Design and Implementation of SRB Matrix

GRADUATE PROJECT FINAL REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Ullas Kapadia
Spring 2005

Committee Members

Dr. Carl Steidley
Committee Chairperson

Dr. Jennifer Smith-Engle
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

This project consists of the design and implementation of data management capabilities for development of infrastructure to support Earthquake Seismic Hazard assessments. It was accomplished developing Web services for Service Oriented Architecture that can be used by other developers for San Diego Supercomputer Center’s (SDSC) Storage Resource Broker (SRB). SRB is a client-server middle-ware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. This project is service oriented computing where software components are offered as services which interact with each other seamlessly and transparently. This system will be used at the San Diego Supercomputing Center, UCSD, San Diego - California.
# TABLE OF CONTENTS

Abstract .......................................................................................................................... ii

... .................................................................................................................................... iii

Table of Contents ........................................................................................................... iii

List of Figures ................................................................................................................ v

List of Tables ................................................................................................................... vi

1. Introduction and Background ..................................................................................... 1

1.1 Web Services .............................................................................................................. 1

1.2 SRB ............................................................................................................................ 2

1.3 Need for SRB Web Services ..................................................................................... 4

2. SRB Matrix ................................................................................................................ 7

2.1 Overview ................................................................................................................... 7

2.2 Requirements of SRB Matrix ................................................................................... 8

2.2.1 SRB Web Service Server .................................................................................... 8

2.2.2 Data Management ............................................................................................... 8

2.2.3 Security and Support for Computer Applications .............................................. 9

2.2.4 Scalability ............................................................................................................ 10

2.2.5 Change is Inevitable ........................................................................................... 10

2.2.6 Ability to Use the Existing Environment ............................................................ 11

2.3 SRB Matrix Development ....................................................................................... 11

2.3.1 Technologies Used for the Development ........................................................... 11

2.4 SRB Matrix Server ................................................................................................... 12

2.5 SRB Matrix Client ................................................................................................... 14

2.6 Description of User Interface .................................................................................. 17

2.6.1 SRB User Information ....................................................................................... 17

2.6.2 Create Connection .............................................................................................. 18

2.6.3 Create a Matrix Ticket ....................................................................................... 19

2.6.4 Authenticate a Matrix Ticket .............................................................................. 20

2.7 Web Service: Create a Collection .......................................................................... 20

2.8 Web Service: List a Collection ................................................................................. 21

2.9 Web Service: Delete a Collection ............................................................................ 21

2.10 Web Service: File Upload to SRB ......................................................................... 21

2.11 Web Service: File Download from SRB ................................................................. 22

2.12 Web Service: Delete File ....................................................................................... 22
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Number</th>
<th>Figure Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SRB Web Services Implementation in Conjunction with Current SRB</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Architecture and Implementation of SRB Matrix Server</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Creating an Environment Variable</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>Setting Up the Classpath</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>SRB Matrix Project Environment</td>
<td>26</td>
</tr>
<tr>
<td>6</td>
<td>Publish, Find and Bind.</td>
<td>29</td>
</tr>
<tr>
<td>7</td>
<td>System Architecture.</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>Matrix API</td>
<td>31</td>
</tr>
<tr>
<td>9</td>
<td>Overview Page of the API document</td>
<td>31</td>
</tr>
</tbody>
</table>
# LIST OF TABLES

<table>
<thead>
<tr>
<th></th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>API Overview</td>
<td>36</td>
</tr>
<tr>
<td>II</td>
<td>Package edu.sdsc.matrix.client</td>
<td>38</td>
</tr>
<tr>
<td>III</td>
<td>Package edu.sdsc.matrix.util</td>
<td>38</td>
</tr>
</tbody>
</table>
1. INTRODUCTION AND BACKGROUND

1.1 Web Services

A neutral description of a Web service is that: “A Web service is a platform and implementation independent software component:

- Described using a service description language
- Published to a registry of services
- Discovered through a standard mechanism (at runtime or design time)
- Invoked through a declared API (Application Program Interface) usually over a network
- Composed with other services” [Graham 2002].

From a technical perspective, the term ‘Web services’ is a marketing terminology representing the natural evolution of component based programming - extending the traditional model of retrieving components needed by a program to include not only the libraries accessible by a local program but also code resident on remotely networked computers. The fundamental difference between this model and traditional Distributed Computing Environments (DCE) such as CORBA (Common Object Request Broker Architecture) and RMI (Remote Method Invocation) is its congenital disposition to consider the components being accessed to be spread across the Internet (under possibly differing administrative entities) [Haneef 2002].

The traditional Distributed Computing Environment (DCE) solutions emphasize Remote Procedure Calls (RPCs), but the norm is the need for loosely coupled applications (where components may be searched or changed as needed). These need-
based solutions were solutions outside the scope of traditional distributed computing approaches:

- Support for both document-centric messaging and RPCs
- Transport encoded data from both applications and business documents
- Work over open Internet protocols such as HTTP and SMTP

Although technologies such as CORBA, COM and RMI are good technologies for tying together distributed objects on a corporate network, Web services are better suited for integration of e-business applications across multiple administrative domains.

Another feature of Web Service, which makes it a more flexible medium for cross platform communication, is its utilization of XML for messaging and its data transport mechanism. XML's text-based form eliminates byte ordering concerns. The expressive nature of attributes and nested elements in XML makes it easier to represent complex data types than to represent them in traditional binary format. In other words, XML makes working with arbitrary data easier [Haneef 2002].

