2-3 February 2006

SRB Workshop

SAN DIEGO SUPERCOMPUTER CENTER

SDSC

SDSC Technical Report
SDSC TR-2006-1

Editor
Reagan W. Moore

Program Chair
Wayne Schroeder

SRB Technical Committee
Arcot Rajasekar
Michael Wan
Wayne Schroeder

Storage Resource Broker - Workshop on SRB Applications

Managing NOAO Distributed Archive using SRB[*]

Irene Barg
National Optical Astronomy Observatory
ibarg@noao.edu

Abstract

The NOAO Data Products Program (DPP) data flow system [1] combines new data storage, data reduction pipelines, VO portals, and a transport system to link these together. This integrated system is NOAO's first step towards establishing a data center that is relevant in the Virtual Observatory (VO) era. The core piece of this integrated system is the data management and transport system (DTS). A prototype DTS, was commissioned in August 2004, with the San Diego Supercomputer Center (SDSC) Storage Resource Broker (SRB) as the core technology for managing NOAO's physically distributed resources. The prototype ran successfully for a year. During this time, design of the NOAO Science Archive (NSA) Data Service (DS) evolved and some of the functional requirements of the DS could be met by the DTS migrating to the federated MCAT, SRB zone model. After running several proofs of concept tests, the task to extend the DTS into SRB zone model, began in June 2005, and deployed in early November. This paper describes the DTS zoneSRB architecture, and the accompanying client software.

1. Introduction

One key requirement of the NSA is to maintain holdings in an access controlled, volume-managed, replicated repository. This will be accomplished through the use of a Data Service [2]. The Data Service provides mechanisms by which data can be transported, stored, managed, replicated, provided, and retrieved. The Data Service will provide a common interface to the NOAO data holdings across multiple domains which include: Kitt Peak National Observatory (KPNO) and NOAO headquarters in Tucson, Arizona; Cerro Tololo Inter-American Observatory and NOAO headquarters in La Serena, Chile, and our offsite NSA partner the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC). The DTS has successfully used SRB as the core technology for file transport and replica management since August 2004. The NSA team wanted to leverage the experience gained from the DTS and extend the current use of SRB to a federated MCAT SRB zone model, which will become the underlying infrastructure of the Data Service.

2. Background

In April 2004, the Data Cache Initiative (DCI) [3] was designed to combined the use of existing software products:

- NOAO Save-the-Bits (STB) project. STB relies on the BSD UNIX line printer daemon, lpd, to provide queued network data transfers from instruments in the various NOAO telescopes, to a central mountain cache.
- NCSA BIMA Archive Real Time Transferor [4] an rsync-based queuing mechanism, used to mirror the mountain cache to downtown data centers.
- SDSC SRB [5] for transport and management of replicas from each hemisphere's data center to NCSA for off-site storage.

Why SRB? The need for a distributed storage system and data management system was identified in 2002. Then in summer 2003, two replication middleware products were evaluated:

- Lightweight Data Replicator (LDR) [6] - a collection of tools provided by the Globus project. The Globus pieces include: GridFTP, Resource Location Service (RLS) and the Metadata Catalog Service (MCS).
- SDSC Storage Resource Broker (SRB) - a client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. SRB, in conjunction with the Metadata Catalog (MCAT), provides a way to access data sets and resources based

[*] This research was sponsored by a funding agency. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the Government, or any person or agency connected with them.

on their attributes and/or logical names rather than their names or physical locations.

There were pros and cons to both packages, but SRB was chosen over LDR because:

- o SRB provides an 'out-of-the-box' data grid solution;
- o SRB was a more mature package at the time of evaluation;
- o LDR had only 2 developers.

January 2004, the NSA began discussions on prototyping a location independent data management system. In March 2004, a proposal [7] to use existing software products (STB, *rsyncer*, SRB) to build a prototype data transport system was accepted by NOAO. In August 2004, DCI was commissioned as a prototype data transport system.

February 2005, DCI began to experience growing pains. Within the first six months, a 4TB data brick was near capacity. The NOAO NEWFIRM instrument was on the horizon, and would contribute an additional 40 Gbytes average per night. At the same time, design discussions for the R3 Data Service began. The DCI design choices were re-evaluated. Alternatives to SRB were briefly evaluated in a trade study [8], and no single solution was a clear winner. Our experience with SRB had so far, been well. We began studying and testing Federated MCAT (SRB zone) models, and once again, we believe SRB was the best choice at the time due to:

- o Three zone use case tests were promising;
- o Short-comings found could be mitigated through smart coding;
- o Momentum in current DCI implementation;
- o Mature package;
- o Track record - large deployments (BIRN, NARA, NASA IPG).

In November 2005, the DCI was successfully converted to a modified replicated data zone model and renamed as the Data Transport System (DTS). Further references to DCI and DTS represent the same system, but at different phases.

3. NOAO Data Flow

Figure 1 illustrates the NOAO data flow system [3], where data and metadata flow from NOAO telescope instruments and related facilities such as pipelines.

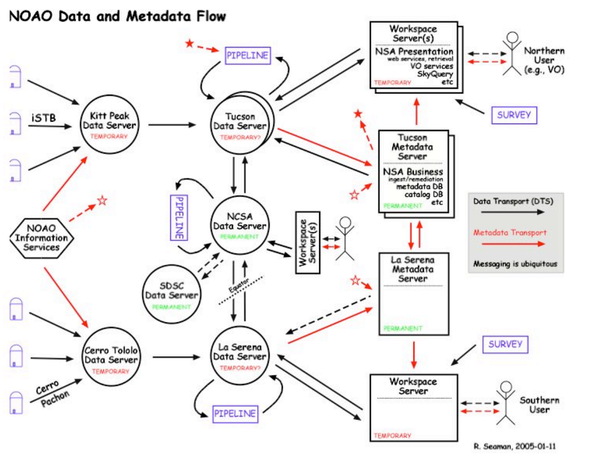


Figure 1. NOAO Data Flow

DCI was originally designed to provide a temporary safe-store (cache) for data collected at NOAO telescope facilities, and to guarantee three copies (Tucson, La Serena, and NCSA). It met this minimum requirement by using one central SRB MCAT server in Tucson that used storage resources in La Serena and NCSA to store replicas. This had two major disadvantages:

- o Single point-of-failure
- o Single site dependency

As the design of the R3 Data Service evolved, it became clear, that with a few changes, the DCI could lay the infrastructure to support much of the Data Service, but it needed to be extended to provide:

- ▪ Location independency
  - o Mountain caches, downtown data centers, must function as independent components.
- ▪ Location transparency
  - o Request for a file can be obtained from any location, even if the file no longer resides at that location.

The ability to make each site independent and robust was crucial. Each zone needs to continue to function even if another site goes down, and pick up where it left off, before communication was lost. Location independency became possible with the introduction of Federated MCAT (SRB zones) in SRB R3.0 [9]. Some highlights of SRB zones include:

- ▪ Multiple MCATs:
  - o MCAT ZONE – Defines a federation of SRB resources controlled by a single MCAT
  - o Each Zone has its own sys admin - local control of users and resources.
- ▪ Peer to peer zone architecture.

        - Each Zone can operate entirely independently from other zones.
- Data and Resource sharing across zones.
    - Use storage resources in foreign zones
    - Access data stored in foreign zones
    - Copy data across zones

3.1. NOAO-NCSA zone Architecture

Figure 2 illustrates the relationship of the DTS components under the federated MCAT architecture:

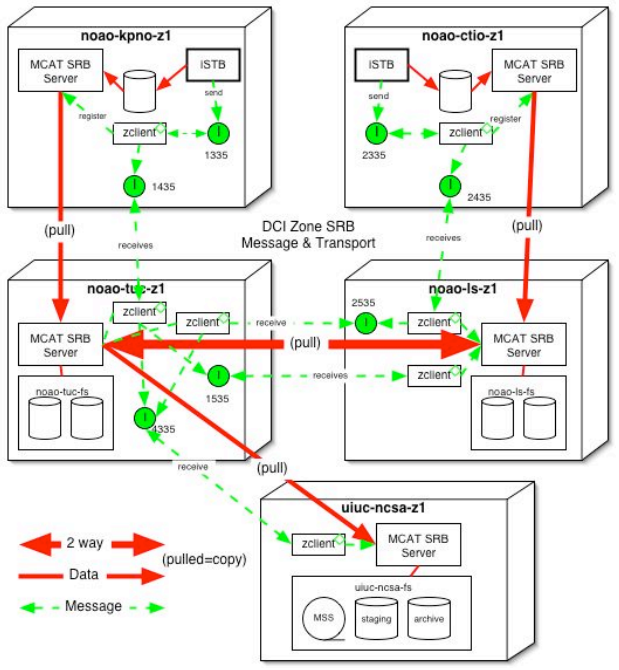


Figure 2. NOAO-NCSA SRB zones

Figure 2 illustrates the robustness of this zone configuration:

- Zclient operates in a 'pull' fashion, reducing the amount of network activity.
- Zclient communicates with local SRB server and transfers are executed between zone SRB servers for efficiency.
- Zones that are unreachable do not affect other zones. Users can still obtain data from the local MCAT, and a transfer between other zones is not affected.
- Each zone has a <u>copy</u> of each other's data.
- All transfers will be threaded for efficiency.
- The DTS is fully automated.

The next revision of DTS will have NCSA as the data hub between Northern/Southern hemispheres. This will increase file transfer efficiency and get the data into a permanent storage as soon as possible.

4. DTS Component Description

The goal was to implement SRB zones with as little code modifications as possible while retaining the basic functionality of the DCI:

- Save-the-Bits (STB) – no change
- DciArchT – message queue client – no change
- DciTrackD - message queue – no change
- Zclient – File transfer client - Perl SRB API client methods replaced *rsync*.
- SRB – complete re-design to support zones

4.1. Save-the-Bits

Data can enter the system anywhere through the message queue (DciTrackD), but the main point-of-entry is through STB. The Save-the-Bits program has been running for more than a decade at NOAO [3]. Data are captured at the instrument, simple remediation and additional meta-data are added to the FITS headers, then STB passes the 'file path, md5sum, file size' to the DciArchT (not shown in Figure 2).

4.2. Message Queue Client (DciArchT)

The DciArchT is simply an interface to the DciTrackD message queue. If DciArchT cannot connect to the daemon, it re-tries every 10 seconds until a successful connection is made.

4.3. Message Queue (DciTrackD)

The DciTrackD is a Perl implementation of a TCP server daemon. It manages the list of tracks that need to be transferred. A track is simply a string containing the file path of the file we want transferred along with its md5sum and file size. The daemon listens to a non-privileged port, and responds to one of four requests: add track; remove track; list tracks in queue; and shutdown the message queue daemon. Figure 3 illustrates the Point2Point (P2P) message system.

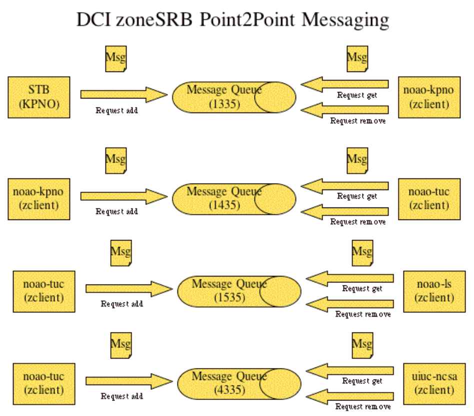


Figure 3. DTS P2P Message System

The P2P message system was chosen because:

- Wanted to keep it simple:
  - Each message has only one consumer.
  - A sender and a receiver of a message have no timing dependencies. The receiver can fetch the message whether or not it was running when the client sent the message.
  - The receiver acknowledges the successful processing of a message by making a *remove* request.
- Wanted to make it robust:
  - Upon any successful request, the message queue dumps to file.
  - Message daemon runs continuously.

4.4. File Transfer Client (Zclient)

The simple reliable *rsync* calls have been replaced with SRB API code, which has advantages and disadvantages:

- Cons:
  - More bookkeeping involved.
  - We must implement our own transaction management.
  - Database interfaces functions are often slow (srbObjStat), and sometimes confusing (srbModifyDataset).
- Pros:
  - Support for collections.
  - Data location transparency.
  - Fast file transfer using parallel I/O.

Zlcient operates in a pull (copy) fashion. It requests tracks from a designated message queue. The track message (a string containing "file path md5sum file size") constitutes a contract. The file path may be an SRB object (SRB *copy* operation) or file system path (SRB *register* operation). The receiving Zclient connects to the local SRB MCAT enable server, and request the file be copied or registered. In a *copy* operation, the actual file transfer is done between the two SRB servers. Once the transfer is complete, the Zclient compares the local copy md5sum against what was sent in the track message. If there is a miss-match, Zlcient flags an error, rolls-back the transaction, and leaves the file in the queue.

4.5. Testing

Testing was conduct at three levels prior to deploying the system:

- Installation tests – Multiple volunteers tested SRB and Zclient distribution bundles.
- Integration tests – full-up simulation of the data flow between mountain, downtown and remote site.
- Workload tests – Nagios was used to capture network, and CPU load statistics.
- File transfer tests – parallel (threaded) I/O is important and can improve transfer rates by a factor of 5 or more.

4.6. Monitoring

At the time of this writing, we have 5 zones operational, and another one pending at Cerro Pachon in Chile. Monitoring is very important. We currently monitor:

- SRB HotPage (under utilities/ping) in the SRB distribution starting with SRB3.3.
- Daily reports generated during the NOON rotation period provides a status of files queued within the last 24-hour period.
- A file status checker runs every 72 hours and compares the number of files entered into the STB system, with each DTS zone.
- Other miscellaneous management reports.

4.7. Conclusions

Please direct any questions regarding NOAO DTS project to ibarg@noao.edu.

