MySRB & SRB – Components of a Data Grid

Arcot Rajasekar, Michael Wan, Reagan Moore
San Diego Supercomputer Center, University of California at San Diego
{sekar,mwan,moore}@sdsc.edu

Abstract

Data Grids are becoming increasingly important in scientific communities for sharing large data collections and for archiving and disseminating them in a digital library framework. The Storage Resource Broker provides transparent virtualized middleware for sharing data across distributed, heterogeneous data resources separated by different administrative and security domains. The MySRB is a web-based interface to the SRB that provides a user-friendly interface to distributed collections brokered by the SRB. In this paper we briefly describe the use of the SRB infrastructure as tools in the Data Grid Architecture for building distributed data collections, digital libraries, and persistent archives. We also provide details about the mySRB and its functionalities.

1. Introduction

The "Grid" is a term used to describe the software infrastructure that links multiple computational resources such as people, computers, sensors and data [1]. The term "Data Grid" has come to denote a network of distributed storage resources, from archival systems, to caches, to databases, that are linked using a logical name space to create global, persistent identifiers. Examples of data grids can be found in the physics community [2,3,5,6], for climate prediction [4] and for ecological sciences [7]. More recently several projects have promoted the establishment of data grids for other communities such as astronomy [8], geography, and earthquake and plate tectonic systems [9], etc. Most of these data grids are under construction and represent different proto-typical systems for building distributed data management environments.

At SDSC we have developed software middleware, called the Storage Resource Broker [12], that can be used to build a data grid. The SRB together with MySRB, a web-based interface to the SRB, provides a suite of functionalities that can be used to implement data and information management systems. We discuss the SRB and MySRB infrastructure within the context of data grids, and demonstrate how data grids can be used to implement distributed data collections, digital libraries, and persistent archives.

2. Data Grid Architecture (DGA)

All Data grids form the core infrastructure for building data management systems that span multiple administration domains, multiple types of storage systems, and multiple types of data access environments. We can characterize data management environments as distributed data collections, digital libraries and persistent archives. Distributed data collections provide a single name space for referencing data stored on multiple storage systems, typically within the same administration domain. Digital libraries integrate remote archival storage systems into a data collection, while providing discovery and manipulation services. Persistent archives support the migration of data collections onto new technologies, while preserving the ability to organize, discover, and access data. Each of these systems builds upon the capabilities provided by the lower level system, and all build upon data grids for managing distributed resources. We can characterize these capabilities as follows:

Distributed data collection capabilities [17]:

- Integrate Data Collections and Associated Metadata. As part of any DGA, we assume that the digital entities within a data collection will be described by attributes that characterize administrative, structural, provenance, and discipline-specific information. A data grid should provide mechanisms to support attributes associated with each registered digital entity.
- Handle Multiplicity of Platforms, Resource & Data Types. The DGA network should handle diverse computational and storage resources. In the network, one should be able to access files on a super computer such as a IBM SP-2 or a desktop system (say an SGI) or a lap top running Linux or Windows OS. The DGA should support access from arbitrary types of compute platforms.

- Seamless access to data and information stored within the DGA. The data from various collections at participating sites will be stored in archival storage systems (such as HPSS, DMF, ADSM, UniTree), file systems (Unix, NTFS, Linux), and databases (Oracle, Sybase, DB2) . Researchers at remote sites should be able to access these data as though they were accessing a local dataset, including support for reading and writing files.

Digital Library capabilities [18]:

- Handle Seamless Authentication. A digital library typically manages digital entities under a collection or community ID. To access data from a remote archive, the DGA should be able to manage the authentication of a user to the data handling environment, the authorization of the user for access to a digital entity, and the authentication of the data handling system to the remote archive. The DGA should be able to provide access to the user to all the storage systems with a single sign on authentication.
- Virtual organization structure for data and information based on a digital library framework. Even though data will be stored at multiple sites, it would help users if the data are organized according to some logical (context-dependent) structure with an easy navigational aid. Hence, the DGA has to provide means to group data into collections (actually hierarchies of collections) and provide management facilities for the same.
- Handle Dataset Scaling in size and number. The sizes and numbers of datasets involved in any DGA will grow in the coming years Hence any solution for the data grid should be scalable to handle millions of datasets, hundreds of Terabytes as well as large files that are tens of Gigabytes in size. Support is also needed for aggregating small data files into physical blocks called containers for storage into archives, and for decreasing latency when accessed over a wide area network.

