SRB is implemented as a federated server system. Each server consists of three layers. The top level "communication and dispatcher" layer listens for incoming client requests and dispatches the requests to the proper request handlers.

The middle layer is the logical layer or the "high-level API handler" layer. This layer handles requests in which all input parameters are given in terms of their logical representations (e.g., logical path name in the logical name space, logical resource name, logical user etc). Upon receiving a request, the logical layer handler queries the MCAT and translates the logical input parameters into their physical representations. It then calls upon the appropriate handler in the physical layer to perform the actual data access and movement.

The physical layer or the "low-level API handler" layer handles data access and data movement requests from its own logical layer or directly from the physical layer of other SRB servers. This layer basically consists of driver functions for the 16 most commonly used POSIX I/O functions: create, open, close, unlink, read, write, seek, sync, stat, fstat, mkdir, rmdir, chmod, opendir, closedir, and readdir. If the handler cannot handle the request locally, it will forward the request to the server that can respond.

## 4. Simple MSS Design

The design goals for a simple Mass Storage System are:

- Provide a distributed farm of disk cache resources backed by a tape library system. The cache system may be configured to contain any number of distributed cache resources that may or may not be on the same host as the tape system. This makes it possible to treat the disk cache as an independent level of the storage hierarchy, with the disk cache created “near” the end user.
- Provide a tape library system to control the mounting and dismounting of tapes. At a minimum, a Storage Tech silo running ACSLS software should be supported.

- Provide a uniform access mechanism for data stored on the Mass Storage System or on disk caches. A file in the logical name space stored in the MSS resource should appear and behave the same as any other files stored on other resources. The physical location (on cache or tape) of the file should be totally transparent to users.
- Files should always be staged automatically to cache before any I/O operations are done. Tools for system administrators to manage the cache system are also needed. i.e., tools to synchronize files from cache to tape and purge files on cache.
- Large files should be stored in segments. The advantages of using segmented files are:
  - The system can handle files of very large size.
  - Parallel data transfer between tapes and the cache system can be implemented. In our first release, although large files are stored in segments on tapes, parallel transfer between tapes and cache has not been implemented. Data transfer between the cache system and clients can be done in parallel using existing SRB infrastructure.

Based of the above design goals, the following software components are needed to build the MSS:

1. A client-server architecture with an authentication scheme appropriate for access across administration domains and a framework for exchange of information between clients and servers that can function over Wide Area Networks.
2. A federated server system that allows cache and tape resources to be located on different hosts.
3. A metadata server that maintains a logical POSIX-like name space and provides a mapping of each logical file name to its physical location.
4. Additional metadata and server functions that allow files stored in the MSS resource to appear and behave the same as any other files stored on other resources.
5. Storage resource servers that have the following capabilities:
   a. Ability to translate user requests to physical actions using metadata information maintained in the MCAT catalog.
   b. A set of driver functions for basic tape I/O operations.
   c. A set of driver functions for basic cache I/O operations.
   d. Functions to stage files from tape to cache and dump files from cache to tapes. The tape and cache resources can be distributed.
   e. Support for data transfer between the cache system and clients.
6. A tape library server whose primary function is to schedule and perform the mounting and dismounting of tapes.
7. A tape database that tracks the usage of all tapes controlled by the MSS.

## 4. Implementation

The SRB framework version 1.1.8 initially provided the functionalities listed above, except for the metadata needed to manage files on the MSS, the drivers for basic tape I/O functions, functions to stage files from tape to cache, and the tape library server and tape database.

A major innovation that was needed to implement a MSS within the SRB data grid was the development of a new compound resource type as a fundamental resource type within the SRB/MCAT system. Compound resources include a cache for I/O operations and a backend tape or archive for long term storage. Files written to a compound resource are treated as residing on a single resource. In order to allow files stored in the MSS resource to appear and behave the same as files stored on other resources, support is needed for compound digital entities. The file that is written to a compound resource can be migrated between the cache and the tape back-end within the compound resource, without requiring separate metadata attributes to describe the residency of the file on either component of the compound resource.

A compound resource contains multiple cache resources for a given tape resource (each of which are called internal compound resources). When a user creates a file using a compound resource, the object created is tagged as a compound digital entity. With the help of MCAT, the server then drills down through the compound resource and discovers all of the internal resources. It selects the cache resource where the digital entity will be stored initially. After the file operation is completed (with a “close” call), the metadata of the just created compound digital entity is updated with the physical description of the file pointing to the digital entity created in the cache resource.

A compound digital entity is treated as any other digital entity for most operations in SRB, except when the digital entity is opened for read or write. In this case, the server will check to see if the digital entity is already in a cache. If it is not, the digital entity will be staged on one of the cache resources configured in the compound resource. If the digital entity is changed, the dirty bit of the cached digital entity is set. The dirty copy is not automatically synchronized to tape. Synchronization is only done via requests by system administrators. A "dump tape" API and command are created to allow system administrators to manage the cache system by synchronizing files in the cache system onto tape, and then purging the files from the cache system.