5. Acknowledgements

The DTS project is a team effort. I would like to thank: Rob Seaman for creating STB. Nelson Zarate who developed the DCI SRB proto-type, and wrote DTS monitoring code. Nelson Saavedra for getting our Apple Xserve RAID system just in the nick of time. Chris Smith for his leadership. Stanley Yao, who conducted trade, studies on distributed storage systems, and proof-of-concept tests for SRB zones. NSCA team members Ray Plante, Ramon Williamson and David Fleming for providing the NCSA Mass Storage System interface. Michal Wronski, creator of SRB Perl API. All the people who took time to respond to my post to srbChat. Finally, to the SRB team at SDSC for creating SRB.

References

[1] NOAO DPP: http://www.noao.edu/dpp/software-changes.html

[2] A. Granados, "Overview HLA" (NSA presentation, February 2005)

[3] R. Seaman, et al, *"The NOAO Data Cache Initiative – Building a Distributed Online Data store"*, ASP Conference Series, Vol. XXX, ADASS XV

[4] Mehringer, D. 2001, "The BIMA Data Archive Real Time Transfer System", http://bimaarch.ncsa.uiuc.edu/RTT.html

[5] SRB - "The Storage Resource Broker Web Page", http://www.npaci.edu/DICE/SRB/

[6] LDR – "Lightweight Data Replicator" web page, http://www.lscgroup.phys.uwm.edu/LDR/overview.html

[7] R. Seaman, N. Zarate, I. Barg, "Save the Bits+Storage Resource Broker Interim Data Transport System", (March 2, 2004 Proposal to NOAO Data Products)

[8] S. Yao, I. Barg, "SRB Federated MCAT - NSA R3 SRB Alternatives and Recommendation", (NOAO White Paper, February 23, 2005)

[9] M. Wan, "An Overview of Federated MCAT Design (SRB 3.0)", (September2003)

Purdue Multidisciplinary Data Management Framework Using SRB*

Lan Zhao, Taezoon Park, Rajesh Kalyanam, Wonjun Lee, Sebastien Goasguen
Purdue University
Rosen Center for Advanced Computing
Email: lanzhao@purdue.edu

Abstract

This paper describes the design and implementation of the Purdue multidisciplinary data management framework. As part of the TeraGrid Resource Provider program, this framework provides a generic infrastructure for managing data collections from different data sources with multiple accessing points, serving local and national users across application domains. The framework's base component is Storage Resource Broker (SRB), a client-server middleware developed at SDSC that provides a uniform interface to heterogeneous resources. On top of SRB, domain-specific data servers such as OPeNDAP (Open-source Project for a Network Data Access Protocol) and THREDDS (Thematic Realtime Environmental Distributed Data Services) are integrated into the framework to provide additional server-side data processing. A Gridsphere based data portal has been developed which consists of customized JSR-168-compliant SRB portlets, enabling easy data discovery, access, and sharing. Our framework has been successfully applied to the management of various datasets from remote sensing images, real-time satellite and radar streaming data, to large-volume scientific datasets from climate modeling. It has made an immediate impact on the corresponding research activities enabling the development of powerful data-driven applications.

1. Introduction

Today's researchers face significant challenges in taking full advantage of large volume of data with increasing complexity generated from a wide variety of sources including remote sensors, instruments, and experiments. It also presents a great challenge for the administrators to collect, process, archive, and publish different types of data in a timely manner. As a result, data management has become a major road block in many cases where research activities require time-critical, efficient data discovery, access, and sharing. As part of the TeraGrid initiative at Purdue University, we address this challenge by developing and deploying a cross-disciplinary data management infrastructure that provides national access to local data collections. The primary goals are to make the infrastructure generic and extensible. It should be easily extendable to manage data collections from different scientific disciplines and capable of handling new data types and storage strategies as technology evolves. In addition, it is important that the system provides easy access to the data with multiple accessing points.

SRB, an open source middleware developed at SDSC [1], is the base component of our data framework. SRB provides a uniform interface to distributed and heterogeneous data resources. It also allows users to discover data based on logical attributes instead of physical file names and path names [2]. To further facilitate data discovery and processing which are often domain-specific, we have built another layer of application servers including OPeNDAP (Open-source Project for a Network Data Access Protocol) and THREDDS (Thematic Realtime Environmental Distributed Data Services) servers on top of SRB [3, 4]. This allows researchers to transform, combine, or subset datasets directly with existing OPeNDAP/THREDDS-enabled tools such as Integrated Data Viewer (IDV) and MatLab.

As the starting point, the framework has been successfully applied to the management of a number of datasets of different types in earth, atmospheric, and environmental research. Our framework has made a positive impact on the research activities that previously required the data curator to locate the DVD or CDR that contains the data being requested and to mail it to the user. A general purpose data portal has been developed, which will be used to access climate modeling data and Doppler radar data in undergraduate courses to be taught in

[*] This research was sponsored by the National Science Foundation through the Purdue TeraGrid Resource Provider award #SCI-0503992. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the Government, or any person or agency connected with them.

the Spring semester of 2006. In the second phase, we plan to extend this framework to manage spectral imaging and confocal microscopy data from bioscience and medical fields as well as to provide support for data services and workflow management.

The remainder of the paper is organized as follows: Section 2 describes different data collections currently hosted by the data management framework. Section 3 presents the design and deployment of our data management framework at Purdue University. Section 4 describes our effort in metadata design. Using NWS and climate modeling datasets as examples, Section 5 describes how data are managed and accessed through the system. Section 6 explains the multiple data access interfaces, and Section 7 describes our future work. Finally, Section 8 presents the conclusions.

2. Data Collections

As part of the TeraGrid Resource Provider, Purdue provides access to traditional High Performance Computing resources such as Linux clusters and storage area. Moreover, Purdue also strives to offer a new set of resources that is seamlessly integrated with the TeraGrid infrastructure: Data collections. It is in this context that the data framework was developed and recently deployed. Currently, multiple data collections from a number of application domains have been successfully integrated into the system.

2.1. LARS Dataset

Provided by Laboratory for Applications of Remote Sensing (LARS) at Purdue University, the LARS image data collection includes multi-spectral (3 to 15 or so wavelength bands) and hyper-spectral (dozens to hundreds of wavelength bands) image data that are being used for education and research purposes [5]. Most of the image data are for locations in the State of Indiana, dated from 1972 to 2004. The data have been collected by satellite-borne and aircraft-borne sensors. Each band of data in a data set represents the energy received by the sensor that is within the wavelength range of that band. Different sensors cover different portions of the optical spectrum. The primary data formats are ERDAS LAN, Leica Geosystems Imagine, GeoTIFF, and HDF. In addition, some are in LARSYS MIST format.

2.2. PTO Satellite Dataset

Provided by Purdue Terrestrial Observatory (PTO), the PTO image data sets currently include data from the GIVSSR sensor on the GOES-12 (also called GOES-East) satellite [6]. These data are collected every 15 to 30 minutes covering different portions or sectors of the earth disk. A full disk scan sector is obtained every 3 hours. The current data products published online include the most recent JPEG-formatted images for the eastern United States and images for the continental United States.

2.3. NWS (National Weather Service) Doppler Radar Data

Purdue University is one of the country's four top-level distributors of high-resolution radar data from the national network of Next Generation Radar (NEXRAD) [7]. The NEXRAD Radar system comprises 159 Weather Surveillance Radar-1988 Doppler (WSR-88D) sites across the United States and in selected overseas locations. Using the Unidata Local Data Manager (LDM) technology, NWS data are collected and redistributed in real time. It is composed of Level II and III data. Level II data are in three meteorological base data quantities: reflectivity, mean radial velocity, and spectrum width. These quantities are further processed to generate numerous meteorological analysis products known as Level III data.

2.4. Climate Modeling Data

This dataset is provided by Professor Matt Huber’s research group in the Department of Earth and Atmospheric Sciences at Purdue University. The data are output from the Community Climate System Model (CCSM) to simulate global climate changes [8]. It consists of four dynamic geophysical models simulating the atmosphere, ocean, land surface and sear-ice, and one central coupler component. It facilitates fundamental research on the earth’s past, present, and future climate states. This set of data is in NetCDF file format [9].

3. Data Management Framework

To accommodate the various data in different application domains described in Section 2, we have designed and developed a general-purpose data management framework at Purdue University. The main objective is to build a flexible and extensible infrastructure that can (1) be used to manage different data sources across different scientific disciplines and (2) provide multiple interfaces for users to easily discover, access, and share data in a timely manner. All resources in the framework are connected to the high-speed TeraGrid network. The architecture of the data management framework is shown in Figure 1. It is based on SRB, which provides a uniform interface to heterogeneous and distributed data resources. Our framework is composed of four modular, extensible software layers: data capture layer, SRB layer, application layer, and presentation layer.

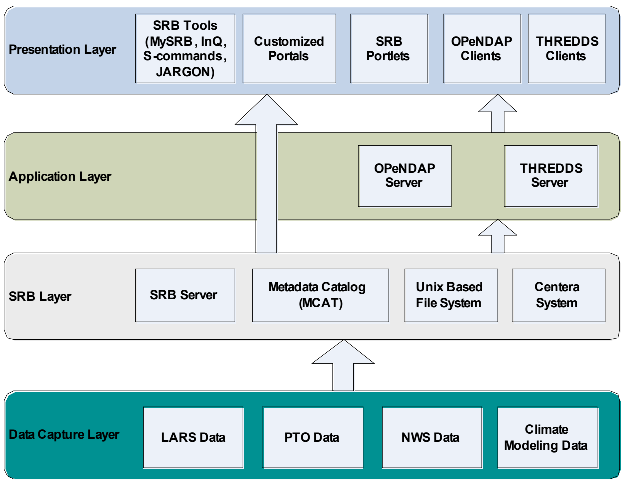


Figure 1 Architecture of Purdue Data Management Framework

3.1. Data Capture Layer

Several data drivers have been developed based on the type of data sources and temporal/spatial requirements. The data drivers connect the data sources to the SRB engine. Programs are developed to automatically extract meta-data, register meta-data in the Metadata Catalog Server at SDSC, and ingest data into the local SRB server. Tools are developed to normalize application-level meta-data to be compliant with the meta-data standards in the corresponding communities. To solve the problem of accessing the high-volume climate modeling data, a 32 TB Centera storage system donated by EMC has been deployed to provide quick online data access.

3.2. SRB Layer

Both raw data and post-processed products are stored in SRB, making them available to users. The data are stored in logical data collections, associated with domain-specific meta-data, which facilitate information discovery and exchange. For the real-time PTO satellite streaming data, SRB shadow directory objects are used to provide access to the latest data that are updated in a near-real-time manner. For Doppler radar and Climate Modeling data, SRB HTTP URL objects are used to integrate dynamically-generated THREDDS data catalog and OPeNDAP server-side processing capabilities with logical-attribute-based query functions provided by SRB metadata catalog.

3.3. Application Layer

Some of the datasets are further processed on the server side by sub-setting, aggregation, or other analysis methods before they are sent to the user. This is achieved by integrating OPeNDAP server with SRB for the climate data, and by integrating THREDDS server with SRB for the Doppler radar data. As a result, users can conveniently access both the data and meta-data using OPeNDAP/THREDDS-enabled clients such as IDV, MatLab, web browser, and Excel. In addition, the size of the data that need to be transferred over the network is significantly reduced after server-side processing.

3.4. Presentation Layer

While each research community has its own means of data presentation, several general-purpose interfaces are provided to users for easy data access. Users are able to access data directly via SRB clients including Unix-like SRB commands, Java Interface, web interface, and Windows GUI. In addition, a more user-friendly data portal is currently under development. In the first phase we have developed a data portal that consists of several JSR-168-compliant SRB portlets based on the GridSphere Potal Framework [10, 11]. It enables researchers to browse, search, and download data without having to learn about the underlying system. The data portal has created an immediate impact on the research communities that generate and utilize the data and will soon be used for data access in college-level course work.

4 Metadata Extraction and Standardization

Standardized descriptive metadata can substantially improve the effectiveness of data discovery. For each file registered in SRB, there are system-level metadata automatically generated and registered on primitive attributes such as file name, path, size, date, access control. In addition, users can define their own application-level metadata. All metadata are registered in the MCAT server. Users can easily construct queries based on the metadata. For example, a simple scenario is to search for all Landsat 7 images captured in May 2004 covering the State of Indiana, and to download them from the SRB server.

In the case of LARS datasets, the metadata are processed in two steps. First, several programs are developed to harvest the metadata from various places. The main part of metadata comes from the file header which contains information about the image type, image size, pixel resolution, channel information, etc. A utility called MultiSpec is extended to extract the internal metadata for each image. In addition, there is also information scattered on the LARS web site such as sensor name, satellite, row/path numbers, location, as well as separate text files with description on how the datasets were generated. A Java tool has been developed to collect and enter the information as much as possible before manual processing.

One of the goals of integrating data collections into our data management infrastructure is to facilitate the discovery, access, and sharing of data sets. To achieve this, the application-level metadata are further transformed and standardized. There are several metadata standards in the geo-science and remote sensing community. We have chosen "Content Standard for Digital Geospatial Metadata" (CSDGM) Version 2 and "Content Standard for Digital Geospatial Metadata: Extensions for Remote Sensing Metadata" (FGDC-STD-012-2002), which have the widest acceptance within the community [12, 13]. Both are proposed by Federal Geographic Data Committee (FGDC). Since the LARS dataset has a lot of legacy data that lack some of the mandatory information specified in FGDC standard, we have decided that the final format of LARS metadata will follow a template that is a simplified version of the FGDC standard. We have also developed a detailed document describing the LARS metadata template. In addition to the user-defined attributes registered in the SRB metadata catalog which enables query-based search, there is an XML metadata file associated with each LARS image file, which enables the IndianaView portal [14] to provide users with the metadata file together with the original image.