1.2 SRB

SRB, or Storage Resource Broker, is a client-server based middle-ware implemented at the San Diego Supercomputer Center (SDSC) to provide a uniform access interface to different types of storage devices. SRB provides a uniform API that can be used to connect to heterogeneous resources that may be distributed and to access data sets that may be replicated. MCAT, or Meta Data Catalog, is a Meta data repository system implemented at SDSC to provide a mechanism for storing and querying system-level and domain-dependent Meta data using a uniform interface. MCAT provides a
resource and data set discovery mechanism that can be effectively used to identify and discover resources and data sets of interest using a combination of their characteristic attributes instead of their physical names and/or locations. Meta data are information about data.

SRB, in conjunction with Meta Data Catalog (MCAT), provides a means for accessing data sets and resources through querying their attributes instead of knowing their physical names and/or locations. To use SRB one needs to be a registered SRB user. One can register by sending an email to an SRB administrator. If there is a local SRB system administrator then requests should be to him/her; otherwise applications could be made to srb@sdsc.edu. There is one mechanism, called ticket-based access that allows unregistered users to access read-only data sets stored under SRB's control. Owners of data sets can issue tickets on their data sets that can allow everyone to access the data sets. Separate APIs have been developed that can be used by unregistered users.

SRB has been ported onto a variety of platforms including AIX (ex. SP-2 machines), Solaris, SunOS, SGI, DEC OSF, Cray C-90, Cray T-90 and Cray T3E. SRB Version 1.1 supports several storage devices for storing files. In file-based systems, SRB supports access to Unix File System, IBM's UniTree and HPSS. In LOB (Large-OBject) database systems, SRB supports file-type access to LOB via DB2, Illustra and Oracle. SRB provides access to remote storage systems through a proxy mechanism. When one stores a data set under SRB, the data set is stored and accessed by SRB, which acts as a proxy for the user. Due to this mechanism, a user can store data sets on remote storage systems without having personal accounts at these sites. In this mode, SRB acts as a 'system privileged proxy' user. The above proxy mode also allows for SRB-SRB
authentication enabling servers to access files that are under the control of another SRB server.

There are several different methods by which one can access SRB/MCAT functionality. The most comprehensive method is the programmatic API, which provides a C routine library that can be linked with any application program. Next, the S-commands interface provides a Unix-type command level interface, which can be used from a shell prompt. The third method is a graphic interface, called SRB Tool, which provides a point-and-click interface to SRB. Finally, there is a set of C-based programmatic APIs that provides an interface for web-based applications.

1.3 Need for SRB Web Services

There is an interest among the Grid community to use Web services for their mundane data management operations on the Grid. Some of the potential projects that will benefit using SRB Web services include TeraGrid, GriPhyN (Grid Physics Network) and Southern California Earthquake Consortium (SCEC). SRB Web services form a part of the preliminary efforts to create a Data Grid Management System (DGMS) at the SDSC/NPACI (National Partnership for Advanced Computational Infrastructure). The Final System will include:

- Synchronous and Asynchronous Web services,

- An event-triggered system to support asynchronous Web service invocations,
• Support for virtual data or virtual services using user-defined and application specific Meta data,

• Flexibility for the data grid clients to choose between protocols for transport of local data to the grids. Some data transport protocols include (SRB-RPC, FTP, GFTP, HTTP) and

• Each data grid operation will be implemented as a separate service having a separate interface and URN (Uniform Resource Name). Resources in the World Wide Web are named using Uniform Resource Identifiers or URIs. The most common and well-known form of URI is the Uniform Resource Locator (URL). A URN is also an URI, but differs from an URL in that it only identifies a web resource. A URN does not indicate the location of a resource, nor does it contain other information that might change in the future.

Work is being done on the schema and protocols which will be based on W3C standards or recommendations regarding Web services like Simple Object Access Protocol (SOAP) 1.1, XML-Schema, Web Service Definition Language (WSDL) 1.1, etc. [Arun 2001]. Figure 1. illustrates how the SRB Web Services fit into the current SRB architecture.
Figure 1. SRB Web Services Implementation in Conjunction with Current SRB
2. SRB MATRIX

2.1 Overview

SRB (Storage Resource Broker) is a middleware for data grid. SRB Matrix is a web-based application implementing the functionalities provided by SRB as Web services for future grid operations. Some of the potential projects that benefit using SRB Matrix include:

- TeraGrid: TeraGrid is a National Science Foundation (NSF) funded multi-year effort to build and deploy the world’s largest, fastest, most comprehensive, distributed infrastructure for open scientific research. This extended TeraGrid environment will provide the national research community with more than 20 teraflops of computing power distributed among five different locations in the country and nearly one petabyte (one million gigabytes) of storage capacity.

- Griphyn: Grid Physics Network (GriPhyn) is a team of experimental physicists and information technology researchers who plan to implement the first petabyte-scale computational environment for data intensive science of the 21st century.

- SCEC-CME: The Southern California Earthquake Center (SCEC), headquartered at the University of Southern California, is a regionally focused organization founded in 1991 with a mission to gather new information about earthquakes in Southern California, integrate that knowledge into a comprehensive and predictive understanding of earthquake phenomena, and communicate this understanding to end-users and the general public in order to increase earthquake awareness, reduce economic losses, and save lives [SCEC 2003].
The Community Modeling Environment (CME) is one of SCEC's major research projects focusing on information technology research collaboration. The goal of the CME project is to develop an integrated environment in which a broad user community encompassing geo-scientists, civil and structural engineers, educators, city planners, and disaster response teams can have access to powerful physics-based simulation techniques for seismic hazard analysis. One component of CME is "digital library" technology, which will allow scientists to organize and retrieve information stored throughout the country. This requires new tools like the SRB Matrix to access existing data collections and simulation programs, as well as the ability to incorporate new collections of data generated by the simulations.