The ability to manipulate data that is stored on tape requires additional capabilities beyond those required for access to data on disk. A set of driver functions for basic tape I/O operations have been defined and have been incorporated into the SRB server. These functions include mount, dismount, open, close, read, write, seek, etc. Currently, the driver has only been tested for 3590 tape drives.

A tape library server for the STK silo running ACSLS software has been incorporated into the SRB system. Its primary function is to schedule and perform the mounting and dismounting of tapes. It uses the same authentication system and server framework as other SRB servers. Currently, tape mount is on a first-come-first-server basis. Some amount of intelligence is built in such that when a client is done using a tape and issues a dismount request, the tape will not be actually dismounted if there is another request for the mounting of the same tape in the queue. In this case, the server will just pass the tape to the new request. Specialized queuing features can be implemented as needed.

A database schema that tracks the usage of all tapes controlled by the MSS has been incorporated in the MCAT. The schema is used to track the tape position, total bytes written, full flag, etc for all tapes controlled by the MSS. A set of system utilities has been created for tape labeling and tape metadata ingestion, listing of tape metadata and modification of tape metadata. By managing these attributes in the MCAT catalog, it is possible to support sophisticated queries against the tape attributes and against the attributes of the digital entities within the MSS. One can readily determine all of the files resident on a given tape, identify all tapes that are filled beyond a given level, and identify all tapes that are needed to retrieve all digital entities within a logical sub-collection in the metadata catalog.

## 5. Comparisons with the IEEE Mass Storage System Reference Model

The SRB MSS provides similar functionality to the IEEE MSS Reference Model [10].

Comparing with the implementations of the reference model, there are both similarities and differences. A major difference stems from the fact that the SRB MSS uses the underlying File System of the operating system for managing data storage instead of the mapping of bitfiles to the logical and physical volume abstractions used in the Reference Model. The use of a File System greatly simplifies the design of the storage manager. This approach is reasonable given that the performance of modern File Systems is quite good. Another source of difference is that the SRB MSS integrates the functionality of several servers of the Reference Model into a single server. This greatly simplifies the architecture of the SRB MSS. Improved robustness and performance of the system is achieved at the expense modularity. The design provides modular interfaces to support addition of new storage repositories and new access APIs, while aggregating all metadata into a single database.

The major components of the Reference Model include:

1. Name Server – provides POSIX-like name space, a mapping of logical names to bitfile IDs and access control (ACL) for the name space objects.
2. Bitfile Server – provides the abstraction of logical bitfiles to its clients and handles the logical aspects of the storage and retrieval of bitfiles.
3. Storage Server – handles the physical aspect of bitfile storage and retrieval. It translates references to storage segments into references to virtual volume and into physical volume references.
4. Mover – transfers data from a source device to a sink device.
5. Migration-Purge server – provides storage management by migrating bitfiles on disks to tapes.

The SRB MCAT server, which maintains a POSIX-like logical name space, is equivalent to the Name Server of the Reference Model. The only difference is that each SRB digital entity is mapped directly to a set of physical attributes rather than to a logical bitfile as in the Reference Model. Because of the direct mapping, the functionality of translating from logical bitfiles into physical volume references is not needed. The rest of the functionalities of the Bitfile Server, Storage Server and Mover of the Reference Model are combined into a single SRB Resource server. Separate SRB Resource servers are created for each type of storage device. For tape resources, the SRB uses a tape library server for mounting and dismounting of tapes. As for the Migration-Purge server of the Reference Model, SRB has an API and a system utility that migrates files on cache to tape resources.

## 6. Usage Examples

The use of data grid technology to implement a Mass Storage System makes it possible to incorporate latency management capabilities directly into the architecture. For access to data distributed across multiple resources, the finite speed of light can severely limit sustainable transaction rates, if the transactions are issued one by one. The SRB data grid provides multiple mechanisms to minimize the number of messages that are sent over a wide area network, including the aggregation of data into containers, the aggregation of metadata into an XML file, and the aggregation of I/O commands through the use of remote proxies.

For a Mass Storage System, the ability to aggregate data into containers is essential for achieving high performance when managing small digital entities. When the size of a digital entity is less than the tape access bandwidth multiplied by the tape latency, it becomes cost effective to work with containers of files. The size of the container is adjusted such that the retrieval time of two digital entities from the same container is smaller that the retrieval time of the two files directly from tape.

The usage scenario that illustrates the generality of the data grid based mass storage system is to consider the storage of a container on a mixture of caches, archives, and compound resources. This scenario requires the use of five different mappings on the logical name space:

- Mapping from the logical file name to a location within a container
- Mapping of the container to one of several replicas
- Mapping of a logical resource name to a physical resource name
- Mapping of access control lists between the user name and the requested file
- Mapping of a compound digital entity to its location in a compound resource

The SRB provides the ability to organize physical resources by a logical resource name. Writing to the logical resource name then results in the replication of the file across all of the physical resources. Separate metadata attributes are maintained for each replica of the file. The SRB also supports the aggregation of files into a container. Containers are manipulated on disk caches, and then written to an archive. A primary disk cache can be identified with multiple secondary disk caches. A primary archive can be identified with multiple secondary archives. When a primary resource is not available, the SRB will then complete the operation to the secondary resource. Note that containers can be replicated onto multiple storage repositories.