Persistent Archive capabilities [14,15,16] :

- Replication of Data: For reasons of fault tolerance, disaster recovery and load balancing it will be useful for data to be replicated across distributed resources. Moreover, the consistency of the replicas should be maintained with very little effort on the part of the users.
- Version Control: Since datasets may evolve over time, providing distributed version control will help in collaborative data sharing. This includes facilities for locking and checking out files.
- Handle Access Control and provide Auditing Facilities. In some communities, data need to be guarded so that access to them is given only to selected and relevant people. Moreover, the selection should be done by the owner of the data. The DGA should be able to control access at multiple levels (collections, datasets, resources, etc) for users and user groups beyond that offered by file systems. Moreover, in some cases, it may be necessary to audit usage of the collections/datasets. Hence, auditing facilities will be needed as part of the framework.

Data grids can provide support for each of the above capabilities, making it possible to extend current data collection, digital library, and persistent archive technology into distributed data management environments.

3. The Storage Resource Broker

The SDSC Storage Resource Broker (SRB) is client-server middleware that uses collections to build a logical name space for identifying distributed data [12,19]. The SRB provides a means to organize information stored on multiple heterogeneous systems into logical collections for ease of use. The SRB, in conjunction with the Meta data Catalog [13], supports location transparency by accessing data sets and resources based on their attributes rather than their names or physical locations [20]. The SRB provides access to data stored on archival resources such as HPSS, UniTree and ADSM, file systems such as the Unix File System, NT File System and Mac OSX File System and databases such as Oracle, DB2, and Sybase. The SRB provides a logical representation for describing storage systems, digital file objects, and collections and provides specific features for use in digital libraries, persistent archive systems and collection management systems. SRB also provides capabilities to store replicas of data, for authenticating users, controlling access to documents and collections, and auditing accesses. SRB also privides a facility for co-locating data together using containers. One can view containers as tarfiles but with more flexibility in accessing and updating files. The SRB can also store user-defined metadata at the collection and object level and provides search capabilities based on these metadata.

SRB is a *federated server* system, with each SRB server managing/brokering a set of storage resources. The federated SRB implementation provides unique advantages:

1) Location transparency - Users can connect to any SRB server to access data from any other SRB server , and discover data sets by either a logical path name or by collection attributes.

2) Improved reliability and availability - data may be replicated in different storage systems on different hosts under control of different SRB servers to provide load balancing .

3) Logistical and administrative reasons - different storage systems may be run on different hosts under different security protocols, through use of a single sign-on environment and Access Control Lists maintained for each digital entity;

4) Fault tolerance – data can be accessed by the global persistent identifier, with the system automatically redirecting access to a replica on a separate storage system when the first storage system is unavailable.

5) Integrated data access – SRB provides the same mechanisms for accessing data in distributed caches and archives, making it possible to integrate access to back-up copies into the data management environment

6) Persistence – data can be replicated onto new storage systems by a recursive directory movement command, without changing the name by which the data is discovered and accessed. This makes it possible to migrate collections onto new resources without affecting access.

The SRB has been implemented on multiple platforms including IBM AIX, Sun, SGI, Linux, Cray T3E and C90, Windows NT, 2000, Me, Mac OSX, etc. The SRB has been used in several efforts to develop infrastructure for GRID technologies, including the Biomedical Information Research Network (NIH) NSF/DOE Particle Physics Data Grid [2], DOE ASCI [10], NASA Information Power Grid [11] and NSF GrPhyN[5]. The SRB also has been used for handling large-scale data collections, including the 2-Micron All Sky Survey data (10 TB comprising 5 million files in a digital library), NPACI data collections, the Digital Embryo collection (a digital library of images) and LTER hyper-spectral datasets (a distributed data collection). More details on the SRB can be found at http://www.npaci.edu/DICE/SRB/.

4. MySRB– a web-based interface to the SRB

MySRB is a web-oriented interface for accessing the data and metadata brokered by the SRB, that allows users to share their scientific data collections with their colleagues in a secure fashion. It provides a system where users can organize their static files and dynamic digital objects (virtual data) according to logical cataloging schemes independently of the physical location and formats of the files and also associate queriable metadata with these files.

MySRB provides three primary functionalities:

- collection and file management: operations collection, maintenance and deletion, operations for data creation, data ingestion, reload and registration, data replication and movement, access control, and for data deletion. Versioning and locking functions are under implementation.

