VOSpace and VOStore Design

Reagan W. Moore
San Diego Supercomputer Center
moore@sdsc.edu

Abstract[1]

Data grid technology provides the ability to manage shared collections that are distributed across multiple storage systems. Based on the principles behind data grids, the design of standard storage repository access mechanisms (VOStore) and standard information management infrastructure for organizing shared collections (VOSpace) are examined, with the intent of specifying the minimal requirements needed for a functional system.

1. Introduction.

The Astronomy community is developing standard services for accessing image archives and object catalogs. The standard services provide simple interfaces to retrieve information about stars and galaxies (Cone Search) and information about images (Simple Image Access Protocol). Two new services are being developed:

- VOStore – a simple access mechanism to retrieve images
- VOSpace – a minimal information management system to organize shared collections.

The development of the new services is being driven by the desire to support access to images that individuals have acquired, not just the large all-sky surveys. The latter typically provide portals for accessing their image archives, as well as catalogs for discovering object of interest.

A major challenge is differentiating between the capabilities that should be supported within VOStore versus the capabilities that would be supported by VOSpace. One way to differentiate capabilities is to note that personal access to personally owned images requires less information. The owner of the images has the knowledge required to interpret the naming conventions of the files, understand where the files are stored, and has the permissions required to access the data. The owner is able to run a utility like GridFTP to directly interact with the storage system and retrieve a file. A VOStore interface to personally owned data can be as simple as GridFTP.

When data is published, such that others can discover and retrieve relevant images, a more sophisticated interface (VOSpace) is needed that provides:

- Descriptive metadata to support discovery (this can be FITS header information that is loaded into a metadata catalog).
- Logical name space to provide a common naming convention across the images in the shared collection and across the remote storage systems where the images reside.
- Ability to organize the logical name space into sub-collections to simplify browsing and discovery of related images or files.
- Support for queries on the descriptive metadata
- Support for access controls to ensure data and metadata are not maliciously altered

[1] This work was supported in part by the NSF SCI0438741 Cyberinfrastructure project and the NSF National Virtual Observatory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation or the U.S. government.

- Remote procedures that can be used to extract metadata, or create image cutouts, or support transformations of the format.
- Support for replicas to improve availability, minimize risk of data loss, improve performance.
- Support for federation with other shared collections to enable the creation of global digital holdings
- Support for state information such as owner, version, audit trail, locks, backups, sticky bits for setting access controls from a parent collection, soft links to other images in the shared collection, deletion flags, synchronization flags for replicas, checksums, verification time stamps, creation time stamps, update time stamps.

The VOSpace interface also can be designed to manage latencies that are inherent in distributed environments, through the provision of bulk operations for metadata and data movement. Typical bulk operations aggregate data before transmission and use parallel I/O streams to minimize the transfer time. Finally, the VOSpace interface should support graceful interactions with network devices such as firewalls, load levelers, and virtual private networks. The network protocols used to implement bulk operations have to differentiate between client-initiated services and remote server-initiated services. The latter enable use of parallel I/O streams from behind firewalls.

2. VOStore:

The VOStore service can be implemented as a software server that is installed as application-level software at the storage repository. The VOStore server responds to commands from an access client or another VOStore server. A preferred design is for VOStore servers to support peer-to-peer communication. The VOStore server can be installed under the same Unix account as the owner of the files that are being accessed.

A simple VOStore interface would support:

- "Put" of files onto the storage system. The source of the files may be another VOStore server or a remote client.
- "Get" of files from the storage system. The files may be delivered to another VOStore server or to a remote client.
- Deletion of files from the storage system
- List of files on the storage system.
- Access to Unix state information such as owner, file name, and creation date.

The advantages of this VOStore server specification is that it can be implemented on top of existing Unix file systems without having to manage a separate metadata catalog. All read accesses are assumed to be to data that are publicly accessible. All write accesses are assumed to be through the account of the person who owns the data.

3. VOSpace:

The VOSpace service implements a metadata catalog to manage the logical name spaces, the shared collection state information, and the descriptive metadata that are generated when files are published. The VOSpace service corresponds to a shared collection that may be distributed across multiple VOStores. The files that are members of the shared collection are owned by an account associated with the VOSpace service.

This appears to impose an authentication barrier. How do files migrate from privately owned data in file systems to shared collections that are owned by a VOSpace account? Data grids manage this transformation through the concepts of registration and shadow links. A shadow link is a pointer to a file that resides on a remote VOStore instance. For operations to be performed upon the remote file, access permission must be given to the VOSpace account. Registration corresponds to the recursive loading of pointers into a VOSpace metadata catalog for the files that exist within a directory.

An example of this approach to migrating data from a private context into a shared collection was the replication of the DPOSS sky survey into a Storage Resource Broker (SRB) data grid. The DPOSS sky survey images resided at Caltech on the HPSS archival storage system. An account was established on the HPSS system for the SRB server. Access permission was then given to the SRB server account for all of the image in the DPOSS survey. A SRB server was installed on the HPSS system. Note that the metadata catalog into which the files were being registered resided at SDSC. No metadata catalog was installed at Caltech.

The SRB registration command was issued from a client running at SDSC, redirected by a SRB server at SDSC to the SRB server running at Caltech (peer-to-peer server architecture), and executed on the HPSS system. The entire DPOSS collection was registered into the SRB collection in 10 minutes. The time would have been shorter, but HPSS provided information for only one file at a time.

Once the files were registered into the SRB collection, then they could be replicated onto resources managed by the SRB over an arbitrarily long period of time.