5. Case Studies

An overview of the current deployment of our data management framework is shown in Figure 2. It demonstrates how different datasets are managed by our common framework and how users could easily access the data via different interfaces. In the case of the NWS data, real-time streaming data are archived in the local file system of a server running THREDDS and LDM. The data are organized based on the name of the source radar station. Each station collection is registered as an HTTP URL object in the SRB MCAT server. The registered URL is the address of a dynamically generated data catalog on the THREDDS server. A user first searches SRB to locate the station of interest. She will then be redirected to a dynamically generated data and metadata catalog managed by the THREDDS server which has native support for the compressed Level II radar data. From there, it is very easy for the user to access and analyze the data.

The climate modeling data are in NetCDF file format, an array-oriented self-describing portable data format. It is a common scenario to append data along one dimension or access a small subset of a large dataset specified by variables. To accommodate this, HTTP URL object is used to link the MCAT metadata with the real data managed by an OPeNDAP server. Similar to the NWS data, a user first searches MCAT to find the modeling data she is interested in. From there, the user is redirected to the WWW interface of the OPeNDAP server which provides services for data attribute description, data download, and data sampling. A user may choose to sample the dataset simply by appending a constraint expression to the URLs given to her visualization or analysis program. It greatly reduces the amount of data that the local program needs to process, as well as the network load for data transfer.

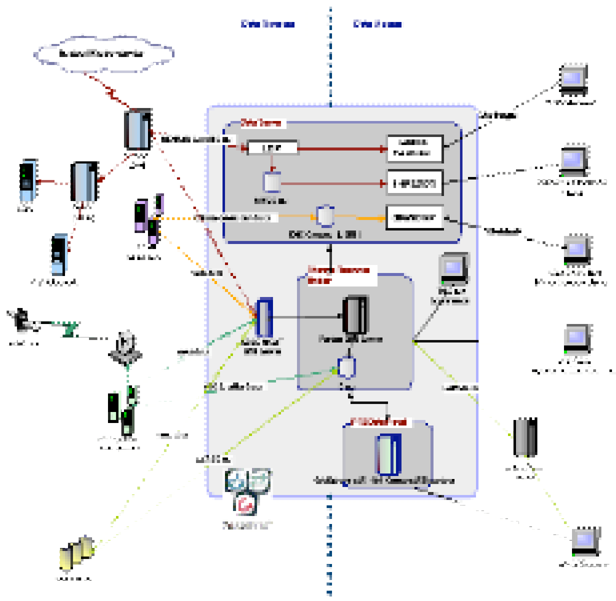


Figure 2. Current Data Collections, Data Management System, and Data Access.

6. Data Access

As demonstrated by the examples in Section 5, one of the key advantages of our data management framework is that there exist multiple ways to access the data sets from various platforms, meeting the needs of various users.

6.1. SRB Clients

Users can interact directly with the SRB server via client-side tools provided by the SRB group, including MySRB, InQ, S-commands. They can also develop customized programs using the client libraries in C, Java, Python, and Perl [15].

6.2. OPeNDAP/THREDDS Clients

In addition to various SRB clients, researchers also wish to use their existing tools to access data. With the added benefit of distributed data management and attribute-based query provided by SRB, the Doppler radar data may be accessed directly using THREDDS-enabled tools such as NetCDF-Java as shown in Figure 3. The NetCDF-formatted climate modeling data are accessible via OPeNDAP-enabled clients, a common way to access data in the environmental research community.

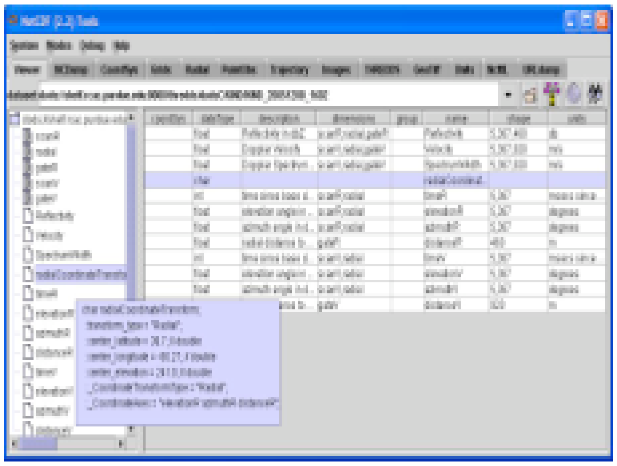


Figure 3Data Access using a THREDDS Client

6.3. Purdue Environmental Observatory Data Portal

As a single access point to various datasets, we have developed the Purdue Environmental Observatory Data Portal that allows researchers in the field and students in environmental engineering to easily access the data without having to learn the underlying software technology [16].

The data portal is developed and deployed using the Gridsphere Portal Framework – an open source web portlet container which allows users to develop and deploy JSR-168-compliant portlets onto a standard portal container like Apache Tomcat. It consists of customized SRB portlets that are web applications providing dynamic content from SRB using JARGON API [17]. The portlets are designed such that it is easy to reuse the portlet modules. Currently, the data portal supports collection browsing, metadata display, metadata query, file download, and preview for HTTP URL SRB objects. The critical requirement is to design the interface to meet the need of end users. For example, we found that the traditional way of searching files via a SQL-like query is not acceptable to most end users who have little knowledge of the query syntax and keyword to choose. We are working on redesigning the query interface by working closely with real-world end users.

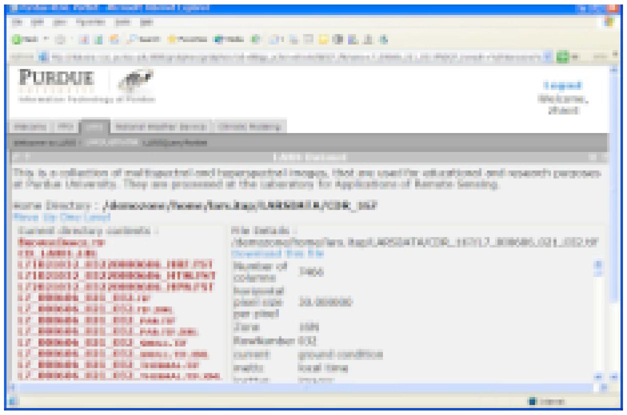


Figure 4. Purdue Environmental Observatory Data Portal

6.4. IndianaView Web Portal

In addition to the general-purpose data portal discussed in 6.3, some specialized web interface has also been developed to access data in the framework. For example, the IndianaView web portal has been developed in collaboration with the LARS group [13]. It allows users to access, preview, and download the LARS data as well as its associated metadata. The web server accesses the data on behalf of the user via a group account. The IndianaView project is part of the AmericaView initiative, a nationwide program to promote the sharing of satellite remote sensing data and technologies [18].

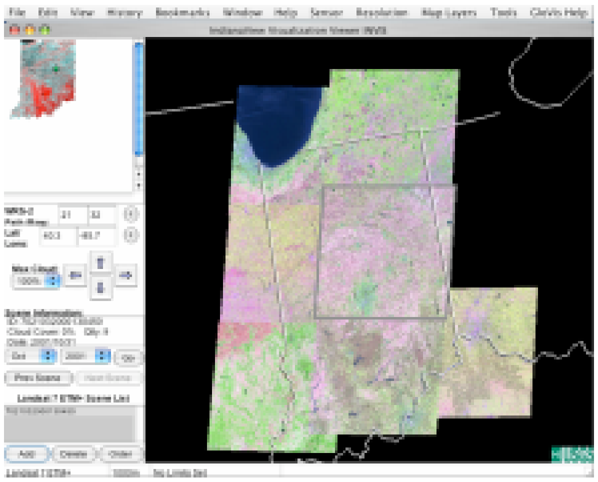


Figure 5. IndianaView Portal

7. Future Work

In the second phase of our data portal development, we will focus on the implementation of features including account management, grid integration, and domain-specific portlet customization. To support data security and collaboration among researchers, we also plan to leverage the built-in features of accounting and zone federation in SRB. In addition, domain-specific work flow and data services will be built to integrate data generation, capture, archive, analysis, and visualization. In the meantime, we are in the initial stage of applying our data management framework to new data sources in the bioscience domain. We also expect to develop drivers that provide real-time data integration into the SRB system.

8. Conclusions

We have presented the Purdue multidisciplinary data management framework. The contributions of our framework include: (1) A generic distributed architecture for effective management of data from different data sources across scientific disciplines, (2) multiple access points for users with different backgrounds and expertise, (3) the integration of OPeNDAP, THREDDS, and SRB functionalities which offers additional server-side post processing capabilities, allowing integrated, efficient data access using existing programs, and (4) a user-friendly data portal consisting of customized SRB portlets serving as a gateway to the datasets.

References

[1] C. Baru, R. Moore, A. Rajasekar, M. Wan, "The SDSC Storage Resource Broker," *Proc. CASCON'98 Conference*, 1998.

[2] MCAT, "MCAT: Metadata Catalog, " SDSC *http://www.sdsc.edu/srb/mcat.html*.

[3] T. Sgouros, "OPeNDAP User Guide," Version 1.14, July 2004.

[4] B. Domenico, J. Caron, E. Davis, R. Kambic, S. Nativi, "2002: Thematic Real-time Environmental Distributed Data Services (THREDDS): Incorporating Interactive Analysis Tools into NSDL," *Journal of Digital Information*, 2.

[5] Laboratory for Applications of Remote Sensing, *http://www.lars.purdue.edu*.

[6 Pu]rdue Terrestrial Observatory, *http://www.itap.purdue.edu/ptol*.

[7]Purdue Level II Radar Data, *http://roskilde.eas.purdue.edu/~level2/index.html*.

[8] M. Huber, R. Caballero, "Eocene El Niño: Evidence for robust tropical dynamics in the "hothourse"," *Science*, 299 (2003) 877-881.

[9] The NetCDF Users' Guide, Data Model, Programming Interfaces, and Format for Self-Describing, Portable Data, NetCDF Version 3.6.1, May 2005.

[10] J. Novotny, M. Russell, O. Wehrens, "GridSphere: An Advanced Portal Framework, " *EUROMICRO 2004*, 412-419.

[1]JSR 168: Portlet Specification *http://www.jcp.org/jsr/detail/168.jsp*.

[12] Content Standard for Digital Geospatial Metadata" (CSDGM) Version 2 (FGDC-STD-001-1998), http://www.fgdc.gov/standards/documents/standards/metadata/v2_0698.pdf.

[13] Content Standard for Digital Geospatial Metadata: Extensions for Remote Sensing Metadata (FGDC-STD-012-2002), *http://www.fgdc.gov/standards/documents/standards/remote_sensing/MetadataRemoteSensingExtens.pdf*.

[14] IndianaView portal, *http://www.indianaview.org/*.

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

[16] Purdue Environmental Observatory Data Portal, *http://gridsphere.rcac.purdue.edu/gridsphere*.

[17] Java API for Real Grids On Networks (JARGON) - version 1.4 *http://www.sdsc.edu/srb/jargon/index.html*.

[18] AmericaView program, *http://www.americaview.org/*.

Globally federated SRB zones*

Yoshimi Iida
KEK Computing Research Center and JST/CREST
Yoshimi.Iida@kek.jp

Yoshiyuki Watase
KEK Computing Research Center
Yoshiyuki.Watase@kek.jp

Stephen J. McMahon
ANU Supercomputer Facility
stephen.mcmahon@anu.edu.au

Takashi Sasaki
KEK Computing Research Center and JST/CREST
Takashi.Sasaki@kek.jp

Glenn Moloney
Univ. of Melbourne
glenn@physics.unimelb.edu.au

Abstract

A world-wide federation of SRB servers was demonstrated successfully among KEK, ANU, KNU, IHEP, ASCC, Krakow and SDSC. We have measured the performance of parallel transfer between the sites and the results are compared with the WAN emulation.

1. Introduction

KEK hosts various accelerator science programs including High Energy Physics, Nuclear Science, Material Science, Bio-medical and so on. These research projects are done by international collaborations and have strong world-wide demands on sharing archived data.

KEK Computing Research Center is continuing efforts on using SRB in a medical application and the Belle experiment. In a project for developing simulation software for particle-based therapy, SRB will be used to share necessary information, e.g. cross section tables, input files and simulation output. The Belle experiment is one of the biggest HEP experiments. They already store two Peta Bytes of data, and are accumulating one Terabyte per day while the accelerator is operating.

Comparing with simple server-client models, SRB zone federation promises independent operation among sites. Every thing is done locally to access local files. It is not necessary to contact servers at any other sites even if they are sharing the single file name space. Only when accessing remote files does remote server access happen. This feature is very attractive for world-wide collaborations who sometimes experience interruption of network connections to other sites.

To demonstrate the capabilities of the SRB, KEK sponsored a workshop in December 2004. People from 7 sites in Australia, Taiwan, China, Korea, Poland, United States and Japan joined it and worked on installing and establishing a world-wide zone federation with great help from Michael Wan from SDSC.

Here we report our experience on world-wide zone federation. Some of our results are already reported in [1].

2. The testbed and the federations of zones

People from 7 sites, ANU, ASCC, IHEP, KNU, Krakow, SDSC and KEK worked together to set up the testbed. In Figure 1, the geographical locations of each sites are shown.

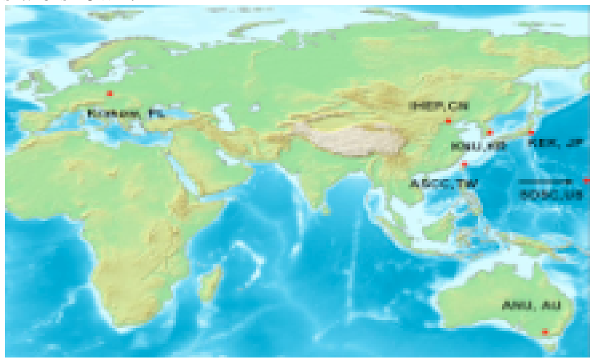


Figure 1: Participating sites

For sites where SRB was not yet installed, they started from scratch during the workshop. At this time, a bug was found in the zonesynch script, but it was fixed quickly by SDSC. After some trial and error, we established the zone federation. We confirmed that zone federation was working among sites during this workshop as shown in Figure 2.