SRB Matrix is used at the San Diego Super Computing Center, San Diego, California. SRB Matrix includes:

- Synchronous Web services

- An event-trigger system to support asynchronous Web service invocations

- SRB-RPC is the data transport protocol.

- Implementation of each data grid operation as a separate service.

2.2 Requirements of SRB Matrix

2.2.1 SRB Web Service Server

SRB Web Service Server will service Simple Object Access Protocol [SOAP 2003] requests through HTTP. Future editions will be designed to support other protocols
like SMTP. The SRB Web Service Server will deliver SRB Web services ("SRB services") using the core SRB technologies developed at the SDSC lab.

2.2.2 Data Management

One of the major requirements of data management is to provide support of ACID (Atomicity, Consistency, Isolation, Durability) properties. In this current project, each SRB service call is the corollary of a transaction in the database. The ACID properties ensure the robust nature of SRB. Some base level functionality is added to SRB to support these properties.

- Atomicity: Either the Web service must complete all the actions or must not make any change in the system. Atomicity is met using a two-phase handshake with the client with a TTL (time to live or number of attempts to retry to commit a service).

- Consistency: Changes in a logical or physical SRB space is consistent across the various users using different clients.

- Isolation: Concurrent SRB services on a shared resource must take place without any conflict.

- Durability: Maintaining the updated states after the completion of a SRB service.

[DAKS 2002]

2.2.3 Security and Support for Computer Applications

Using HTTP protocol for transfers over the wire can provide security. The SRB is quite secure. SRB provides a reasonable level of security while still providing convenience features and high performance. The Encrypt1 challenge/response is secure against network eavesdropping, while the use of user passwords is convenient and straightforward for both users and administrators. Generally, the SRB is as secure as the
1. MatrixTicket
2. ParentCollection
3. newCollection

2.8 Web Service: List a Collection

This service lists a collection. The user can specify a collection she would like to view. The parameters required for these services are:

1. Matrix Ticket
2. Collection Name

2.9 Web Service: Delete a Collection

This service deletes a collection. The user can specify a collection she would like to delete. The parameters required for this service are:

1. Matrix Ticket
2. Collection Name

2.10 Web Service: File Upload to SRB

This service uploads a file from the local directory to a collection. Every data object under SRB/MCAT is associated with a collection. The user can upload a file from a local directory to a collection specified. The parameters required for this service are:

1. MatrixTicket – The SRB account holder gets a SRB Ticket.
2. ParentCollection – The collection specified by the user where the file is to be uploaded.
3. Filename – the file name to be uploaded.

2.11 Web Service: File Download from SRB

This service downloads a file to a local directory from SRB.

Every data object under SRB/MCAT is associated with a collection.

The user can download a file from a SRB collection to a local directory specified. The parameters required for this service are:

1. MatrixTicket
2. ParentCollection
3. Filename

2.12 Web Service: Delete File

This service deletes a file in a collection in SRB. Every data object under SRB/MCAT is associated with a collection. The user can delete a file from a SRB collection specified. The parameters required for this service are:

1. MatrixTicket
2. CollectionName
3. Filename

2.13 Web Service: Create Container
DBMS used to store the MCAT and the physical resources used to store the data. User identity is as secure as the client host system. Since the SRB server runs as a non-root user, it does not present a vulnerability to the OS if compromised. This is a big advantage over software systems that need to be run as a root user.

Support is provided to help get sites up and running with the SRB. There is an SRB chat email list for SRB administrators, developers and users to discuss questions, problems, and solutions (it includes an archive of previous posts). The web site includes information on current bugs, future plans, current projects, etc.

2.2.4 Scalability

The system is scalable in the following aspects:

- Amount of data. Assuming the presence of required hardware, the system is ready to handle any amount of data without any significant degradation of performance.
- Number of users. The software designed is not a bottleneck to the maximum number of users. The number of users might be limited by the capability of the hardware.

2.2.5 Change is Inevitable

The initial design is highly flexible to adapt to any change in the technology. A loose coupling approach will help later when some part needs to be altered without affecting the entire architecture. The interfaces between various layers are kept simple. SDSC Matrix is a data grid workflow management system. Matrix can be used to create, access and manage workflow process pipelines. Matrix internally uses the Data Grid Language, which can be used to describe, query and control process-flow pipelines. This design helps to adapt to any changes in the future. Matrix API can be used to define
multiple SRB commands (and non-SRB grid services) as a single dataflow process and execute it on multiple servers. Matrix is available as a (SOAP/WSDL) web service. Matrix client programming for SRB is made very simple using a developer friendly Java API (smaller learning curve).

2.2.6 Ability to Use the Exiting Environment

It is possible to use the existing SRB infrastructure with minimal changes. Java Native Interface (JNI) can be used if needed to use any of the existing clients of SRB.

2.3 SRB Matrix Development

The Matrix uses the standards based on the SOAP protocol. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses [Java 2003]. Technically, clients could be written by any one using any of the SOAP tools and it’s not necessary to have a separate client package. SRB Matrix will have client packages that will help SDSC and SRB users to use it.