The ability to register filesinto a VOSpace collections requires no additional capabilities in the VOStore interface.

4. VOSpace implementation

The Storage Resource Broker data grid provides a proof of concept that it is possible to build a viable VOSpace system. The SRB system consists of peer-to-peer servers that are installed at each storage repository where the shared data reside, and a metadata catalog that resides anywhere on the network linking the servers. The SRB server implements the VOStore interface functionality using standard Posix I/O functions. Actually, the set of operations include not only single file "get" and "put", but also a wide variety of bulk operations that deal with firewalls.

The metadata catalog manages both state information for the shared collection (replica locations, versions, checksums, owner, access controls, time stamps, etc.) and descriptive information. The SRB also supports the ability to write to remote storage systems through the GridFTP interface, and the ability to write files under a user account ID. Note that writing data under a user account ID means that the data cannot be shared until access permissions are established for the SRB shared collection account ID.

The design principles on which the SRB is based are:

- Latency management. The number of messages and the amount of data sent over wide area networks are minimized.
- Trust virtualization. Authentication, authorization, and audit trails are managed independently of the remote storage system.
- Data virtualization. The properties of the shared collection, including the name spaces used to describe the shared files are managed independently of the remote storage system.
- Collection management. The shared collection can be organized and managed as a collection hierarchy. The descriptive metadata can be extended dynamically, schema extension supports user-specified table structures for metadata, import and export of XML files is supported, and a template language for automated extraction of metadata is supported.
- Federation management. Multiple independent SRB data grids can cross-register name spaces, enabling the creation of hierarchies of shared collections. Each data grid retains control of their data, while enabling access from a user in a remote data grid under appropriate access controls. All

authentication information remains with the original home data grid of each user. This is similar to the Shibboleth model for authentication, but does not require redirection through http proxies.

For each of these functional areas, the SRB supports the associated logical name space, an extensive set of operations, and the associated state information that is generated by each operation. A representative set of capabilities is listed in Table 1 for the SRB.

	Logical naming	Standard operations	State information
Latency	Logical resource names	Load leveling	Quotas on storage and usage of storage
Management		Fault tolerant replication	Replication state
	Compound resources	File staging	Names for file system cache
		Automated access control setting	Sticky bits to inherit access controls of parent collection
		Client and server initiated parallel I/O on access	Creation time, update time
		Client and server initiated bulk file registration	
		Client and server initiated remote procedures	Location in SRB of remote procedures
		Client and server initiated bulk metadata load	
		Bulk delete - trash can	Deletion flag
		Automated checksum verification on load	
		Third party transfer	
		Store files in a logical container	
Trust	Logical user names	Add or delete user	User:Group:Zone
Virtualization		GSI authentication	Certificate authority location
		Challenge-response authentication	Encrypted user password
		Issue ticket-based authentication	Time to live and number of allowed accesses
	User roles	List user roles	Curate, audit, annotate, read, write, group administration, superuser, public
		Set access control by role for user	Access controls on users
	Group names	Set access control by role for group	Access controls on groups
		Set access control on metadata for user	Access controls on metadata
		Set access control on resource for user	Access controls on resources
		Turn on audit trails	Audit trails
		Enable client-based encryption	Encryption key
		Resolve error number	System log of all accesses
Data	Logical entity names	Define SRB physical file name structure	SRB physical file pathname structure
Virtualization		Load a file into SRB collection (Sput)	Physical location where SRB stores file
		Unload a file from a SRB collection (Sget)	
	Shadow links	Register existence of external file	Location of external file
		Register existence of external directory	Location of external directory
	Logical container names	Create container	Physical file in which data is aggregated
		Create checksum	Checksum
		Verify checksum	
		Synchronize replicas	Dirty bit for writes
		Synchronize remote files with SRB files	
		Synchronize SRB files with remote files	

		Synchronize SRB files between two SRB collections	
		Posix I/O - partial read and write	Replica location
		Delete file	
		Recursive directory registration	
		Register a file as a replica of existing file	Owner, size
		Create version	Version number
		Create backup	Backup time
		Lock a file	Lock status
		Register SQL command	
		Issue a registered SQL command	
		Create and issue a Datascope query	
		Register URL	
Collection Managment	Descriptive metadata	Extensible metadata	Descriptive metadata for SRB file
	Collection hierarchy	Create/delete subcollection	Parent collection identity
		Create collection metadata	Descriptive metadata for SRB collection
		Extensible schema	Table structure of metadata
		Create soft link between two logical files	Soft link
		Import of XML files	
		Export of XML and HTML files	
		Remote template-based metadata extraction	Location in SRB of templates
		Synchronize slave catalog with master catalog	Location of slave catalog
		Queries on descriptive and state information	
Federation Management	Distinguished zone names	Access zone authority to register zone name	Zone name and port number
	Zone authority name	User authentication by home zone	
		Cross-registration of resources between zones	
		Synchronization of user names between zones	
		Synchronization of file names between zones	
		Synchronization of metadata between zones	

Table 1. Storage Resource Broker logical name spaces, global data manipulation operations, and global state information for the functional areas of latency management, trust virtualization, data virtualization, collection management, and federation management.

A simple VOSpace implementation would provide a subset of the SRB capabilities by eliminating support for:

- Bulk operations
- Containers
- Ticket based authentication
- Separate access controls on metadata and resources
- Encryption
- Versions
- Backups
- File locks
- URL, SQL registration
- Datascope query access
- Extensible schema
- Slave metadata catalogs

The other features are already in use in astronomy data grids such as NOAO and in the Teragrid replication of sky survey image archives.