- metadata handling: operations for ingestion, extraction, copy, maintenance, update, and deletion of user-defined and standardized metadata. Standardized metadata might be based on lists of elements such as the Dublin Core, or might be from definitions based on Semantic Web.

- access and display of files and metadata: functions for browsing files in the collection hierarchy and to search and query using system-level, user-defined and standard metadata.

MySRB uses the secure-http (https) protocol with 128-bit RSA authentication. Each session to MySRB is given a unique session key (stored as an in-memory cookie at the Browser). These session keys have a maximum time-limit set on them (currently 60 minutes). MySRB also performs security checks on the session keys when validating a user request.

The web-browser interface for MySRB uses a split-window, shown in Figure 1. The small top-window is used to display metadata about data objects and collections, and the larger bottom-window is used for displaying elements in a collection or for displaying data objects accessed by the user. Hence, when a user "opens" a file, the attributes about the file are displayed along with the contents of the file. We show a screenshot of the MySRB interface in Figure 2 that is used to enter metadata. Default attribute values can be specified, as well as restricted vocabularies for attribute values, and as well as user-defined metadata.

We illustrate the functionalities supported by MySRB through an exemplar scenario:

Consider a curator who wants to form a new collection called "Avian Culture" under an existing "Cultures" collection. Her aim is to gather in one `folder' all documents and multi-media available on the topic even though they might be located as distributed files, images, and movies stored on diverse media-formats in file systems, archives, databases and web-sites. Some of the files would be under the control of the collection being created but others might be owned and curated by outside administrators with only links provided to them. She would also like to allow other curators to include their own materials into the collection. But she wants to have them include some minimal set of metadata based on entities defined under "MetaCore for Cultures" which she has augmented with more attributes relevant to her specialized topic. Also, she would like a set of selected users to add additional metadata for the collected items as and when they come across more information. Moreover, she would like users to add their own comments, ratings, errata and dialogues and annotations which will make the collection richer and more useful. Also an important criteria to her, is to include multi-modal relationships among the collection items so that one can link the objects in many ways for ease of browsing. Finally, she would like the public users to be able to access her collection by browsing in a pre-determined fashion (which is done by organizing the collections as sub-collections as well as through multi-modal relationships), and/or search/query the collection using the rich mix of metadata based on standardized meta data, curatorial meta data, user annotations, ratings and dialogues.

All the above operations are facilitated through the MySRB interface. The collections and sub-collections structure provides a means to organize the objects in a hierarchical fashion. Metadata at the collection-level can be used to provide (queriable) information about the collections and sub-collections as well as to enforce metadata that need to be provided when new items are added to the collection. Metadata at object level and collection level can be used to encode multi-modal relationships between the objects in the collection as well as to provide links to other documents and data outside the collection. The metadata stored in MySRB are not just entity-value pairs, but have a richer structure including associated ontology, units of the meta-value, groupings of the meta entities in schemas and sub-groupings. (The ontology part is under development). MySRB also allows for storing annotations to capture dialogues, comments, ratings and user annotations and errata. MySRB supports a rich set of metadata management operations to help curators, users and public to ingest, maintain and access multiple kinds of metadata. A rich set of access control mechanisms provides a role-based access matrix from curator to public. The SRB facilitates federated and seamless access to remote storage from web-servers and file servers to tape archives and databases. It also provides for copying data as well as remote linking to enable read-only access to data not curated by current collection. Also, the SRB provides a feature to link objects and sub-collections across multiple collections without copying them. Finally SRB provides a replication management capability that can be used to provide fault-tolerance, load balancing and archival functionalities.

5. Features in MySRB

We briefly discuss the mySRB in terms of its data movement capabilities and metadata management features

Data Movement Operations:

At the collection-level, a user can ingest a file into SRB or create new sub-collection through the mySRB interface. At ingestion time, the user can choose the logical resource that will be used for storing in SRB. The logical resource can be a single physical resource (say a Unix or NT file system, database, or archival file system) or it can be a logical resource that ties together two or more physical resources. In such a case, the file is replicated and stored in the underlying physical resources. For example, consider a logical resource logrsrc1 which consists of two resources: unix-sdsc, a unix file system at SDSC and hpss-caltech, a HPSS archival system at CalTech; then storing a file into logrsrc1 will ingest the file into both physical resources, unix-sdsc and hpss-caltech, synchronously and the two copies will be shown as two replicas of the same SRB object. During retrieval, the user can ask for a particular copy or or let SRB choose its own
access for the file.