2.3.1 Technologies Used for the Development

Current clients are using WSIF (Web service Invocation Framework), wsdl4j specifications [IBM-WSD4J 2003a]. WSIF is a project initiated by Apache and is currently maintained and developed under Apache [APACHE 2003].
Web Service Invocation Framework (WSIF) version 2.0: The WSIF is a simple Java API for invoking Web services, no matter how or where the services are provided. It frees the developer from the constraints of having to develop services for particular transport protocols or service environments. Thus, it has an API that provides binding-independent access to any Web service. It allows stubless or completely dynamic invocation of Web service, based upon examination of the metadata about the service at runtime. It also allows updated implementations of a binding to be plugged into WSIF at runtime. It can also allow a new binding to be plugged in at runtime. It allows the calling service to choose a binding deferred until runtime. Finally, it is closely based upon WSDL (Web services Description Language), so it can invoke any service that can be described in WSDL [IBM - WSIF 2003a].

Web Services Description Language (WSDL): The Web services Description Language for Java Toolkit (WSDL4J) allows the creation, representation, and manipulation of WSDL documents describing services. This code base will eventually serve as a reference implementation of the standard created by JSR110: Java APIs for WSDL [IBM - WSDL 2003].

Java version 1.4: Java technology is both a programming language and a platform [JCP 2003].

2.4 SRB Matrix Server

The following are the basic components of the SRB Matrix Server architecture shown in Figure 2:
Figure 2. Architecture and Implementation of SRB Matrix Server
• Srb4Ws: Srb4Ws is a SRB SOAP Server that can handle SRB requests made using the SOAP. This could be run using the servlet container on any J2EE (Java 2 Platform, Enterprise Edition) compliant application server supporting SAAJ (Soap With Attachments API for Java) and JAXM (Java API for XML Messaging). The development is done with Apache Axis version-1.0, which is free software with Apache open source license matching the requirements.

• Srb4Soa: Srb4Soa is a wrapper over the JNI layer. There is an object oriented layer of SRB using Java that could be used by Srb4Ws and other similar projects in the future.

• SrbJNI: SRB JNI layer is available as a SRB Java Glue package.

• Transaction database: Stores service status and other events of interest to the SRB4WS.

• Query Interface: Interface to query the status of asynchronous services offered by SRB.

• Service Publisher: SRB Web service server could publish itself to a UDDI (Universal Description Discovery and Integration) like registry and host the WSDL (Web Services Description Language) documents that have the binding information for various services provided.

2.5 SRB Matrix Client

Matrix Client is essentially a "wrapper" or layer on top of the SRB. It provides coordinated execution of process-flow pipelines in Grid environments based on the use of a Data Grid Language (DGL), which functions for data grids like the Structured Query Language (SQL) does for databases. It can create and manage process flow pipelines,
providing dynamic control of SRB and other services and facilitating scientific computing processes.

Scientific computing requires integrating data from various locations with computing and other resources from different locations into complex workflows. Such distributed Grid workflows are subject to uncertainties, dynamic changes in constraints, and failures, and the Data Grid Language allows Matrix to describe these processes as pipelines, giving greater capability to respond to dynamic Grid environments. Matrix can coordinate data flows from sensors to analysis pipelines, digital libraries, and persistent archives, and will be particularly valuable in large distributed data-intensive environments.

SRB Matrix Client is written in Java for portability and supported on all platforms with Java runtime, including AIX and other UNIX and Linux platforms, Windows, and others.

The following are the characteristics of the SRB Matrix Client:

- **System Requirements:**

  The system requires Java 1.4 (or higher – could be checked by running “java –version”) environment. The system has to be connected to the Internet.

- **Matrix download**

  Extract the compressed file matrix.zip to a directory for eg. C:\SRB. The unzip operation will automatically create a directory with name “matrix” and unzip the matrix files. Hence all the matrix files will be stored in the directory C:\SRB\matrix

- **Set Up**
Make an environment variable named MATRIX_HOME that points to the directory in which matrix files are present. In this case it would be C:\SRB\matrix. This can be accomplished using the dos command set as shown in the Figure 3.

![Figure 3. Creating an Environment Variable](image)

Open the matrix config directory and run the command “matrixClasspath” as shown in the screen shot in Figure 4. This will set up the class path required by matrix as shown in Figure 4.
2.6 User Interface

The SRB Matrix user interface allows the user to login to the system. The user is assigned a ticket and is authenticated. Once authenticated, the user can access every operation separately. Each operation is a separate service. There are also some Matrix Client examples the user can run to get more familiar with the process.

2.6.1. SRB User Information

When the user accesses the system he/she is first asked to enter user information. This information is used to create and authenticate a ticket to allow the user further access to the system. Any user who is granted the ticket can access any service in the system.

The following information is required from the user:

1. *WSDL location* – wsdll location is the URL to be accessed on the web.
2. **Username** – username is the SRB username to be used by the account/ticket holder. This username is sent to the server for further transaction.

3. **Domain** – A domain is a string used to identify a site or project.

4. **SRB hostname** – The hostname is the site where SRB is hosted. This host is used to connect to the server.

5. **Port** – Port is the port on the SRB server side that the SRBMaster is listening on. This port is used to connect to the server.

6. **Password** – Password is chosen by the user to access the system.

Once the user submits the information, the system echoes it back for verification and prompts the user to verify the user information provided by him/her. Once the user verifies his/her input, it is stored for the remaining transaction.