The network connections are normal production lines from KEK to other sites. The associated bandwidths are shown in Figure 2. At each site, a SRB server, a storage

* This research was partially sponsored by JST/CREST. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the Government, or any person or agency connected with them.

resource, and an MCAT server were installed. For this test, SRB version 3.1.2p was used at all sites and each site was set up as a separate SRB zone. At the KEK local SRB zone, the MCAT server was configured using an IBM DB2 database management system. We also configured a SRB server for the Belle data files with NFS access to the Belle data server. The necessary ports were opened on the KEK firewall to allow connections from the fixed IP addresses of the other sites

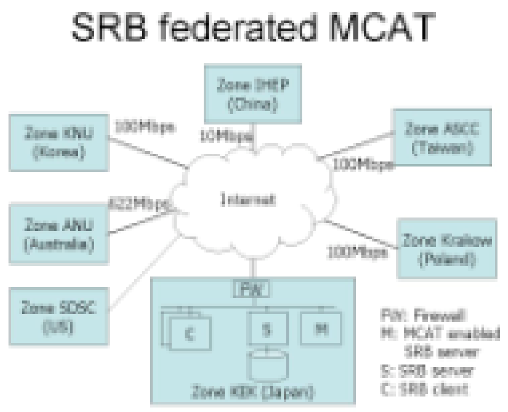


Figure 2: Federated zones

.

3. Performance Measurement

The data transfer performance was measured among 6 sites excluding SDSC.

A detailed study of the data transfer performance was previously conducted between the KEK and ANU zones. These two sites have a very long transfer distance and correspondingly high latency for network communications. Initial tests showed a transfer rate between these zones of about 120KB/s for a single threaded transfer. This transfer rate wasn't enough to support the Australian share of the simulations. In order to improve the data transfer performance the TCP window size was increased to 4MB on the ANU server and 8 MB on the KEK server. With a higher TCP window size, idle time will decrease since more data is shipped between packet acknowledgements. For high latency networks, this was expected to improve the performance and the result shows that the data transfer rate increased to about 260KB/s.

Tests were then carried out using the parallel transfer feature of SRB that is enabled through the '-m' option of the Sput and Sget commands. Parallel transfers are the default for server-to-server transfers such as Sreplicate and Scp commands in SRB. Files ranging from 0.5MB to 500MB were used in the tests as the number of threads that SRB chooses to use is determined by the file size. SRB server parameters controlling the number of threads are SizePerThread and MaxThreads. By default SizePerThread is set to 30MB and MaxThreads is set to 4. After some test was done with these default settings, SizePerThread was then set to 2MB and MaxThreads was set to 16 and the tests were repeated. The results for some parallel transfer testing between ANU and KEK are shown in Figure 3 with

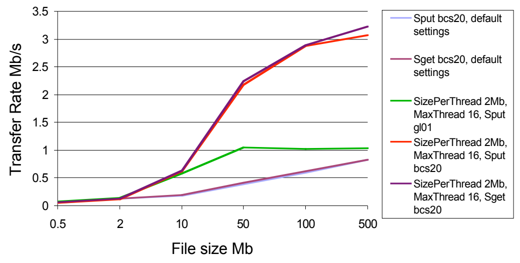


Figure 3: Parallel transfer performance between ANU and KEK

We obtained better results with SizePerThread of 2MB and MaxThread 16 than the default values. The following federation tests between all 6 sites was carried out with a 100MB data file transfer utilizing SRB's 16-thread parallel transfers. The results of data transfer from KEK to remote sites for various sizes of data file are shown in Figure 4. Also the transfer tests with a fixed data size of 100MB among 6 sites are listed in Table 1.

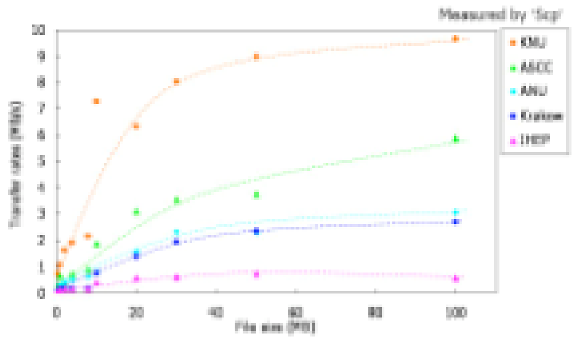


Figure 4: Transfer performance with various file sizes

Source	Destination				
	KNU	ASCC	ANU	Krakow	IHEP
KEK	9.65	5.89	3.06	2.69	0.59
KNU	-	5.68	4.45	4.47	0.61
ASCC	7.54	-	2.84	5.09	0.64
ANU	4.79	2.53	-	3.44	0.58
Krakow	1.21	1.64	4.19	-	0.60
IHEP	0.70	0.39	0.44	0.25	-

Table 1: Transfer rates among sites with a file with 100MB in size (MB/sec)

In Table 2, the measured transfer rate versus the latency (Round Trip Time) is shown. As expected, although ANU has higher bandwidth than other sites, the performance rate was not good enough because of higher latency.

Destination	RTT(msec)	Nominal Bandwidth (Mbps)	Transfer Rate (Mbytes/s)
IHEP	502	10	0.59
Krakow	327	100	2.69
ANU	292	622	3.06
ASCC Taiwan	33	100	5.89
KNU	23.6	100	9.65

Table 2: Transfer performance between KEK and other sites. The latency was measured in December 2004.

The setting of TCP window size means that the specified value was requested for use but actual size is determined by best effort. Once a packet loss happens, the size is reset to the default one. The TCP logic for increasing the window size after packet loss recovers the specified window size very slowly. For high latency networks, this process takes more time (slower negotiations) and the transfer rates go down. If the quality of the networks is not good enough, packet loss happens often and this becomes the reason for a lower transfer rate then the nominal bandwidth. We confirmed this behavior by measuring performance under WAN emulation using the NIST Net [2].

4. Conclusions

A federation of SRB servers in world-wide was demonstrated successfully among 7 sites. The functionality of SRB is very rich for use in WAN environments. We have measured the performance of parallel transfer provided in SRB among those sites. The result shows that the latency and quality of the network is the key to obtain better performance and that higher network bandwidth alone is not sufficient.

5. Acknowledgements

We would like to thank following people who joined the workshop at KEK and contributed to this work.

Michael Wan (SDSC, US),
Gang Chen(IHEP,Beijing),
Ma Mei (IHEP, Beijing),
Henryk Palka (Krakow, Poland),
Huimin Lin (ASCC, Taiwan),
Kihwan Kwon (CHEP, Korea) and
Nobu Katayama (IPNS, KEK)

Especially, the authors are very much grateful to Michael Wan and the SRB team at SDSC. Mike prepared the lectures and helped us very much to establish the zone federation. The SRB team provided us very fantastic support on their software.

References

[1] **Y. Iida *et al.*, "SRB SYSTEM AT BELLE/KEK",** *KEK-PREPRINT-2004-57, Oct 2004.* Computing in High Energy and Nuclear Physics (CHEP'04), Interlaken, Switzerland, September, 27 - October 1, 2004.

[2] **Mark Carson and Darrin Santay,** *NIST Net - A Linux-based Network Emulation Tool, June 2003, Computer Communication Review*. http://snad.ncsl.nist.gov/itg/nistnet/

.

SRB Data Grid and Compute Grid Integration via the EnginFrame Grid Portal

Francesco Beltrame
DIST, Università di Genova
Genova, Italy
francesc@dist.unige.it

Paolo Maggi
DAUIN, Politecnico di Torino
Torino, Italy
paolo.maggi@polito.it

Maurizio Melato
Nice S.r.l.
Cortanze (AT), Italy
maurizio@nice-italy.com

Elisa Molinari
DIST, Università di Genova
Genova, Italy
elisa@bio.dist.unige.it

Riccardo Sisto
DAUIN, Politecnico di Torino
Torino, Italy
riccardo.sisto@polito.it

Livia Torterolo
DIST, Università di Genova
Genova, Italy
livia@bio.dist.unige.it

Abstract

Support for distributed computation has been one of the earliest areas of exploration in Grid computing and many tools have been developed in order to allow people to run jobs or use services on distributed and heterogeneous computing resources.

In this paper we will show how the EnginFrame Grid portal can be used to integrate this kind of tools with the data Grid capabilities of the Storage Resource Broker (SRB).

In particular, we will describe how to use the EnginFrame Grid portal to develop innovative services making use of both the SRB capabilities of sharing and managing data and metadata in an heterogeneous and distributed environment, and the power and flexibility provided by a computational Grid.

We will also present two demonstrative services exploiting the described EnginFrame-SRB integration: a fire propagation simulation and a neuroscience application.

1. Introduction

Grid computing is one of the most important topics appeared in the computing field in the last decade. This is due to the increasing needs arising in the IT world: the need for more computational power in order to solve very complex problems in a variety of scientific or industrial fields and the need to leverage the usage of the information technology resources owned by organizations.

Support for distributed computation has been one of the earliest areas of exploration in Grid computing and in the last years many tools have been developed to allow people to take advantage of distributed and heterogeneous computational environments.

However, the Grid is not only a computing infrastructure for computationally-intensive applications, but it is also a technology that allows the controlled sharing and management of large amounts of distributed data. Very large volumes of archived data can be exposed on the Grid in a secure, authorized and efficient way.

One of the greatest challenges of Grid computing is the complete integration of heterogeneous computing systems and data resources with the aim of providing a global computing space. The achievement of this goal will involve revolutionary changes in the field of computation, because it will produce innovative services that, at the same time, will be able to exploit the great computational power and flexibility offered by a computational Grid and have access to large amounts of geographically distributed data. We can envision, for example, Grid enabled applications that will be able to cope with problems like protein folding, financial modeling, earthquake simulation, climate/weather modeling, etc.

Grid portals can help to reach this aim because they provide the user community with an intuitive and simplified interface to exploit the Grid, its services and resources while supplying the developers with tools that can greatly simplify the integration of data Grid and computational Grid components.

In this paper, we will show how the EnginFrame Grid portal[1] can be used to develop innovative services that integrate applications that are not only computationally-

[1] http://www.enginframe.com

intensive, and so particularly suited for a computational Grid, but also require controlled access to large amounts of geographically distributed data as provided by the Storage Resource Broker (SRB) [1], thanks to its data Grid capabilities.

In particular, in section 2, we will first introduce the EnginFrame Grid portal. Then, in section 3, we will explain how EnginFrame can be extended to add support for SRB.

Section 4 describes a couple of demonstrative services we have developed to test and evaluate our EnginFrame-SRB integration. The first one is a service for the dynamic simulation of fire propagation in a building and has been developed to show how our EnginFrame-SRB integration enables legacy applications to run in a data and computational Grid environment. The second one is an application in the neuroscience area for the analysis of PET images for the early diagnosis of the Alzheimer's Disease. It shows how our EnginFrame-SRB integration can be used to easily add a friendly user interface to existing data Grid enabled applications and, at the same time, to make them run in a computational Grid environment.

Section 5 summarizes conclusions and future enhancements.

2. The EnginFrame Grid Portal

EnginFrame is a Web-based innovative technology, by the Italian company Nice S.r.l.[2], that enables the access and the exploitation of Grid-enabled applications and infrastructures.

It allows organizations to provide application-oriented computing and data services to both users (via Web browsers) and in-house or ISV applications (via SOAP/WSDL based Web services), hiding all the complexity of the underlying Grid infrastructure.

In particular, EnginFrame greatly simplifies the development of Web portals exposing computing services that can run on a broad range of different computational Grid systems (including Platform LSF[3], Sun Grid Engine[4], Altair PBS[5], Globus[6], LCG[7] and EGEE gLite[8]).

EnginFrame supports several open and vendor-neutral standards and seamlessly integrates with JSR168 compliant enterprise portals, distributed file systems, GUI virtualization tools and different kinds of authentication systems (including Globus GSI).

[2] http://www.nice-italy.com
[3] http://www.platform.com/Products/Platform.LSF.Family/
[4] http://gridengine.sunsource.net/
[5] http://www.altair.com/software/pbspro.htm
[6] http://www.globus.org/
[7] http://lcg.web.cern.ch/LCG/
[8] http://glite.web.cern.ch/glite/

Because EnginFrame greatly simplifies the use of Grid-enabled applications and services, it has already been adopted by numerous important companies all over the world.

EnginFrame is also well known in the Grid research world for being the technology on which is based GENIUS[2], the official Grid portal of the European project EGEE[3].

Thanks to the specific grant provided by the INFN Grid Project[9] and to the agreement between INFN and Nice S.r.l., the GENIUS code is open source and the EnginFrame license is free of charges for the academic and research organizations.

As we said, EnginFrame allows exposure of computing services solutions on the Web. This is done by simply developing XML-based descriptions of the services and scripts representing the actual services implementations.

EnginFrame receives incoming requests from the Web, authenticates and authorizes the requests and then executes the required actions into the underlying Grid computational environment.

Then, EnginFrame gathers the results and transforms them into a suitable format before sending the response to the client. Transformation of results is performed according to the nature of the client: HTML for Web browsers and XML for Web services client applications.

For each submitted service, a data area staging the service input and output files is created on the file system. This area is called *Spooler*.

Most of the information managed by EnginFrame are described by dynamically generated XML documents. The source of such information is typically the service execution environment: an *XML abstraction layer* aims to submit service actions and translate raw results coming from the computational environment into XML structures.

The XML abstraction layer is designed to decouple EnginFrame from the actual Grid working environment, hiding the specific Grid technology solution.

This characteristic, together with other important ones, makes possible to easily extend EnginFrame functionalities by developing ad-hoc plugins for specific computational and data Grid middlewares or legacy applications.

To support the integration of data Grid middleware solutions, EnginFrame introduces the concept of *Virtual Spoolers*. Virtual Spoolers represent distributed data areas that reside outside the EnginFrame spoolers file system, but that can be remotely accessed by EnginFrame itself through the targeted data Grid technology. The structure and the content of a Virtual

[9] http://grid.infn.it/

Spooler is described by a dynamically generated XML document.