Instead of specifying the resource, the user can specify a container when ingesting the file. In this case, the file is added to the container. Note that a container specification on ingestion overrides a resource specification. During ingestion, the user can specify the data type as well as any metadata that is required by the collection as well as a few user-defined metadata (metadata is discussed later in the section). MySRB uses the file-browse mechanism of web Browsers to identify the local file that need to be ingested. Files from Unix, Windows and Macintosh have been successfully ingested using mySRB. At this stage, only single file ingestion is supported. Apart from

ingesting a file, a user can reingest a file (i.e., all metadata associated with the file by the SRB are still linked to it) and edit a file, if it is a small ASCII file (the edit facility is allowed only for a few data types).

Apart from ingesting a file, a user can register SRB objects where no physical copy of the file is maintained or controlled by the SRB but a pointer to a physical location is maintained. There are five such types of objects that can be registered through mySRB:

1. A file that can exist either in a file system, an archival storage system or as a LOB in a database system. In this case, the user specifies the physical resource in which the file exist and the path name to the file in that resource. The user can perform all operations that mySRB offers, including deletion on registered files. Since the file is not fully under SRB's control, the file size and other characteristics might change without SRB being aware of these changes.
2. A directory in a file system or an archival storage system. The user specifies the physical resource and the directory path name. The mySRB registers this path name as a 'shadow directory object' (i.e., the cone of files under this directory is visible through this object), and provides all operations that can be performed by the user on a file except new file ingestion into the directory structure or update/deletion of files. These functionalities have some security implications but might be supported in a later version if these concerns are resolved.
3. A SQL query for a database resource. The user specifies a SQL query which can be either partial (i.e., the user can specify reminder of the query at retrieval time) or a full SQL query. Note that for security reasons, we recommend that one register only 'select' commands and also have at least a partial query starting with select as part of the SQL. The select statement can be any query supported by the underlying database, including table joins, functions, stored-procedures, sub-queries and union queries (limitation of size might apply). The query is executed at retrieval time, and is not stored on registration. Hence the answer to the query can vary with time. During registration of the SQL, the user can specify the template to be used for pretty-printing the retrieved table. The mySRB supports three built-in templates that can be used: the first template HTMLREL, prints the result as a relational table in HTML format, the second template HTMLNEST, prints the result as a nested table in HTML, and the third template XMLREL, prints the result in XML using a simple DTD. Apart from the built-in template, the user can specify their own 'style-sheet'. In this case, the user specifies a file already in SRB as the style-sheet file. Currently the style-sheet is written in T-language, an interpreted language native to SRB that supports rule-based data extraction and style-sheet for data organization. Support for other style-sheet languages such as XSLT will be provided in a later version. (Note: for additional information on T-language please refer to the primer given as part of the SRB package.) Deletion operation on this SRB object just removes the query from the SRB (and also any associated metadata and annotations) but does not change or delete tables in the underlying database. Currently mySRB does not support ingestion into databases (note that the SRB allows ingestion through command line and API), but the feature might be supported in a later version.
4. A URL. The user can specify any URL including ftp calls and cgi queries. On retrieval, the contents of the URL are retrieved and displayed. The contents of the URL are not stored in the SRB on registration. Deletion operation just removes the URL and any associated metadata from SRB and does not damage the contents of the URL at its physical location.
5. A method object or virtual data. The user can specify two types of registered method objects. The first type of method object runs an executable program that is invoked by the SRB as a remote proxy command. A proxy command is an executable that is available in the bin directory of a SRB server and is made available for execution by the SRB administrator (users have to ask a SRB administrator to place an object in a, possibly remote, SRB bin directory; this is done as a security precaution). When the method object is 'accessed', the command is executed on the remote server and any results of that execution piped back to the browser. The user can provide command-line parameters at the invocation. As an example, one can register a method object that invokes a 'srbps' command on a remote host. The 'srbps' shows the process status similar to 'ps' command in Unix, and the result is sent back to mySRB browser. The second method is an invocation of a proxy function inside SRB. One can compile user-defined functions in SRB and can invoke them using this feature. For example the metadata extraction function (explained later) is implemented in this manner.