For demonstration purposes, the user can opt to use the default values of the program. The default values are as follows:

1. **WsdIUrl** = http://multivac.sdsc.edu: 9054/axis/services/urn: Datagrid?wsdl
2. **Username** = wsdl-demo
3. **Domain** = sdsc
4. **Port** = 5824
5. **Password** = ENCRYPTED

### 2.6.2 Function: Create Connection

The first step invokes the service to establish the requested connection. The service prompts the user to enter the wsdl location to which he/she wishes to connect. Once the user gives the required information, the system checks for the
validity/availability of the connection. If it is an invalid location or if the location is unavailable the program sends an error message to the user otherwise it establishes a connection to the specified location.

2.6.3 Function: Create a Matrix Ticket

This function invokes a service to create a SRB Matrix Ticket. Once the connection is established the system retrieves the information collected from the user and prepares a Matrix Ticket. The ticket consists of following information:

- **username** - SRB account holder’s username
- **domain** - SRB account holder’s requested domain name. Domain is the string used by the user to identify a site or project in which he/she is interested.
- **srbHost** - SRB account holder’s host name is the site where SRB is hosted.
- **srbPort** - SRB account holder specifies the port number to which the server connects.
- **passPhrase** - SRB account holder’s password
- **default Collection** - Default Collection name is the default collection the user specifies.
- **default Resource** - Default Resource name is the default collection the user specifies.
- **month** - month when the ticket was created
- **day** - day when the ticket was created
- **year** - year when the ticket was created
- **hour** - hour when the ticket was created
- **minute** - minute when the ticket was created
2.6.4. Function: Authenticate Matrix Ticket

This function invokes the service to authenticate the Matrix Ticket created. The SRB account holder gets the Matrix Ticket. The account holder or any ticket holder can access any of the services provided. The Matrix ticket is a means of limiting access to the system. Only those authorized users are provided with all the necessary information from their respective organizations to access the SRB Matrix. Once the user has a ticket she can access any or all Web services. Each SRB operation is a separate Web service.

2.7 Web Service: Create a Collection

This service creates a collection. A collection is a logical name given to a set of data objects. All data objects stored in SRB/MCAT are stored in a collection. A collection can have sub-collections, and hence provides a hierarchical structure. As a simple analogy, a collection in SRB can be equated to a directory in a Unix file system. But unlike a file system, a collection is not limited to a single device (or partition). A collection is logical but the data objects grouped under a collection can be stored in heterogeneous storage devices. There is one obvious restriction; the name given to a data object in a collection or sub-collection should be unique in that collection. The user may wish to give the home directory (/home/user.domain) as the parent collection in which a new sub-collection is to be created. A domain is a string used to identify a site or project. SRB administrator has the authority to create domains.

The following parameters are required for this service:
1. MatrixTicket
2. ParentCollection
3. newCollection

2.8 Web Service: List a Collection

This service lists a collection. The user can specify a collection she would like to view. The parameters required for these services are:

1. Matrix Ticket
2. Collection Name

2.9 Web Service: Delete a Collection

This service deletes a collection. The user can specify a collection she would like to delete. The parameters required for this service are:

1. Matrix Ticket
2. Collection Name

2.10 Web Service: File Upload to SRB

This service uploads a file from the local directory to a collection. Every data object under SRB/MCAT is associated with a collection. The user can upload a file from a local directory to a collection specified. The parameters required for this service are:

1. MatrixTicket – The SRB account holder gets a SRB Ticket.
2. ParentCollection – The collection specified by the user where the file is to be uploaded.
3. *Filename* – the file name to be uploaded.

### 2.11 Web Service: File Download from SRB

This service downloads a file to a local directory from SRB.

Every data object under SRB/MCAT is associated with a collection.

The user can download a file from a SRB collection to a local directory specified. The parameters required for this service are:

1. *MatrixTicket*
2. *ParentCollection*
3. *Filename*

### 2.12 Web Service: Delete File

This service deletes a file in a collection in SRB. Every data object under SRB/MCAT is associated with a collection. The user can delete a file from a SRB collection specified. The parameters required for this service are:

1. *MatrixTicket*
2. *CollectionName*
3. *Filename*

### 2.13 Web Service: Create Container
This is a service to make a container in SRB. A Container is a structure containing a number of small files; a container is created to improve performance. This works very well with resources that include tapes. The whole container is retrieved from tape, cached on SRB disk, and then multiple files can be quickly read from and written to the container copy on disk. The SRB handles the bookkeeping for the container.

The SRB container stores multiple files as one single file. It grows the container continuously by adding new files as they are ingested into the container. Hence, the container can be grown as needed. Also, unlike a tar ball, users can read individual files without downloading the container to their desktops.

The SRB maintains an index to the container layout in its Metadata Catalog (MCAT) and uses it when retrieving individual files. A user can modify or delete files in a container as though they are doing these operations on a normal file and the SRB takes care of the operation.

The container is not made on the desktop and then loaded into the SRB. Instead it is created at that time on the resource. Containers are normally assigned a logical resource, which has two physical components: an archive resource such as the HPSS or roadnet-sam, and a cache resource such as a Unix file system (e.g. roadnet-unix). All the construction, file access and modifications are done on the cache resource and the storage of a full container or a non-needed container is accomplished on the archive resource. Hence, the archive sees a single file and the construction is completed before getting into the archive on the cache resource (not on the users desktop), which is also a resource, controlled by the SRB.
Containers grow in size and are pinched off into physical pieces by the SRB so that a single container might look very long, but actually consists of multiple files of smaller sizes. Normally the pinching off is to be around 100 Mbytes but then it can be in the GB range also. This is akin to block in a tape system.