3. EnginFrame-SRB integration

While designing the integration of SRB with the EnginFrame framework we envisioned two possible solutions. The first one is a light, low-coupling integration consisting of a simple plugin that interacts with SRB through the *S-commands* provided by SRB itself. The second one is a tighter integration requiring an extension to the EnginFrame spooler concept in order to seamlessly and directly embrace SRB collections. In this case, the EnginFrame core would see SRB collections as "standard" spoolers.

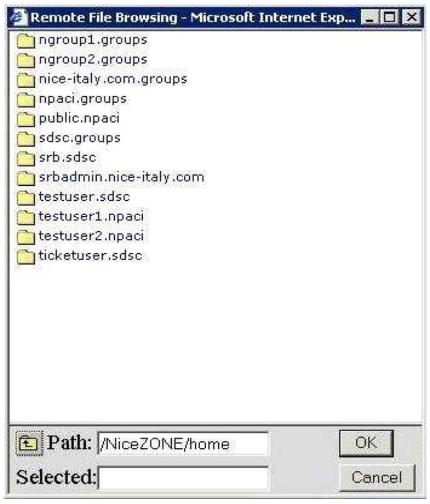


Figure 1. SRB enabled Remote File Browsing.

The first approach is well decoupled from the EnginFrame internal architecture but less reliable and robust since parsing of raw S-commands' output and transformation to XML is required. Nevertheless, it quickly provides valuable results for a first prototype implementation.

The second is a cleaner and more robust approach but implies more development efforts to write new Java modules to be plugged into the EnginFrame core architecture.

For the integration prototype presented in this paper we considered the first approach more adequate to quickly exploit the technology and build some test cases.

The first task we targeted was to extend the EnginFrame feature called *Remote File Browsing* (RFB) in order to support the SRB data Grid. EnginFrame RFB allows users to browse remote file systems and select files from within a standard Web browser. The files selected by the user are then used as parameters at the service submission phase.

To extend standard RFB functionalities to make RFB work with a different data provider, a simple script must be written. In our case a shell script has been developed. This one interacts with SRB using the S-commands `Scd`, `Spwd` and `Sls` and then parses the commands' raw output through standard Unix tools like `awk` and `sed` in order to produce a XML document representing the required collection in the SRB virtual file system (Figure 1).

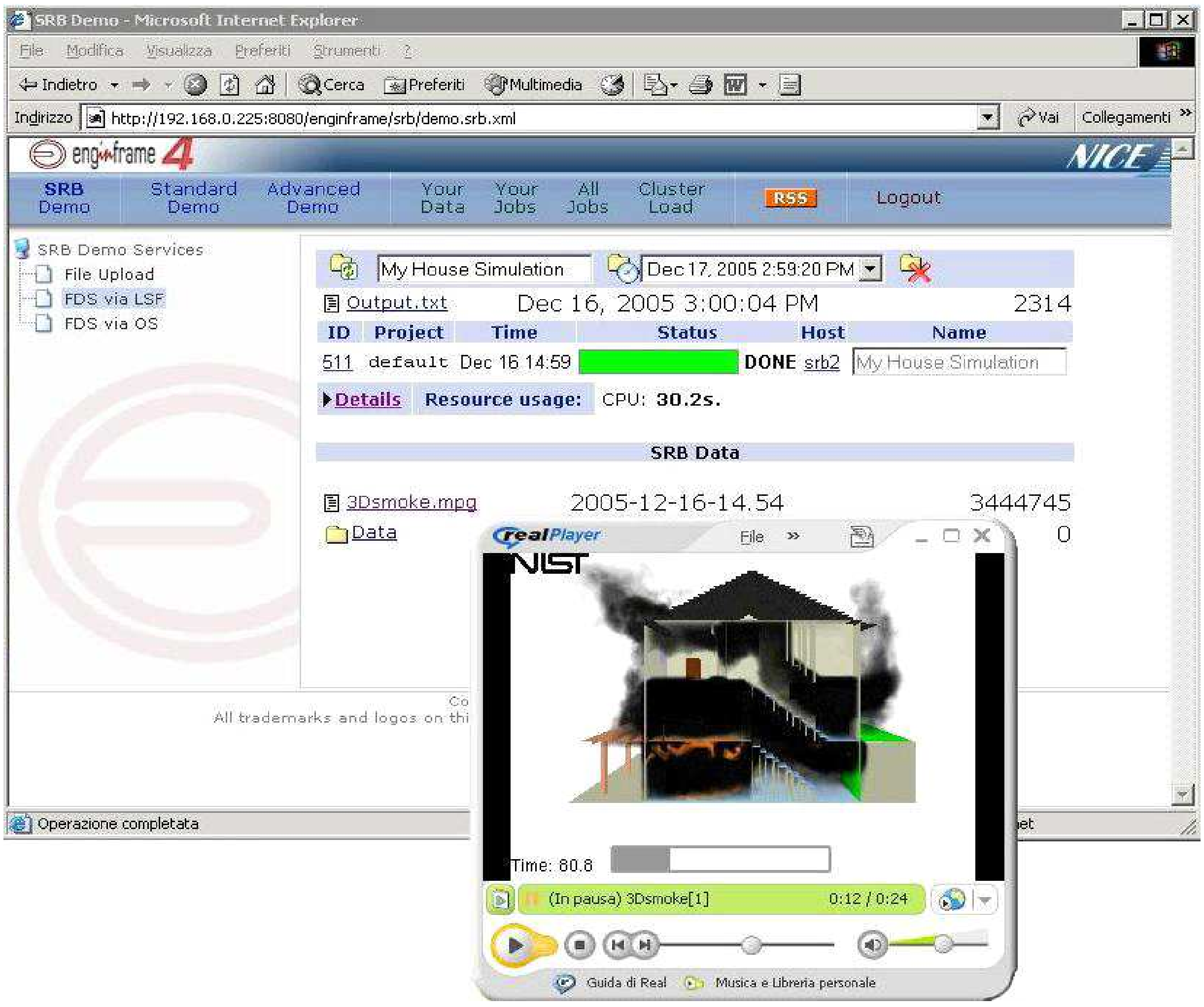
In a similar way, by exploiting the Virtual Spoolers feature we previously described, the standard EnginFrame spooler view has been extended to seamlessly show files and collections stored in an SRB virtual file system (Figure 2).

Both the SRB RFB and Virtual Spooler features support the view of metadata associated to SRB files. Metadata are queried via the `Smeta` command and the result is structured into a custom XML representation that is transformed into a piece of JavaScript and HTML code by applying an ad-hoc XSL transformation.

Finally, the plugin allows to send files to SRB through the `Sput` command and retrieve data through `Scat`. Files can thus be uploaded from the user's local host through the Web browser and EnginFrame into SRB; in the same way files can be downloaded from SRB through EnginFrame and the Web browser into the user's local file system. Using the *streaming download* feature of EnginFrame, files can also be streamed from SRB to the user's Web browser. This feature can be especially useful when, as in the case reported in Figure 2, the streamed file contains a video clip.

One of the major values of the integration work we presented is that, from a user point of view, it does not change the usual way EnginFrame manages data. The end user can be so totally unaware about SRB since all the SRB complexity is completely hidden by EnginFrame.

By using the SRB plugin it is also possible to enhance the standard data management functionalities of EnginFrame by means of the SRB metadata. The user, through specific services or service options, can add metadata to files stored in SRB and view/query metadata when remotely browsing SRB data from RFB or from EnginFrame spoolers.


SRB Demo - Microsoft Internet Explorer
File Modifica Visualizza Preferiti Strumenti ?
Indietro Cerca Preferiti Multimedia
Indirizzo http://192.168.0.225:8080/enginframe/srb/demo.srb.xml
Vai Collegamenti
enginframe 4
NICE
SRB Demo
Standard Demo
Advanced Demo
Your Data
Your Jobs
All Jobs
Cluster Load
RSS
Logout
SRB Demo Services
File Upload
FDS via LSF
FDS via OS
All trademarks and logos on thi
Operazione completata
realPlayer
File
NIST
Time: 80.8
(In pausa) 3Dsmoke[1] 0:12 / 0:24
Guida di Real
Musica e Libreria personale

My House Simulation — Dec 17, 2005 2:59:20 PM

Output.txt — Dec 16, 2005 3:00:04 PM — 2314

ID	Project	Time	Status	Host	Name
511	default	Dec 16 14:59	DONE	srb2	My House Simulation

Details — Resource usage: CPU: 30.2s.

SRB Data

3Dsmoke.mpg	2005-12-16-14.54	3444745
Data		0

Figure 2. **SRB enabled Virtual Spooler.**

4. Demonstrative services

In order to test and evaluate our SRB plugin, we have developed two different demonstrative services. The first one is meant to demonstrate how EnginFrame together with our SRB plugin can be used to enable the execution of legacy applications in data and computational Grid environments. The second one, instead, shows how EnginFrame can be used to add a Web interface to data Grid enabled applications and how our solution enables the easy integration of such applications in a computational Grid environment.

4.1. The FDS demonstrative service

The first demonstrative service we have developed implements the following scenario: a couple of engineers, working in different sites, collaborate to analyse the fire propagation dynamics in a building using the Fire Dynamics Simulator (FDS)[10], a tool developed and maintained by the Fire Research Division in the Building and Fire Research Laboratory (BFRL) at the National Institute of Standards and Technology (NIST).

The first engineer is the designer of the building, while the second one is a specialist in fire propagation analysis.

The Grid portal Web interface provides the first engineer with a service to upload the design of the

[10] http://fire.nist.gov/fds/

building and store it into the SRB data Grid. To better describe the files contents, the same service allows to attach to the files SRB metadata about the project.

To run the fire propagation simulation, the Grid portal exposes another service for the second engineer.

The simulation service allows to select the building design file via the extended EnginFrame Remote File Browsing for SRB and to submit the simulation to a computational Grid. While browsing and looking for files in SRB, the user can take advantage of the SRB metadata system.

Service implementation performs some preliminary operations to get the input file from SRB and then it runs a FDS simulation. At the end of the simulation, FDS output files are stored into SRB and post-processed to produce a video file containing a graphical representation of the fire propagation in the building.

EnginFrame Grid portal provides both the engineers with a Web interface to access the simulation output files and to stream the video file (Figure 2).

4.2. The SPM demonstrative service

The second service we created consists in the "Webification" of an innovative data Grid enabled application for neuroscientists developed by the Bio-Lab of the University of Genoa and based on the Statistical Parametric Mapping (SPM)[11] software developed by the Institute of Neurology at the University College in London and largely used by the neurological scientific community [4].

The application has been designed to help neuroscientists in the early diagnosis of Alzheimer's disease (AD) through the quantitative comparison of PET images of non pathological cases stored in a SRB data Grid [5].

Our service allows the physicians to easily access to a large set of PET/SPECT images stored on distributed SRB servers in different hospitals and run the SPM-based tool for the analysis.

It is worth noting that SRB plays a crucial role in this application. In fact, normally a very few PET images of healthy subjects to use for the comparison are owned by a single hospital. Hence, the capability of SRB to aggregate in a single virtual file system data objects stored in distributed and heterogeneous physical resources is very useful for giving the physicians seamless access to a distributed set of images that is large enough to guarantee accurate results without the need of actually storing all the needed images in all the hospitals.

[11] http://www.fil.ion.ucl.ac.uk/spm/

Furthermore, SRB satisfies all the security and privacy requirements of this kind of applications.

The application by Bio-Lab has been first developed as a command line application that makes use of the

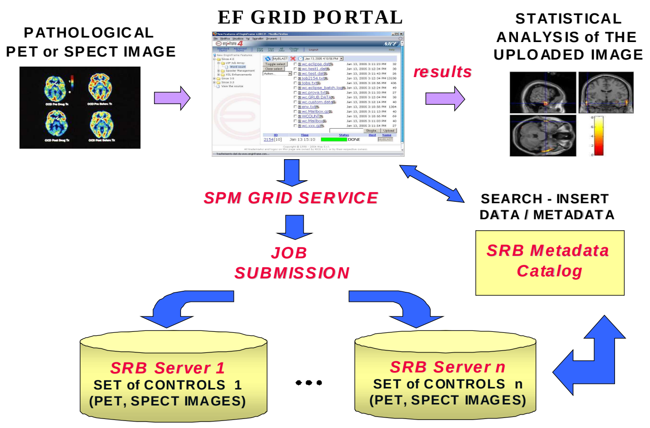


Figure 3. Functional description of the SPM service.

SRB API to get access to the SRB data Grid. Then, it has been integrated into EnginFrame using the SRB plugin we presented in this paper in order to provide the physicians with a very friendly user interface that hides all the inherent complexity of the underlying Grid architecture.

The integration with EnginFrame has also allowed to easily extend the original application to execute the analysis phase as a job in a computational Grid environment.

As shown in Figure 3, the resulting service provides the physicians with the following functionalities:

User Access: users can log on the portal and access to SPM service through a user authentication mechanism. The system is able to define specialized authorization rules that give different users different permissions within the system based on their Grid identities.

Data uploading: users can upload through the EnginFrame portal PET images of pathological patients in order to analyse them with the SPM tool and put new PET images in the SRB space in order to increase the number of healthy cases for comparison.

Data visualization: by queries the Metadata catalog, users can search PET studies and patient informations stored on distributed and heterogeneous SRB servers, download them from the portal on the users' local machines and visualize them.

Job submission: the Grid portal submit the SPM analysis as a job for a computational Grid. The job first extracts PET image information needed by the SPM tool from different SRB servers using the SRB API and then uses these results to run statistical analyses.

Result visualization: when a job is terminated, the user can download the resulting images from the portal on his local machine and visualize them.

5. Conclusions

In this paper we have proposed a method to integrate SRB data Grid and compute Grid components via the EnginFrame Grid portal.