Apart from ingesting or registering files into mySRB, the user can also perform other data movement/maintenance operations. We briefly discuss them below:

replicate: In mySRB, a user can replicate any file that is either ingested into the SRB or one that has been registered into SRB. Files inside a registered directory is not replicable. When replicating, the user specifies the resource that will be used for storing the replica. The new replica inherits all metadata associated with its siblings. A replica number is uniquely determined for the new replica and is displayed for the user in the listing. At this stage, mySRB does not support replication of files inside a container using this operation. Replication of a container (and its objects) is done by the SRB system

using semantics associated with the logical resource specification of the container.

register replicate: When a SRB object is a registered directory, URL, or SQL, then another object which has similar characteristics can be registered as a replicate. For example, if a SQL object which queries an oracle database, say dlib1, is registered in SRB, one can also register as a replicate of the first object another SQL object which queries another database, say a db2 databese dlib2. This might imply that they are equal (in some sense) and either queries will give the same result. Note that SRB does not check whether a registered replica is really an equal of the other copy. One can use this technique to register another object as a semantically equal copy of each other. For example, two SQL queries one giving an HTML output and another giving an XML output can be registered as replicas.

ingest replica: For a SRB object (ingested or registered), one can ingest another file as a replicate. This is very useful when you want two different files in SRB that are syntactically different but semantically equal (eg. a tiff file and a gif file of the same image). Note that SRB does not check for syntactic or semantic equality.

copy: A SRB collection, file or registered file can be copied as another SRB file or collection in another collection possibly with a new name. Currently we do not support copy of URL, SQL or method objects. The copy command does not copy any user-defined metadata or annotations for the new copy. We discuss these operations later. A copy is different from the replication operation because these two objects are considered to be entirely different and unconnected. Notions of synchronizations, trying for alternate replicas for retrieval and other operations and semantics associated with replicas do not apply to copied objects. The user specifies the new resource, path name and collection for the copy operation.

move: Both files (ingested or registered) and sub-collections in SRB can be moved from one collection to another. The user-defined metadata remains unchanged. This move is considered a logical move. Another type of move supported by the mySRB is a physical move of the object. This is possible only for files ingested into SRB resources (container-based files cannot be moved using this operation). In this case, the user provides the resource and path names of the new location for the file.

link: One can link a SRB object (ingested or registered) in another collection. The operation is similar to soft linking in Unix. The access control of the original object is inherited by the linked object. Metadata and annotations associated with the original object can be viewed as part of the link object's metadata but does not allow modification of the original object's metadata. One can associate metadata and annotations with the link object apart from those available for the linked. One can also link a collection as a sub-collection of another collection. One can have more than one link to the same data (though replicas are not allowed). Chaining of links is not allowed. An attempt to link to another link object will result in a direct link to the parent object. Linking will be supported in the next version of SRB release.

delete: A user can delete an ingested or registered file using the delete operation. A registered directory, SQL, URL or method object are unlinked without any deletion of the physical object. The objects or links when deleted is done one replica at a time and when the last replica is deleted all the metadata and annotations are also deleted.
A linked file cannot be deleted through the link; a delete operation on a link basically performs an unlink operation.

lock, pin, checkout: Using mySRB, an object can be locked so that operations on it are restricted. Two types of locks are supported: a 'shared' lock which locks the object from being written to by any user other than the locking user but reads from the object and associated metadata are allowed, and 'exclusive' lock which allows no interactions with the object. A lock placed by a user has an expiry date at which time it gets unlocked. A user-driven unlock operation is also supported. Pin operation makes sure that a SRB object does not get deleted from a particular resource. This is useful for pinning a file in a cache resource from being purged by SRB when performing cache management. An expiry time is also associated with pins and an explicit unpin operation is also supported. Checkout and checkin operations provide very crude forms of version control in mySRB. A checkout by a user disallows any changes to be made to that object and when checkin occurs, the older version of the object is still maintained as an earlier version with a distinct version number. Note that this is a very rudimentary version control and will be improved in later versions. These operations are implemented in mySRB but are not available in the current version of the SRB (1.1.8); they will be supported in the next version of the SRB release.

Metadata Operations:

The mySRB interface provides a very rich set of operations for creating, maintaining, viewing and searching different types of metadata for SRB objects as well as collections. There are five types of metadata in mySRB:

1. system-defined metadata,
2. user-defined metadata,
3. type-oriented (domain-oriented) metadata,
4. file-based metadata, and
5. annotations and commentary metadata.