Hence, the user sees one container where the user "puts" the data, but like a goods-train, the container is physically divided. Individual files are much smaller than the container size. The parameters required for this service are:

1. Matrixticket
2. parentContainer - the default container in SRB
3. newContainer - the container to be created in SRB
4. containerSize - the size of the container
5. resourceName - resource name

In the terminology of SRB, a resource is a software/hardware system that provides storage functionalities. The term is equivalent to "physical resource". For example, HPSS can be a resource, as can a Unix file system.

2.14 Web Service: List Container

This is a service to list a container in SRB. The parameters required for this service are:

1. Matrixticket
2. parentContainer - the default container in SRB
3. newContainer - the container to be created in SRB
2.15 Web Service: Delete Container

This is a service to delete a container in SRB. The parameters required for this service are:

1. Matrixticket
2. Container Name

2.16. Project Environment

As shown in Figure 5, the SRB Matrix is a web-based application and provides Web services for SRB. As mentioned earlier, SRB is the tool for Storage and Data Management and SRB Matrix is the data management capability of SRB such as Web services for Service Oriented Architecture that can be used by other developers. SRB is used to support data sharing, data publication and data preservation through the use of the digital library, data grid and persistent archive software components.

The interface between the Grid Port and SRB technology was managed through CGI scripts. This interface is upgraded to Java / WSDL technologies to better interact with grid services. Each of the required SRB services needed for managing a collection is wrapped with a WSDL interface. An initial version has been written for a representative data management service. The Web service Description Language implementation of a grid service can be modified in the future to the emerging Open Grid Services Architecture (OGSA).

SRB Web service Server services SOAP (Simple Object Access Protocol) requests through HTTP. The SRB Web service Server delivers SRB Web services ("SRB
services") using the core SRB technologies developed. Future editions will support other protocols like SMTP.

Figure 5. SRB Matrix Project Environment
3. SYSTEM DESIGN OR RESEARCH

3.1 System Analysis

The Design and Analysis Phase of the project to develop the SRB Matrix started with the following steps:

1. Since SRB Matrix was a module of a project to develop the SCEC (Southern California Earthquake Consortium) Community Model, I attended meetings held by all the participating institutes such as SCEC (Southern California Earthquake Center), SIO (Scripps Institute of Oceanography), and SDSU (San Diego State University) to better ascertain the complete project and the role of my module within that project.

2. I discussed with Dr. Reagan Moore (Project Chair) and Dr. Stuart Johnson about what type of features were needed in the SRB Matrix. The features discussed included user interface, security, database design, and data integrity.

3. Designed a framework of what the user interface would look like, proposed and analyzed the design with Drs. Moore and Johnson.

4. Researched the Data grid language to choose for the development of SRB Matrix.

Matrix is provided as a SOAP based Web service; the main issues that came up during the design process were to provide user interface, security and data integrity.

- User Interface: It was important to provide a user-friendly Web environment. Most users of this system will be new to the idea and will have a very limited understanding of the whole process of Web Services. It is important to design a system in which the user will have no problems in using the tools provided to
navigate through the system. Providing proper documentation and help features also forms an integrated user interface.

- **Security**: Security is important in an environment where access criteria depend on the type of user. The system uses ticket set during the user login to provide appropriate access to the system. Security features like user identification and authentication, incorporating secure transactions and disabling options when the user is not qualified to use a certain feature are some of the security features that will be used in developing the system.

- **Data Integrity**: There are issues of data integrity related to the design of any system. The various data integrity features that were used while developing the SRB Matrix were: handling improper data, updating data and mapping standard error messages with user friendly messages.

### 3.2 System Architecture and Overview

The SRB Matrix architecture requires three fundamental operations: publish, find and bind. Figure 6 shows a graphical representation of the architecture.

![Figure 6. Publish, Find and Bind](image-url)
Service providers publish services to a service broker. Service requesters find required services using a service broker and bind to the services.

The SRB Matrix architecture is ordered into two tiers: the user interface in the client browser and the Matrix server. Figure 7 shows a graphical representation of the architecture.

![System Architecture Diagram](image)

**Figure 7. System Architecture**

The functional description of the architecture is listed below:

- **Matrix Client**: This is comprised of the user interface as viewed on the client's Web browser. The user interface is implemented using JavaScript support. JavaScript will be used to validate user input and will be executed on the client's Web browser.

- **Matrix Server**: Srb4Ws is the SRB SOAP Server that can handle SRB requests made using the SOAP. This could be run using the servlet container on any J2EE (Java 2 Platform Enterprise Edition) compliant application server supporting SAAJ (Soap with Attachments API with Java) and JAXM (Java API for XML Messaging). It was developed using Apache Axis version-1.0, which is free software with Apache open
source license matching the requirements. Srb4Soa is a wrapper over the JNI (Java Native Interface) layer. It is an object-oriented layer of SRB using Java that could be used by Srb4Ws and other similar projects in the future.

An overview of the system installation is: The user will install the SRB Matrix application and set up the class path as described in the manual with the application. Then the user connects to the system and is authenticated. Finally, the Matrix will send an XML document to invoke data grid service and will receive a Matrix response. SRB Web service Server will service SOAP (Simple Object Access Protocol) requests through HTTP. The SRB Web service Server will deliver SRB Web services using the core SRB technologies developed.

3.3 Matrix API Design

3.3.1 API Document Organization

The API (Application Programming Interface) is an object-oriented design created for Java programmers to use SRB Web services with little or no prior knowledge of WSDL or other Web service technologies. Figure 8 below depicts a program that is running on the Java platform. The Java API and the virtual machine insulate the program from the hardware.