Two different demonstrative services have been described to show how it is possible to exploit this type of integration.

The SRB plugin we presented is still a prototype. We are still working on it both for improving its overall quality and for adding new features.

We expect it will be useful in the construction of general data exchange infrastructures for wide range of different application areas.

It will be also very useful for the development of innovative services that need to make use of both the SRB capabilities of sharing and managing data and metadata in an heterogeneous and distributed environment, and the computational power and flexibility provided by a computational Grid.

References

[1] A. Rajasekar, M. Wan, R. Moore, “MySRB & SRB – Components of a Data Grid”, 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 2002), 301.

[2] A. Andronico, R. Barbera, A. Falzone, P. Kunszt, G. L. Re, A. Pulvirenti, and A. Rodolico, “Genius: a simple and easy way to access computational and data grids,” *Future Gener. Comput. Syst.*, vol. **19**, no. **6** (2003), 805–813.

[3] R. Berlich, M. Kunze, K. Schwarz, “Grid computing in Europe: From research to deployment,” in *Australasian Workshop on Grid Computing and e-Research (AusGrid2005),* ser. CRPIT, R. Buyya, P. Coddington, A. Wendelborn, Eds., vol. **44**. Newcastle, Australia: ACS, (2005) 21–27.

[4] Y. Jeong, S. S. Cho, J. M. Park, S. J. Kang, J. S. Lee, E. Kang, D. L. Na, S. E. Kim, “^{18}F-FDG PET Findings in Frontotemporal Dementia: An SPM Analysis of 29 Patients”, *Journal of Nuclear Medicine*, Vol. **46** No. 2, 233-239.

[5] K. Herholz, H. Schopphoff, M. Schmidt, R. Mielke, W. Eschner, K. Scheidhauer, H. Schicha, W.D. Heiss, K. Ebmeier, “Direct comparison of spatially normalized PET and SPECT scans in Alzheimer’s disease”, *Journal of Nuclear Medicine,* Vol. **43** (2002), 21-26.

The Storage Resource Broker and e-science in the UK

L. Blanshard, R. Downing, G. Drinkwater, D. Hanlon, K. Kleese van Dam, L. Roberts, R. Tyer.

CCLRC Daresbury Laboratory

P. Berrisford, G. Brown, K. Haines, C. Moreton-Smith, A. Hasan.

CCLRC Rutherford Laboratory

Abstract

The e-science Data Management Group is tasked with providing advice, software and support for the data management needs for all UK e-science projects and the Storage Resource Broker plays an important role in many of the projects. In this paper we describe some of the projects that use the SRB and describe the approach and our experience in using the SRB for these projects.

1. Introduction

The UK e-science program [1] aims to provide the information technology infrastructure necessary to allow UK-funded scientific projects to make the most use of their findings, by allowing cross-collaboration to occur and by providing sufficient computing capacity to exploit large-scale data-sets.

The Data Management Group (DMG) [2] is part of the UK Council for the Central Laboratory for the Research Councils (CCLRC) e-science centre and is tasked with providing data management solutions for all e-science projects. The DMG must provide systems that can be readily adapted to projects from a wide-variety of disciplines ranging from Geology, Chemistry, Physics to Biology. A key component of the data management system is the Storage Resource Broker (SRB) [3] which provides the underlying management of data.

In this paper we describe a few of the projects the DMG works on and how the SRB was implemented to help with the data management of those projects. We also describe our experience in using the SRB.

[*] This work has been funded by a range of UK agencies incl. the e-Science Programmes of the Natural Environmental Research Council, the Engineering and Physical Science Research Council, the Council of the Central Laboratory of the Research Councils, the Biotechnology and Biological Sciences Research Council and the Joint Information Systems Committee.

2. The e-minerals/e-materials Projects

The e-minerals project [4] is aimed at providing a mini-grid test-bed to allow Earth scientists and Chemists to simulate environmental problems at the molecular level, such as the transport of pollutants, weathering and the containment of high-level radioactive waste.

The mini-grid comprises of computational clusters and storage devices housed at collaborating institutes, figure 1 shows a schematic of the setup.

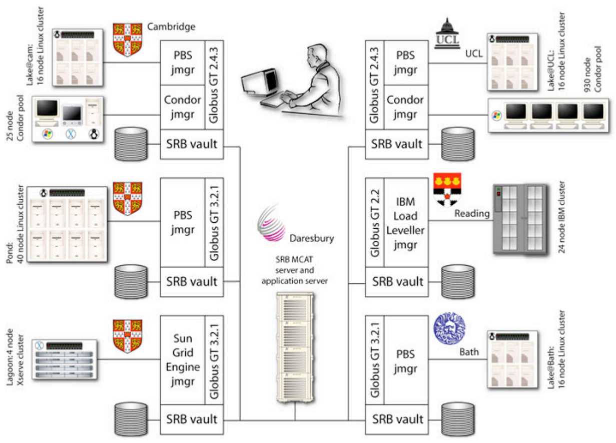


Figure 1: A schematic showing the layout of the e-minerals mini-grid (taken from [5]).

The mini-grid uses Condor pools and PBS (Portable Batch System) managed clusters to carry out the simulations and makes use of SRB resources at each site to hold the simulated data. The SRB holds the logical-to-physical file mapping ensuring that the user does not need to know the physical location of the data. All the SRB resources are GSI-enabled and are registered in the metadata catalogue at Daresbury Lab. The mini-grid is accessed through the e-minerals web-portal [6] and provides a thin-client that hides much of the complexity of dealing with grid-middleware tools. The e-minerals portal integrates the CCLRC data portal used to locate the data and the CCLRC HPC portal used to submit jobs to the grid.

The data portal is used to seamlessly access the e-minerals metadata catalogue and the SRB MCAT to locate data-sets of interest. The HPC data portal is used to create and submit jobs using Condor tools to create workflows including post- and pre- processing steps that contain retrieval and storage of SRB data.

The e-materials project [7] is dedicated to providing the IT infrastructure to aid in the simulation of complex materials. The project's architecture is similar to that of the e-minerals project: containing an SRB to hold simulated data and a computational grid to perform the simulations. Before the data is stored in the SRB a subset containing information on crystal structures is extracted and stored in a relational database to allow easier display and comparison of crystal properties.

Currently over 200,000 files have been stored in the SRB amounting to more than 160GB of data.

3. The ISIS and Diamond Projects

The ISIS facility [8] is currently the world's leading pulsed neutron and muon source and is located at Rutherford Appleton Laboratory in the UK. The facility provides muon and neutron beams to experimenters from such diverse fields as physics, geology, biology and engineering. The ISIS computing group provides a data acquisition system that stores the experimenter's data on a central NT disk farm where the data is then copied daily to the RAL tape system (Atlas Data Store, ADS) for archival.

The computing group is currently developing a system where the SRB will manage the process of archiving the data to the ADS. The system comprises of an SRB server on the NT disk farm, a server for the ADS and a metadata catalogue (ICAT) [9] containing experiment-specific information. The data will then be accessed through the data portal that will interface to the SRB and ICAT. As many of the data-sets are small in size the use of containers is essential in order to make efficient use of the tape resource. Currently, more than 5620 files stored in more than 700 containers is stored in the test system.

The new Diamond facility [10] will provide a synchrotron light source to allow scientists to study the structure of materials in much more detail than ever before. As with the ISIS facility, Diamond will host scientists from a wide variety of fields. The facility will become operational in 2007.

The DMG is currently working with the Diamond computing staff to develop a system capable of managing the experimental data covering ingestion at the beam-lines through to archival in the RAL data store. A first phase test system (figure 2) is now in place allowing performance testing for data ingestion and movement and providing the necessary infrastructure for primarily Java-based application development.

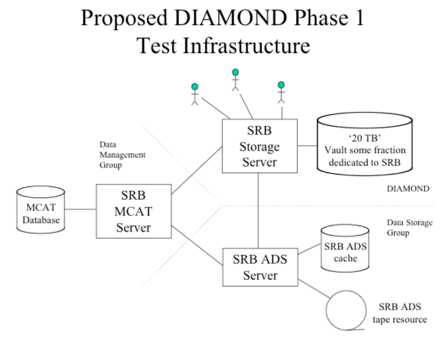


Figure 2: Schematic of the first phase test system for the Diamond facility (courtesy of P. S. Berrisford).

As with the ISIS facility, a large number of small files are expected making the use of containers important. Typically, approximately 1TB of data per day will be produced by the facility. Federation of the ISIS and Diamond SRB systems is considered, as it is possible that a researcher may be interested in data from both facilities.

4. The National Grid Service and the Integrative Biology Project

The National Grid Service (NGS) [11] went into production in September 2004 and provides a production grid of computing and data resources for a wide-range of UK research activities. The resources are provided by universities and computing centres spread throughout the UK. The data grid comprises of SRB resources at Leeds, Manchester, Oxford and CCLRC-RAL with the MCAT-enabled server at RAL and provides multiple terabytes of disk space to NGS users. Users who register to use the NGS are automatically registered to use the SRB and authentication using GSI certificates or passwords is allowed. There is currently more than 800GB of data stored in the NGS SRB corresponding to more than 800,000 files.

The integrative biology project [12] aims to provide an IT infrastructure to facilitate the modeling of biological processes, initially modeling heart arrhythmia and tumor growth. The project includes access controlled secure data management through the SRB and tools to provide visualization of the simulation as well as tools to provide interactive control of the simulation. A high-level diagram of the services is shown in figure 3.

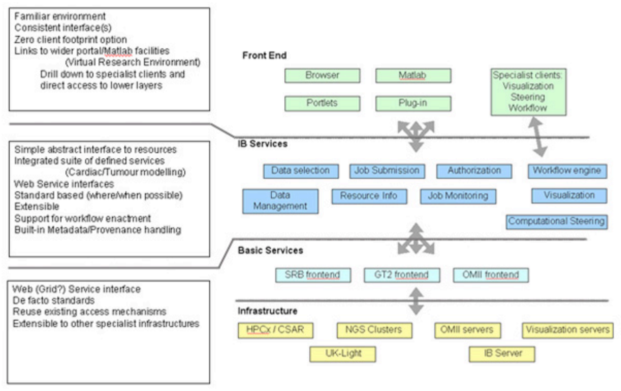

Familiar environment
Consistent interface(s)
Zero client footprint option
Links to wider portal/Matlab facilities
(Virtual Research Environment)
Drill down to specialist clients and direct access to lower layers
Front End
Browser
Matlab
Portlets
Plug-in
Specialist clients: Visualization Steering Workflow
IB Services
Simple abstract interface to resources
Integrated suite of defined services
(Cardiac/Tumour modelling)
Web Service interfaces
Standard based (where/when possible)
Extensible
Support for workflow enactment
Built-in Metadata/Provenance handling
Data selection
Job Submission
Authorization
Workflow engine
Data Management
Resource Info
Job Monitoring
Visualization
Computational Steering
Basic Services
SRB frontend
GT2 frontend
OMII frontend
Web (Grid?) Service interface
De facto standards
Reuse existing access mechanisms
Extensible to other specialist infrastructures
Infrastructure
HPCx / CSAR
NGS Clusters
OMII servers
Visualization servers
UK-Light
IB Server

Figure 3: Schematic of the services comprising the Integrative Biology project (taken from [13]).

The current release uses the NGS to provide security, grid computing nodes to facilitate the simulation and visualization and data storage by means of the SRB. Currently, the project has stored more than 300GB of simulated data in SRB collections and has approximately ten people actively accessing the data.

5. Experience with the SRB

For all projects the SRB fills a desperate need: for easy access to data residing at geographically distributed data resources. The system has proved to be relatively simple to setup and use. Some of the projects have simply taken Scommands and incorporated them into scripts to provide tools that the clients of the system can more easily use; other projects have a requirement of platform independence for client-side tools requiring tools to be developed using Jargon the Java API.

Many of the projects require access to an additional metadata catalogue and we have found the use of the data portal developed by Drinkwater, Sufi et al [14] to be invaluable in providing a common interface to both the SRB and the project-specific metadata catalogues making it possible to extract data by querying the project metadata catalogue.

Almost all the projects are stressing the SRB in different ways: large quantities of small files stored in containers requiring large containers to be constructed and bulk operations to ingest the data, complex collection names and many large-scale operations within SRB space (such as moving a file from one resource to another). In some cases projects have highlighted problems within the SRB that the SRB developers have been quick to resolve providing advice or patches. We have found the SRB bugzilla system and the srb-chat mailing list to be a good resource for problem solving with suggestions coming from the SRB developers and also from the other members of the SRB community.

Initially, we found upgrades from one version of SRB to another to be difficult, but the SRB developers have worked hard on architecting the system such that upgrades are relatively simple with patches that can be applied to the MCAT, and, in more recent cases, even allowing backwards compatibility so clients are not forced to upgrade their code.

We have found the need to develop a number of tools to aid in providing a production system based on the SRB. Some of these tools are the topic of other papers [15] and we are looking at providing these tools back to the SRB so they can be of benefit to others. We have also found it essential to provide good documentation complete with examples on the SRB for potential and existing users so that they can understand the capabilities of the system and how to use it.

The introduction of zones and extended schema to the SRB are features that a number of projects are interested in using. Zones allow more federation of one SRB system with another making it possible for a user to access a remote SRB system without having to remember the new SRB hostname and other attributes. The extended schema allows the MCAT schema to be extended to include tables containing project-specific metadata, or more file-specific information allowing for a much richer metadata catalogue, this can be of interest where the extended metadata is very closely coupled to the SRB metadata and does not merit access outside of the SRB.

There is also interest in investigating the Matrix web-services to see how they could be used in a number of projects possibly simplifying the architecture of some systems.