The system-defined metadata is created and maintained by the SRB system and the user can view them and also use them in their search mechanism. User-defined and type-oriented metadata for SRB files and objects are descriptive in nature and are made of name, value and units triplets. The metadata for a SRB collection can be of two types: descriptive and structural. Descriptive metadata are tripletswhich describe the content of the collection where as the structural metadata describe metadata that is required/suggested by the collection creator/curator for objects ingested or registered in the collection. The structural metadata has two additional parameters, default values and comments for explaining the metadata and its requirements. For these type of metadata, one can associate either no default values, or one default value or a set of reserved keywords which appear as a drop-down list in mySRB. Also the creator can designate zero or more of these metadata as mandatory wherein the file ingestor is required to provide a value for the metadata.

There are four ways of associating user-defined metadata in mySRB. The first method allows the user to associate metadata when ingesting or registering an object, or when creating a new sub-collection. The second method is to invoke the insert metadata function which provides a form for the operation. This operation can be performed as many times as required and hence there is no limits for the number of metadata associated with a SRB object or collection. The third method is to copy metadata from other SRB objects or collections.

The fourth method is to extract metadata from an extraction method associated with the data-type of the file. The metadata can be extracted from the object itself (eg. FITS files, HTML files) or one can extract the metadata from a second SRB object and associate the metadata to the first object (eg. AMICO image metadata with XML metadata files, or DICOM image metadata from separate header files). One can associate more than one metadata extraction method for a data-type and the user is allowed to choose one at the time of metadata creation. If necessary, more than one method can be applied to the same object to extract different metadata. Metadata extraction methods can be written in T-language, which has a simple form of rules for identifying metadata values and associating them with metadata names.

The type-oriented metadata are pre-defined sets of metadata elements that can be associated with the SRB objects through their data types or for all SRB objects. For example, Dublin Core metadata can be associated with any SRB object and an entry form for Dublin Core can be invoked when needed. Data-type designated metadata can be ingested for SRB objects of particular type and can be done through forms, by copying from other objects and/or by extracting through metadata extraction methods. User-defined metadata and type-oriented metadata can be ingested only by users who have 'ownership' permission for the SRB object or collection.

File-based metadata is as name suggests a file in SRB that is associated as a metadata-carrying file for another SRB object. This metadata is used only for viewing and cannot take part in querying (at the current time). One can associate the same file to be a metadata file for more than one SRB object. Currently triplets are the only form of metadata supported in this manner. XML-based metadata will be supported in a later release.

Annotations and commentary metadata are useful for associating free-form metadata to a SRB object. They can be used for providing notes, comments, errata, queries and answers, annotations, memoranda, etc. These have a type/location associated with them and the timestamp and the annotation writer's name. Unlike other types of metadata, the annotations and commentary can be inserted by any user with a read permission on the object.

Having seen all the different types of metadata ingestion methods, one can be very creative in the type of metadata being ingested. Currently one can associate a URL as a metadata and if the URL is designated as being of 'inlineable' type then the mySRB shows the contents of the URL. One can also associate other SRB objects as related to this object and in that case, a reference is provided as a clickable hot-link in mySRB. If this SRB object isdesignated as 'inlineable', mySRB shows the content of the object. This is useful when showing thumb-nail images for larger images or when showing some properties that are stored in a database. Other creative modes of metadata support will be implemented in future mySRB releases.

Metadata in mySRB can be viewed in two ways. When a user selects an object for viewing, then the associated metadata is shown in a split screen on the browser. Hence the user can see both the data and the metadata at the same time. In the case of collections, the user-defined metadata is shown in one part of the screen and the collection listing with some of the system-metadata is shown in the other part of the screen. In the second method, the user can select to just view the metadata for

an object. In the case of collections users have a choice of seeing the descriptive, structural or all metadata.

The importance of metadata in SRB comes from the queriability of the metadata. MySRB provides a query interface where one can either query only the user-defined and type-defined metadata or query also with annotations and some system-defined metadata. When a user selects the mySRB icon in a collection-page of mySRB, it opens a new page with a set of query conditions where each condition has four parts: a metadata name part which is a drop-down menu containing all the metadata names that are queryable in that collection and every collection in the hierarchy under the collection. Hence, one can query across collections by being above the collections. The second part is a comparison operator where one can choose =,>,<,<=,>=,<>,like, not like, etc. The third part is a text box where the user can provide values for the comparison condition. The fourth part is a checkbox where a user can check if the user wants to see the values for a metadata-name in the query result listing. One can check the box of a metadata name without using it as part of any query condition. The query is taken as a conjunctive query in the current implementation, i.e., an AND of all the condition is used for search purposes. The result of the query is a listing of SRB objects that satisfy the search conditions.