![Matrix API](image)

Figure 8. Matrix API
The Matrix API document has pages corresponding to the items in the navigation bar, described as follows:

**Overview**

The Overview page is the front page of this API document and provides a list of all packages with a summary for each. This page can also contain an overall description of the set of packages as shown in Figure 9 below.

<table>
<thead>
<tr>
<th>Packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>edu.sdsc.matrix.srb.client</td>
<td>Contains list of classes and interfaces available to the SRB Matrix Client</td>
</tr>
<tr>
<td>edu.sdsc.matrix.srb.util</td>
<td>Contains list of classes and interfaces available for internal use of SRB Matrix</td>
</tr>
</tbody>
</table>

**Figure 9. Overview Page of the API Document**

**Package**

Each package has a page that contains a list of its classes and interfaces, with a summary for each. This page can contain four categories:

- Interfaces (italic)
- Classes
- Exceptions
- Errors

**Class/Interface**

Each class, interface, nested class and nested interface has its own separate page. Each of these pages has three sections consisting of a class/interface description, summary tables, and detailed member descriptions:

- Class Inheritance Diagram
- Direct Subclasses
- All Known Sub Interfaces
- All Known Implementing Classes
- Class/Interface Declaration
- Class/Interface Description
- Nested Class Summary
- Field Summary
- Constructor Summary
- Method Summary
- Field Detail
- Constructor Detail
- Method Detail

Each summary entry contains the first sentence from the detailed
description for that item. The summary entries are alphabetical; while the
detailed descriptions are in the order they appear in the source code. This
preserves the logical groupings established by the programmer.

Use

Each documented package, class and interface has its own Use page. This
page describes which packages, classes, methods, constructors and fields
use any part of the given class or package. Given a class or interface A, its
Use page includes subclasses of A, fields declared as A, methods that
return A, and methods and constructors with parameters of type A. The
user/developer can access this page by first going to the package, class or
interface, then clicking on the "Use" link in the navigation bar.
Tree (Class Hierarchy)

There is a Class Hierarchy page for all packages, plus a hierarchy for each package. Each hierarchy page contains a list of classes and a list of interfaces. The classes are organized by inheritance structure starting with java.lang.Object. The interfaces do not inherit from java.lang.Object.

- When viewing the Overview page, clicking on "Tree" displays the hierarchy for all packages as shown below.
  - class java.lang.Object

```java
class edu.sdsc.matrix.srb.client.MatrixCollection
edu.sdsc.matrix.srb.client.MatrixConnection
edu.sdsc.matrix.srb.client.MatrixContainer
edu.sdsc.matrix.srb.client.MatrixFile
edu.sdsc.matrix.srb.client.MatrixQuery
edu.sdsc.matrix.srb.client.MatrixResource
edu.sdsc.matrix.srb.client.MatrixServerURL
edu.sdsc.matrix.srb.client.MatrixService
edu.sdsc.matrix.srb.client.MatrixServiceResult
edu.sdsc.matrix.srb.client.MatrixTicket
edu.sdsc.matrix.srb.client.MatrixUser
```

- When viewing a particular package, class or interface page, clicking "Tree" displays the hierarchy for only that package.
Deprecated API

The Deprecated API page lists the entire API that has been deprecated. A deprecated API is not recommended for use, generally due to improvements, and a replacement API is usually given. Deprecated APIs may be removed in future implementations.

Index

The Index contains an alphabetic list of all classes, interfaces, constructors, methods, and fields.

Prev/Next

These links take the user to the next or previous class, interface, package, or related page.

Frames/No Frames

These links show and hide the HTML frames. All pages are available with or without frames.

Serialized Form

Each serializable or externalizable class has a description of its serialization fields and methods. This information is of interest to re-implementers, not to developers using the API. While there is no link in the navigation bar, the user can get to this information by going to any serialized class and clicking "Serialized Form" in the "See also" section of the class description.

3.3.2 API Overview
The API consists of two main packages: edu.sdsc.matrix.client and edu.sdsc.matrix.util. The client package is for the SRB matrix users and the utils package is for internal use of SRB Matrix. The API contains all the classes in each package.
<table>
<thead>
<tr>
<th>Package Hierarchies:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>edu.sdsc.matrix.client, edu.sdsc.matrix.util</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class Hierarchies:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>• class java.lang.Object</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixCollection</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixConnection</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixContainer</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixFile</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixQuery</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixResource</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixServerURL</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixService</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixServiceResult</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixTicket</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixUser</td>
<td></td>
</tr>
<tr>
<td>o class java.lang.Throwable (implements java.io.Serializable)</td>
<td></td>
</tr>
<tr>
<td>o class java.lang.Exception</td>
<td></td>
</tr>
<tr>
<td>o class java.io.IOException</td>
<td></td>
</tr>
<tr>
<td>o class java.rmi.RemoteException</td>
<td></td>
</tr>
<tr>
<td>o class org.apache.wsif.WSIFException</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.ServiceClientException</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.MatrixException</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.util.Utils</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.WSDLOperation</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.WSDLOperationWSIF</td>
<td></td>
</tr>
<tr>
<td>o class edu.sdsc.matrix.client.WSDLProviderWSIF</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface Hierarchy:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>interface edu.sdsc.matrix.client.WSDLProvider</td>
<td></td>
</tr>
</tbody>
</table>

Table I. API Overview
3.3.2 API Summary

The API is summarized in Tables II and III. There are two packages: Package edu.sdsc.matrix.client and Package edu.sdsc.matrix.utils. The tables show the name as well as the description of each class.
## Package `edu.sdsc.matrix.client`