6. Conclusion

The Data Management Group is responsible for providing data management solutions for e-science projects. One of the key elements of these solutions has proved to be the SRB. In this paper we have described a sample of the projects, some of them are in production and some under development and we have described our experience in using the SRB both in production and in development. We have found the SRB to provide the basis for a data management system and have developed a number of tools to provide a production system. We have found the SRB developers and the SRB community to be extremely helpful in suggesting solutions to problems and have found the SRB developers more than willing to listen to new requirements and to implement them in a very timely manner.

References

[1] http://www.rcuk.ac.uk/escience

[2] http://www.e-science.clrc.ac.uk/web/groups/Data-Management/Data-Management

[3] http://www.sdsc.edu/srb

[4] http://www.eminerals.org

[5] http://www.eminerals.org/highlights/minigrid/index.html

[6] http://www.eminerals.org/highlights/dataportal/index.html

[7] http://www.e-science.clrc.ac.uk/web/projects/complexmaterials

[8] http://www.isis.rl.ac.uk/aboutIsis/index.htm

[9] http://www.isis.rl.ac.uk/dataanalysis/

[10] http://www.cclrc.ac.uk/Activity/Diamond

[11] http://www.ngs.ac.uk/

[12] http://www.integrativebiology.ox.ac.uk/

[13] http://www.integrativebiology.ox.ac.uk/softarch.html

[14] http://www.e-science.clrc.ac.uk/web/projects/dataportal

[15] R. Downing et al, "Some Tools for Supporting SRB Production Services", *SRB Users Meeting, Feb 2005.*

Storage Resource Broker Actors and Applications in Kepler

Nandita Mangal, Efrat Jaeger-Frank, Ilkay Altintas, Chien-Yi Hou, Lucas Gilbert and Arcot Rajasekar

San Diego Supercomputer Center, UCSD, 9500 Gilman Drive,
92093-0505 San Diego, California
{nmangal, efrat, altintas, chienyi, iktome, sekar}@sdsc.edu

Abstract

We present a set of SRB actors implemented in the Kepler scientific workflow management system to develop solutions to data grid issues arising in various scientific domains. Today, scientists in many scientific disciplines require applications for handling large data sets as well as performing complex analysis on this data. Kepler, together with the SRB framework, provides a scientific workflow environment for coordinating grid resources using storage brokering and file access technologies. Kepler and SRB together offer convenient and efficient data access, interaction and management in scientific workflows.

1 Introduction

Data Grid applications can easily evolve into large inflexible applications with varying levels of complexity and limitations from anywhere like the size of the data that can be handled, the proper maintenance of such data (such as adding/updating datasets) or simply the varying formats of data from different sources. As these data-intensive applications often require transfer of huge amounts of data, another big issue is the efficiency in data management among all shared parties and applications.

Storage Resource Broker (SRB) [2] is a storage management system designed for Data Grid environments. The SRB provides secure and optimized file transfer functionalities including transparent data replication; archiving, caching, and backup. Using logical name spaces, the SRB provides a uniform mechanism for seamlessly accessing data from various physical resources. Moreover, SRB offers bulk data ingestion, version control and Metadata query functionalities through a MetaData Catalog (MCAT).

Data Grid workflow is the automated process of ingesting, transferring and processing data in the Grid environment. Kepler [1, 3], a cross-project collaboration to develop a scientific workflow system for multiple disciplines, provides a workflow environment for scientists, in which they can design and execute Data Grid workflows and applications. Kepler builds on top of the mature Ptolemy II software developed at UC Berkley [4]. Ptolemy II is a Java-based system along with a set of APIs for heterogeneous hierarchical modeling. The focus of Ptolemy II is to build models based on the composition of existing components, which are called 'actors'. Actors are encapsulations of parameterized actions performed on input data to produce output data. Inputs and outputs are communicated through ports within the actors. Kepler implements SRB functionality by adding SRB actors, which perform SRB operations such as data access, metadata-based querying, update, file, storage, data transfer functions as well as server side processing of data using *Sproxy* commands. Within Kepler, SRB provides data access to diverse repositories using a single name space across all storage systems.

In the rest of the paper, we describe the Kepler SRB actors and models of their use in order to demonstrate Data Grid applications with Kepler-SRB.

2 SRB Actors in Kepler

Several SRB actor interfaces have been developed in Kepler to provide efficient storage functionality in Grid workflows. These actors are available in Kepler using the SRB JARGON java API [5]. Two specialized actors were created for connecting and disconnecting to and from a user's SRB space. Namely, the *SRBConnect actor* creates an authenticated socket connection using a connections pool. The connection is specified by a SRB host, port, username, password and domain. The *SRBDisconnect actor* releases the connection back to the shared pool. Within a scientific workflow all components/SRB actors accessing the same SRB space share the same connection socket by passing a connection token through actors input and output ports via channels. Other implemented SRB actors can be classified into the following categories: data access and transfer, server side processing of data, and an efficient search functionality using MCAT. Below we elaborate on each of these categories.

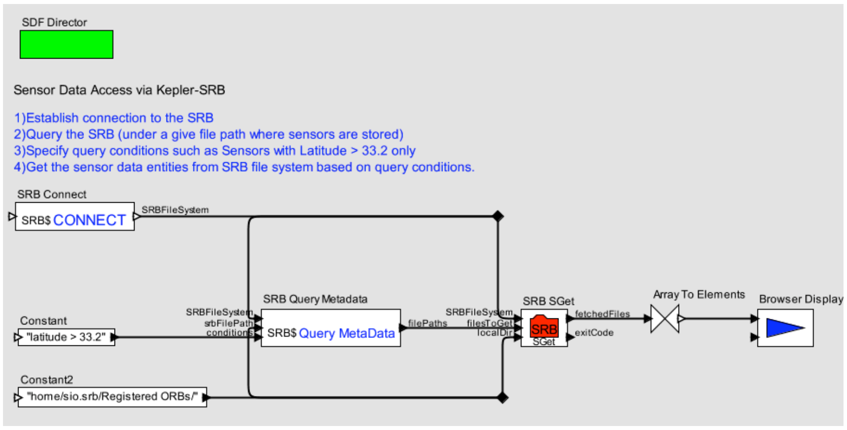
Data access and transfer actors. Kepler provides several components for data access and transfer, such as *StreamPut* and *SPut*, and *StreamGet* and *SGet* to enable streaming and parallel upload and download respectively. The streaming actors read and write files

from and to SRB as a sequence of bytes arrays, whereas the SPut and SGet use a parallel put and get approach provided by the JARGON API. A failure in parallel Put/Get automatically activates the streaming mode for data transfer. Another actor, called *SProxy* coordinates several common proxy commands in a single actor interface. Currently the following commands are supported: *list directory*, *copy*, *move*, *remove*, *replicate*, *create directory*, *remove directory* and *change mode*. More functions can be added based on necessity.
Server side processing actor. A special actor, called *SRBProxyCommand* was developed wrapping the SRB Spcommand. This actor enables executing any server side command, deployed on an SRB space, on SRB stored data.
Metadata actors. The last set of actors implement the SRB metadata functions. Through the workflow environment a user can add metadata to a file or a collection, get the metadata and also query for files satisfying a specific set of metadata conditions. These provide Kepler with an efficient Grid resources search functionality.

3 SRB Workflows

3.1 Accessing Large Distributed Datasets

With SRB native actors such as *SRBGet* and *SRBQuery Metadata* we are able to gain access to many forms of sensor data and information including ecological, oceanographic and hydrological data. The accessed sensor data is used in scientific applications to visualize and analyze environment conditions and phenomenon.


Sensor Data Access via Kepler-SRB

1)Establish connection to the SRB
2)Query the SRB (under a give file path where sensors are stored)
3)Specify query conditions such as Sensors with Latitude > 33.2 only
4)Get the sensor data entities from SRB file system based on query conditions.

Fig. 1. Accessing Sensor Data with a Logical View

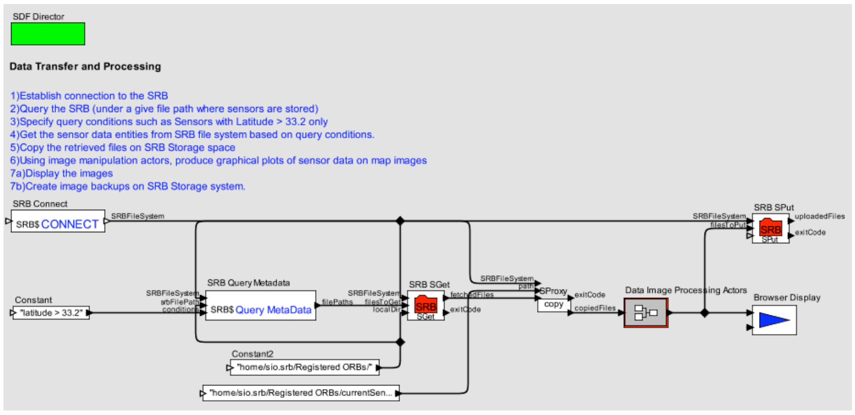
Such applications pose the need for quick real time and up to date sensor information access as well as efficiency in interaction with large datasets. An example workflow that is able to seamlessly access sensor data registered into a logical file name space, regardless of physical resource locations and data formats is given in Figure 1. The important operations that were used in building the workflow are as follows:

- **Logical Querying Using the SRBQuery Actor.** Metadata Querying is not constrained by various data packet forms or locations, however it can be based on the data's attributes or logical names.
- **Accessing Data from Different Sources Using the SRBGet.** With SRB actors in Kepler, we can access data from heterogeneous physical resources or repositories through a logical namespace. Properties can be associated with each file, such as format type, creation date, access controls, checksums, replica locations.
- **Getting Real Time Data via a Web Service.** In order to make sure the data being analyzed is real time and up to date, the above Kepler workflow can be executed as a web service and hence re-executed and updated with new incoming sensor data.

3.2 Data Transfer and Processing

With large datasets comes an equally challenging issue of processing the real time data as per the application needs as well as transferring and sharing the retrieved data and/or produced results to other third party applications and workflows.

The first issue is the overhead of retrieving large datasets. Common methods of saving datasets on the local hard drive and then processing or transferring data can cause great amounts of inefficiency and thereby affect application performance As a solution to this problem, SRB Proxy and SRB query actors in Kepler offer the capability of server side file processing as well as storage of files on the SRB storage space itself. Secondly with the datasets saved on the SRB storage space, the data can be easily transferred to other applications/workflows or even separate internal workflow components. Legacy applications can easily gain access to such data again via SRB Get and further process and analyze data.


Data Transfer and Processing

1)Establish connection to the SRB
2)Query the SRB (under a give file path where sensors are stored)
3)Specify query conditions such as Sensors with Latitude > 33.2 only
4)Get the sensor data entities from SRB file system based on query conditions.
5)Copy the retrieved files on SRB Storage space
6)Using image manipulation actors, produce graphical plots of sensor data on map images
7a)Display the images
7b)Create image backups on SRB Storage system.

Fig. 2. Data Transfer and Visualization among Workflow Components

The workflow in Figure 2 demonstrates data access and processing (in this case: visualization) using the actors and sub-workflows explained below:

- **Data Storage and Reuse using the SRBProxyCopy Actor.** Utilizing the functionality of SRB Proxy commands we can copy and store the retrieved sensor data after performing query operations thereby creating a backup archive of the specific sensor data retrieved during the given time.
- **Data Transfer.** While saving the sensor data on the SRB Storage space at all times, we transfer the data to another image manipulation and processing composite actor in Figure 2. This approach can also be utilized in transferring data to various other third party applications/workflows.
- **Sharing Application Results using the SRBPut Actor.** The graphical images produced as a result of data visualization in the above application can be further saved for future reuse / act as input sources to other applications.

3.3 Archiving and Management of Data Sets

Data Grid applications require the need of archiving data for long-term storage purposes as well as data reusability in other possible applications. *SRBPut* actor provides the capability to put back data sets on the SRB storage. On the other hand the functionality to manage such data archives can be used by *SRB Proxy* command actors available in Kepler. SRB Proxy commands include Scommands functionality such as data replication, copying, removing files/directories as well as changing file/directory permissions. Data from different resources can be further archived and maintained via the SRB Query metadata functionality.

UCSD-TV Digital Media Archival. "Conversations with History", hosted by Harry Kreisler, Executive Director of UC Berkeley's Institute of International Studies, is a long-running UCSD-TV series that has interviewed hundreds of prominent people, including Nobel Laureates, economists, historians, etc. In order to ensure that future generations can benefit from these intellectual legacies, these media files need to be archived for long term storage. The collection has three hundred one-hour programs, and each program, about 15GB in size, is preserved in three different formats: mov, mpeg, and rm. Our goal is to design a system to help UCSD-TV people to transfer and make replications on long term preservation resources easily without obvious impact on content production and dissemination.

We use a scientific workflow approach using Kepler with SRB as a middleware to provide a reliable parallel replication of the data to long-term replica machines such as HPSS or SANQFS. As access to these machines from a local machine may be very slow or unavailable, the following approach is used: the data is first internally staged from the local machine to an SRB zone at SDSC using the SPut actor, providing a parallel replication of the data. A checksum is performed on both ends to verify a successful upload. If the checksum fails the workflow persistently tries to re-upload the data. Once the data is successfully uploaded, it is being staged from the SRB zone to its long term storage replica and again a checksum is used to verify a successful replication. The benefits of using SRB as a middleware staging area are as follows:

- SRB provides a parallel upload which makes staging of the data much faster.
- Staging of data between SRB and their final destination is done to and from machines on the same network (SDSC).
- The long term storage may be unavailable or having problems, thus while the data is already replicated on SRB, a persistent staging of the data from SRB to its final destination can be done in the background.

4 Conclusions and Future Work

Kepler SRB actors have been designed with Data Grid computing in mind and hence provide efficient solutions in common scenarios in Grid environment applications today. Within a workflow system like Kepler, which enables efficient design and execution of workflows as common desktop applications, third party applications or web services, SRB is providing scientists with flexible as well as powerful Data Grid applications environment.