Apart from creation, viewing and querying of metadata, MySRB provides functionality for updating, copying and deleting user-defined metadata and annotations.

Apart from these operations, the MySRB interface provides additional functionalities such as user registration, access to resource, user and container metadata, ability to navigate the collection hierarchy and on-line help. The MySRB interface is available at https://srb.npaci.edu/mySRB.html.

6. Conclusion

Data Grids are becoming increasingly important in scientific communities for sharing large data collections and for archiving and disseminating them in a digital library framework. The Storage Resource Broker provides transparent virtualized middleware for sharing data across distributed, heterogeneous data resources separated by different administrative and security domains. The MySRB is a web-based interface to the SRB that provides a user-friendly interface to distributed collections brokered by the SRB. In this paper we saw brief descriptions of the use of the SRB and the MySRB infrastructure as potent tools in the Data Grid Architecture for building distributed data collections, digital libraries, and persistent archives.

7. References

[1] Foster, I., and Kesselman, C., (1999) “The Grid: Blueprint for a New Computing Infrastructure,” Morgan Kaufmann.

[2] PPDG, (1999) “The Particle Physics Data Grid”, (http://www.ppdg.net/, http://www.cacr.caltech.edu/ppdg/).

[3] Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H., and Stockinger, K. (2000) “Data Management in an International Data Grid Project,” *IEEE/ACM International Workshop on Grid Computing Grid’2000*, Bangalore, India 17-20 December 2000. (http://www.eu-datagrid.org/grid/papers/data_mgt_grid2000.pdf).

[4] Hammond, S., (1999). "Prototyping an Earth System Grid", at the *Workshop on Advanced Networking Infrastructure Needs in Atmospheric and Related Sciences*, National Center for Atmospheric Research, Boulder CO, 03 June 1999. (http://www.scd.ucar.edu/css/esg/presentations/nlanr/index.htm).

[5] GriPhyN, (2000) “The Grid Physics Network”, (http://www.griphyn.org/proj-desc1.0.html).

[6] NEES, (2000) “Network for Earthquake Engineering Simulation”, (http://www.eng.nsf.gov/nees/).

[7] KNB, (1999) “The Knowledge Network for Biocomplexity”, (http://knb.ecoinformatics.org/).

[8] NVO, (2001) “National Virtual Observatory”, (http://www.srl.caltech.edu/nvo/).

[9] EarthScope, (2001) “EarthScope”, (http://www.earthscope.org/).

[10] ASCI, (1999) “Accelarated Strategic Computing Initiative”, A DOE Project, (http://www.llnl.gov/asci/).

[11] IPG, (2000) “Information Power Grid”, A NASA Project, (http://www.ipg.nasa.gov/).

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

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

[14] Rajasekar, A., R. Marciano, R. Moore, (1999), "Collection Based Persistent Archives," Proceedings of the 16th IEEE Symposium on Mass Storage Systems, March 1999.

[15] Moore, R., C. Baru, A. Gupta, B. Ludaescher, R. Marciano, A. Rajasekar, (1999), "Collection-Based long-Term Preservation," GA-A23183, report to National Archives and Records Administration, June, 1999.

[16] Moore, R., C. Baru, A. Rajasekar, B. Ludascher, R. Marciano, M. Wan, W. Schroeder, and A. Gupta, (2000), "Collection-Based Persistent Digital Archives – Parts 1& 2", D-Lib Magazine, April/March 2000, http://www.dlib.org/

[17] Moore, R., (2001a) "Knowledge-based Grids," Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Ninth Goddard Conference on Mass Storage Systems and Technologies, San Diego, April 2001.

[18] Moore, R., (2001b) "Knowledge-Based Data Management for Digital Libraries", NIT2001, Beijing, China, May 2001

[19] Moore R., and A. Rajasekar, (2001) "Data and Metadata Collections for Scientific Applications", High Performance Computing and Networking (HPCN 2001), Amsterdam, NL, June 2001.

[20] 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.

Figure 1: SRB Main page showing the Collections with different objects and Operations

Figure 2: File Ingestion Page with Metadata for Dublin Core Attributes and other user-defined attributes.