<table>
<thead>
<tr>
<th><strong>Interface Summary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WSDL Provider</strong></td>
</tr>
<tr>
<td><strong>Class Summary</strong></td>
</tr>
<tr>
<td><strong>MatrixCollection</strong></td>
</tr>
<tr>
<td><strong>MatrixConnection</strong></td>
</tr>
<tr>
<td><strong>MatrixContainer</strong></td>
</tr>
<tr>
<td><strong>MatrixFile</strong></td>
</tr>
<tr>
<td><strong>MatrixQuery</strong></td>
</tr>
<tr>
<td><strong>MatrixResource</strong></td>
</tr>
<tr>
<td><strong>MatrixServerUrl</strong></td>
</tr>
<tr>
<td><strong>MatrixService</strong></td>
</tr>
<tr>
<td><strong>MatrixServiceResult</strong></td>
</tr>
<tr>
<td><strong>MatrixTicket</strong></td>
</tr>
<tr>
<td><strong>MatrixUser</strong></td>
</tr>
<tr>
<td><strong>WSDL Operation</strong></td>
</tr>
<tr>
<td><strong>WSDL Operation WSIF</strong></td>
</tr>
<tr>
<td><strong>WSDL Provider WSIF</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Exception Summary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MatrixException</strong></td>
</tr>
<tr>
<td><strong>ServiceClientException</strong></td>
</tr>
</tbody>
</table>

Table II. **Package edu.sdsc.matrix.client**

- **Package edu.sdsc.matrix.utils**

<table>
<thead>
<tr>
<th><strong>Class Summary</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Utils</strong></td>
</tr>
</tbody>
</table>

Table III. **Package edu.sdsc.matrix.utils**
4. EVALUATION AND RESULTS

The SRB Matrix is to be used at the San Diego Supercomputing Center by various projects like SCEC-CME, GRIPHYN, etc. The SRB Matrix will provide Web services for SRB. Also, the API will be provided for a Java developer to be able to invoke the services without having a strong knowledge of Web service technologies like WSIF, WSDL, etc.

This application can be further extended to provide a .NET interface, Grid Security Infrastructure (GSI) [IBM-Web services 2003] authentication and also can be ported to Open Grid Services Architecture (OGSA) [GLOBUS 2003] to provide grid capabilities. The complete implementation of the system will include testing the system with active data and developing a user manual.

4.1 Testing

- Tomcat and temporary file cache

In Microsoft Windows if the Apache Tomcat was started from the shortcut (instead of the regular command line prompt), there was a “file not found” problem during ingest or download.

This happened because of the temporary file written to disk could not be read back again. This bug was fixed by ensuring that all temporary files were written to a pre-defined file cache specified in the properties file.

- Accessing multiple Attachments in a single request

When a flow was executed in parallel, and multiple steps tried to retrieve the attachments, they were not thread-safe. This defect was fixed by storing all the
attachments in a thread-safe hash table, where they can still run concurrently or in parallel.

- Selection of multiple user-defined meta data fields

  There is a problem in this release when we use XQuery to select more than one user-defined Meta data field. This is a bug from the underlying Jargon code, which was removed.

- Local Identifier

  The Data Grid Language Schema for ingestion of a dataset (file) has 'logical identifier' as an optional element. But, the current Matrix implementation requires it to be present.
5. FUTURE WORK

The focus of future SRB Matrix will be defining the underlying peer-to-peer protocols by which GfMS's (Gridflow Management Systems) can coordinate with one another over the WAN. There is a need to define a set of powerful and minimally verbose protocols by which GfMS's can carry out such collaboration. Aggravating this challenge is the difficulty of distributed coordination over wide area networks (e.g. atomic commit, rollback, etc.) Once these underlying protocols are built, any type of scheduling strategy could be implemented on top of these grid flow protocols. The current work on extensions to Matrix will enable deploying it more readily in production settings. Addressing a need in Matrix for a persistent storage mechanism as well as more comprehensive interfaces to existing remote execution systems.

Also, the future focus will be to make SRB Matrix an ultimate in “data grid services” and also help other institutions to develop similar infrastructure. The newer SRB Matrix versions will also include:

- Extended Metadata Query Functionality
- Upgrade Matrix code to work with latest Apache jar files
- Persistence layer using RDBMS
- Matrix agents to invoke any WSDL web service (during data-flow process)
- Matrix agents to invoke any GGF OGSA service (during data-flow process)
- OGSA Grid File System, OGSA Data Services interfaces
- Iterative and conditional invocation of services
6. CONCLUSION

SRB Matrix has been developed successfully and is implemented with the required resources to install the system. While there was an existing infrastructure being used in production to support multiple scientific disciplines, the challenge was to understand the unique requirements of the SCEC program, develop the requisite features, and implement SCEC specific collections.

However, the current system has room for improvement; for example, it is not using the full capability of SRB. It is also not very flexible when it comes to handling attachments and files. Improvement to the system can be made by performing additional research and development to deliver grid workflow protocols and the workflow language descriptions necessary to build a peer-to-peer infrastructure for the Gridflow Management Systems (GfMS).

Since SRB Matrix is a workflow management system, it can further polish its ability to query and control the execution of grid flow pipelines using Data Grid Language.
BIBLIOGRAPHY AND REFERENCES


APPENDICES

Appendix A  CD-ROM Containing:

- Microsoft Word file for this report. (Report.doc)
- Source Code for SrbMatrix Interface
- readme.txt (installation file)
- 10 Page summary of this work as a draft paper for submission for publication.