The SRB also supports compound resources composed of a cache and either a tape or archive. Writing a file to a compound resource results in the creation of a single set of metadata, with an attribute used to specify which component of the compound resource holds the data. The interesting management scenario is the creation of a logical resource name that includes a compound resource, a primary disk cache, secondary disk caches, a primary archive, and secondary archives. Writing a container to this logical resource then results in the creation of a replica of the container in the compound resource (disk cache and backend tape), and the creation of a replica on one of the disk caches. A synchronization command will cause the replica on the disk cache to be copied onto one of the archives.

The ability of data grids to support sophisticated resource management functions on top of distributed storage repositories makes it possible to greatly increase the number of options when archiving data. An example is the implementation of alternate completion scenarios, in which a file is assumed to be archived when it is written to “k” of the “n” physical resources specified by a logical resource name. Another example is the implementation of load balancing, by the writing of digital entities in turn to a list of resources specified by the logical resource name. The ability to replicate files across trees of storage resource options instead of the traditional simple storage hierarchy, greatly increases the ability to manage archival copies of data.

A second data grid capability that greatly enhances mass storage systems is the organization of the digital entities as a hierarchical collection. It is possible to use digital library discovery mechanisms to identify relevant files within the mass storage system. The discovery mechanisms can be exercised through interactive web interfaces, or directly from applications through C library calls.

A third data grid capability that simplifies management of mass storage systems is the association of access controls with the data, rather than the storage system. This makes it possible to include sites across administration domains within the mass storage system, while simplifying administration of the system. Data grids support collection-owned files, in which access to the files is restricted to the collection. Users authenticate themselves to the collection. The collection uses access control lists that are specified separately for each registered digital entity to determine whether a person is authorized to access a file. The collection in turn authenticates itself to the remote storage system.

A fourth data grid capability that simplifies incorporation of new technology into the mass storage system is the use of a storage repository abstraction that defines the set of operations that will be performed when accessing and manipulating digital entities. The storage repository abstraction makes it possible to write drivers for each type of storage system, without having to modify any of the higher software levels of the storage environment. The storage repository abstraction is also used to support dynamic addition of resources to the system.

## 7. Conclusions

Because of the cost of licensing, access efficiency and transaction performance issues, we have implemented a simple MSS system in the Storage Resource Broker data grid. We were able to accomplish this task within a relative short time (slightly over 6 man months) because we were able to leverage existing capabilities within the SRB infrastructure. We believe this approach will radically change how archives are constructed. The ability to manage replicas of data on low cost storage media is as simple as making a replica of a digital entity on a disk cache. The ability to discover, access, and manipulate digital entities stored on tape media can now be done through the same sophisticated interfaces that data grids provide for access to all types of storage systems.

## 8. Acknowledgements

This research has been sponsored by the Data Intensive Computing thrust area of the National Science Foundation project ASC 96-19020 “National Partnership for Advanced Computational Infrastructure,” the NSF National Science Digital Library, the NARA supplement to the NSF NPACI program, the NSF National Virtual Observatory, and the DOE ASCI Data Visualization Corridor.

## 9. References

[1] Baru, C., R, Moore, A. Rajasekar, M. Wan, (1998) "The SDSC Storage Resource Broker," Proc. CASCON 98 Conference, Nov.30-Dec.3, 1998, Toronto, Canada.

[2] SRB, (2001) "Storage Resource Broker, Version 1.1.8", SDSC (http://www.npaci.edu/dice/srb).

[3]Rajasekar, A., and M. Wan, (2002), “SRB & SRBRack - Components of a Virtual Data Grid Architecture**”,** *Advanced Simulation Technologies Conference (ASTC02)* San Diego, April 15-17, 2002.

[4]Rajasekar, A., M. Wan, and R. Moore, (2002), “MySRB & SRB - Components of a Data Grid,” *The 11th International Symposium on High Performance Distributed Computing (HPDC-11)* Edinburgh, Scotland, July 24-26, 2002.

[5] HPSS, High Performance Storage System, http://www4.clearlake.ibm.com/hpss/index.jsp.

[6] UniTree, http://www.unitree.com.

[7] DMF, Data Migration Facilitty, http://136.162.32.160/products/software/dmf.html.

[8] ADSM, ADSTAR Distributed Storage Management, http://searchstorage.techtarget.com/sDefinition/0,,sid5_gci214398,00.html.

[9] MCAT, (2000) "MCAT:Metadata Catalog", SDSC (http://www.npaci.edu/dice/srb/mcat.html).

[10] Sam Coleman and Steve Mller, “Mass Storage System Reference Model: :Version 4", Goddard Conference on Mass Storage Systems and Technologies, Volume 1, 1992.