Currently Kepler supports a few *SRB Proxy* commands which help in data archive management and retrieval. However we are planning to add more SRB SCommands (e.g. *Serror* to display error and *SgetColl* to display information on SRB data objects) functionality to existing native Kepler-SRB actors. We also plan to design an experimental SRB domain in Kepler, which optimizes data transfers and provides connections at the system level so that the user doesn't have to concentrate on details of the interaction of the workflow system with SRB.

References

1. Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J., Zhao, Y.: Scientific Workflow Management and the Kepler System. Concurrency and Computation: Practice & Experience, Special Issue on Scientific Workflows, 2005.
2. Baru, C., Moore, R., Rajasekar, A., Wan, M.: The SDSC Storage Resource Broker, Proc. CASCON'98 Conference , Nov.30-Dec.3, 1998, Toronto, Canada. http://www.sdsc.edu/dice/Pubs/srb.ps.
3. Kepler Website: http://kepler-project.org
4. Ptolemy II Website: http://ptolemy.eecs.berkeley.edu/ptolemyII/
5. Java Api for Real Grids On Network (JARGON), http://www.sdsc.edu/srb/jargon/

Integration of HDF5 and SRB for Object-level Data Access*

Peter X. Cao
Univ. of Illinois, Urbana
xcao@ncsa.uiuc.edu

Mike Wan
Univ. of California, San Diego
mwan@sdsc.edu

Mike Folk
Univ. of Illinois, Urbana
mfolk@ncsa.uiuc.edu

Abstract

Fast partial access to objects from very large files in the SDSC Storage Resource Broker (SRB[5]) can be extremely challenging, even when those objects are small. The HDF-SRB model integrates the SRB and NCSA Hierarchical Data Format (HDF5[6]), to create an access mechanism within the SRB that is more efficient than current methods for accessing object-based file formats.

This model integrates two successful technologies, the SDSC SRB and the NCSA HDF, to create a new, more sophisticated distributed data service. The SRB serves as standard middleware to transfer data between the server and client. HDF5 provides interactive and efficient access to datasets or subsets of datasets in large files without bringing entire files into local machines. A new set of data structures and APIs have been implemented to support such object-level data access. A working prototype of the HDF5-SRB data system has been developed and tested.

1. Introduction

Storing massive data presents two big challenges: management of distributed data systems and efficient access to complex data content. The NCSA Hierarchical Data Format (HDF) and the SDSC Storage Resource Broker (SRB) have addressed the two issues. The SRB is client-server middleware (or grid data software) that provides a uniform interface and authorization mechanism to access heterogeneous data resources (UNIX FS, HPSS, UniTree, DBMS, etc.) distributed on multiple hosts and diverse platforms. The HDF is a file format and software library for storing all kinds of data, simple integers and floats or complex user-defined compound data types. The HDF employs a common data model with standard library APIs, providing efficient data storage and I/O access.

The HDF and the SRB offer valuable and complementary data management services, but they have not previously been integrated in an effective way. Earlier work had the SRB accessing HDF data either (a) by extracting entire HDF files, or (b) by extracting byte-streams through the SRB's POSIX interface. Approach (a) fails to take advantage of HDF's ability to offer interactive and efficient access to complex collections of objects. Approach (b) has been shown to be far too low-level to perform reasonably for some data extraction operations.

In discussions between NCSA and SDSC, it has been determined that a more effective approach is possible, one that uses modified HDF APIs on the server side to extract data from large files at the instruction of client-side HDF APIs and SRB as middleware to transfer data between the server and client. This approach would insert the HDF library and other object-level HDF-based libraries (such as HDF-EOS) between the SRB and a data storage source (such as a file system), making it possible to extract objects, rather than files or byte streams. Furthermore, these libraries typically offer query, subsetting, sub-sampling, and other object-level operations, so that these services might also be available.

2. Overview of SRB and HDF5

This section is a brief introduction of SRB and HDF5. For more information, you can visit the SRB and HDF5 websites at http://www.sdsc.edu/srb/ and http://hdf.ncsa.uiuc.edu/.

2.1. What is SRB

SRB is client-server middleware (or grid data software) that provides a uniform interface and authorization mechanism to access heterogeneous data resources (UNIX FS, HPSS, UniTree, DBMS, etc.) distributed on multiple hosts and multiple platforms. It is a distributed file system, a data grid management system, a digital library, and a semantic web.

* Sponsored by NLADR[1], NFS PACI[2] project in support of NCSA[3] and SDSC[4] Collaboration.

2.2. What is HDF5

HDF5 is a general-purpose library and file format for storing scientific data. At its lowest level, HDF5 is a physical file format for storing scientific data. At its highest level, HDF5 is a collection of utilities and applications for manipulating, viewing, and analyzing data in HDF5 files. Between these levels is the HDF5 software library that provides high-level APIs and a low-level data interface. HDF5 is a file format for storing all kinds of data and a library with standard APIs. It provides efficient data storage and I/O access and software and tools.

HDF5 can store two types of primary objects: datasets and groups. A dataset is essentially a multidimensional array of data elements, and a group is a structure for organizing objects in an HDF5 file. Using these basic objects, one can create and store almost any kind of scientific data structure, such as images, arrays of vectors, and structured and unstructured grids. You can also mix and match them in HDF5 files according to your needs.

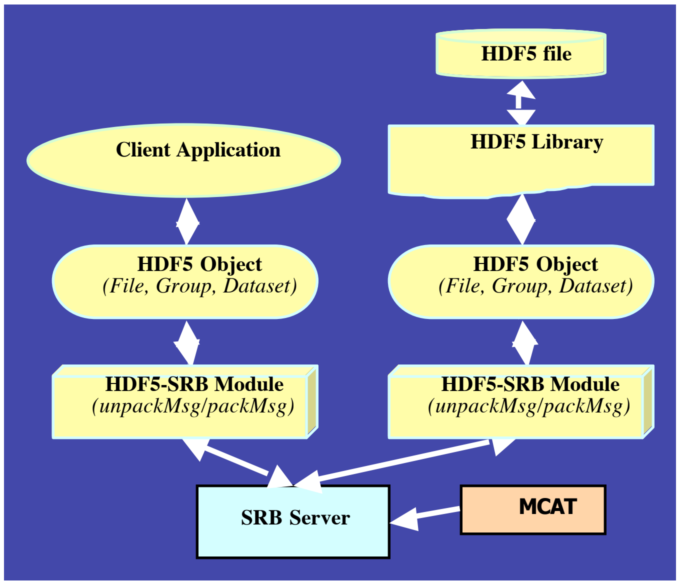

HDF5 file
Client Application
HDF5 Library
HDF5 Object (File, Group, Dataset)
HDF5 Object (File, Group, Dataset)
HDF5-SRB Module (unpackMsg/packMsg)
HDF5-SRB Module (unpackMsg/packMsg)
SRB Server
MCAT

Figure 1 The HDF-SRB Model.

3. The HDF-SRB model

We have designed a new mechanism, the HDF-SRB model to support object-level data access. The two basic requirements of the HDF-SRB model are simple and efficient. The HDF-SRB model has minimum changes the SRB code. It uses one set of objects for both server and client. It should have efficient data access by transferring only the required data (no redundant member object within an object) between client and server.

3.1. The HDF-SRB architecture

The HDF-SRB model consists of four basic components: the client (HDF client application or SRB client), the HDF-SRB module, SRB server, and the HDF library. Figure 1 illustrates the basic architecture of the HDF-SRB model.

Client applications are implemented using a set of APIs provided by SRB for sending requests and receiving responses to/from the SRB servers. The requests and responses are packed with HDF objects. The critical component is the HDF-SRB module, which connects the HDF clients to the HDF library on the server. The HDF-SRB module is responsible for packing and unpacking messages, or HDF objects, between the SRB and HDF components. The HDF library is installed with the SRB server for interactive access to HDF files on the server side.

3.2. HDF data objects

In the HDF-SRB model, data objects are passed between the client and server rather than the entire file. There are several advantages for object level access. First, passing objects is more efficient than passing an entire file especially for large files. For example, if we want to access a small subset of a gigabyte dataset, we just bring the selected data to the client instead of the whole file. Second, it is easy to pass complex requests such as sub-setting. Because messages passing between the client and server are packed in data objects, there is no need to specify the format of the messages; messages, simple or complex, are self-explained in the object. Thirdly, it is easy to maintain the source code and extend to support new objects and new functions. Adding a new function to the object will not require any change to the data model.

There are three basic HDF5 objects (C structures): H5File, H5Dataset and H5Group. H5File is used to hold metadata about the file and the root group of the file. The file structure can be constructed by following the links that flow from the root group. H5Dataset contains data values and information about the data, such as data type and data space. H5Attribute is similar to H5Dataset, but contains user-defined metadata for groups and datasets. H5Datatype contains information about the data type of the dataset, such as data type class, order and size. H5Dataspace contains sizes of the dimensions of dataset. It is also used to calculate the data size and pass the subsetting information.

3.3. Client side API

h5ObjRequest() is the client side API, which is responsible for sending a client request to the server and for processing the response from the server. Request and response messages are packed in the HDF object structures.

3.4. Server side API

h5ObjProcess() is the server side API, which processes client request and sends results to back the client. The server side API does not call the HDF5 library directly. It calls unpackMsg() to construct the data object passed from the client. The data object then takes the operation and calls the HDF5 library.

3.5. Pack/Unpack routine enhancements

The packMsg() and unpackMsg() routines exchange structured data between client/server. A data structure is packed into a single byte stream before sending cross the network. Byte stream received is unpacked back into the data structure based on its definition. The enhanced packMsg() and unpackMsg() routines handle complicated data structures – string, pointers, pointers to arrays, arrays of pointers, etc.

3.6. General proxy functions

In SRB, proxy functions allow the execution of certain functions on the servers to improve performance. Examples of the use of proxy functions include data subsetting and filtering type operations where they can most efficiently be carried out on the servers where data reside.

To make it easier to implement and handle object-level HDF5 requests which can be quite complex, a new and more general SRB proxy function framework has been added. This framework can also be used by other developers to implement their own proxy functions.

In this framework, a client calls a new client API - srbGenProxyFunct() to make proxy request. Inputs for the srbGenProxyFunct function include:

- int functType - the type of proxy function. e.g., HDF5_OPR_TYPE
- void *inputStruct - Pointer to input struct.
- FormatDef inputFormat - packing instruction for inputStruct.
- void **outputStruct - Pointer to output struct.
- FormatDef outputFormat - packing instruction for outputStruct.

Inputs for the proxy function are given in an "inputStruct" which is a pointer to an arbitrary data struct and a "inputFormat" which is a character string containing the instruction for serializing the "inputStruct" into a single byte stream before sending across the netwotk to the server.

For example, an "inputStruct" may contain an "int" and a pointer to a string:

```
struct foo {
        int myIndex;
        char *myName;
};
```

The serializing instruction is a string containing "int myIndex; str *myName;" which instructs the serializing routine to treat the first member of the struct as an integer and the second member as a pointer to a string.

Similarly, the "outputStruct" and "outputFormat" specify the output and packing instruction for the output of the proxy function.

On the server side, the genProxyFuncEntries[] table defined in genProxyFunct.h is a switch table used by the server to determine the handling function for each proxy function type. Currently, the genProxyFuncEntries[] is defined as follows:

```
genProxyFunc_t genProxyFuncEntries[] = {
   {HDF5_OPR_TYPE, (func_ptr) h5ObjProcess},
};
```

The table contains only one entry, the HDF5 type (HDF5_OPR_TYPE) proxy function. The h5ObjProcess() function will be called to handle the HDF5_OPR_TYPE request. To implement a new type of proxy function, one needs to simply add one more entry to the genProxyFuncEntries[] table and a function to handle this type of request on the server.

4. Client application: the HDFView

The HDFView is a visual tool for browsing and editing NCSA HDF4 and HDF5 files. Using HDFView, you can

- view a file hierarchy in a tree structure
- create new file, add or delete groups and datasets
- view and modify the content of a dataset
- add, delete and modify attributes
- replace I/O and GUI components such as table view, image view and metadata view

For more details on HDFView, visit the NCSA HDFView webpage at http://hdf.ncsa.uiuc.edu/hdf-java-html/hdfview/index.html.

Supporting HDF-SRB in HDFView requires implementing HDF-SRB Java Native Interface(JNI) and adding new GUI components and data object

4.1. The HDF-SRB JNI

The HDF-SRB Java Native Interface (JNI) consists of an Java class and dynamically linked native library. The Java class declares static native methods, and the library contains C functions which implement the native methods. The C functions call the standard HDF-SRB client module.

The HDF-SRB JNI class contains only one native interface, h5ObjRequest(). h5ObjRequest () does two things: load the dynamic library and pass client requests to the C function.

```
public synchronized static native int h5ObjRequest
(String srb_info[], Object obj, int obj_type) throws
Exception;
```

The dynamic library (C implementation of the native interface) wraps the SRB client and converts data object between C and Java. When client calls the Java interface h5ObjRequest(), the dynamic library does the following tasks:

- Make connection to the SRB server
- Decode the Java object and construct C structure
- Send requests to the server in the form of C structure
- Encode server result in to Java object

4.2. The Java HDF-SRB objects

HDFView is implemented based on the HDF object package, a Java package which implements HDF4 and HDF5 data objects in an object-oriented form. The HDF Java object package provides a common standard Java API to access both HDF4 and HDF5 files. For more information on the HDF Object Package, visit http://hdf.ncsa.uiuc.edu/hdf-java-html/hdf-object/index.html.

To support HDF-SRB data objects, we have implemented the following Java package, ncsa.hdf.srb.obj, which contains:

- H5SrbFile extends FileFormat
- H5SrbGroup extends Group
- H5SrbScalarDS extends ScalarDS
- H5SrbCompoundDS extends CompoundDS
- H5SrbDatatype extends Datatype

These objects implement methods to deal with the client requests and data from the server. The native call, h5ObjReques(), passes the information through the objects. For example, the following code is how the we read data from remote file using H5SrbScalarDS::read().

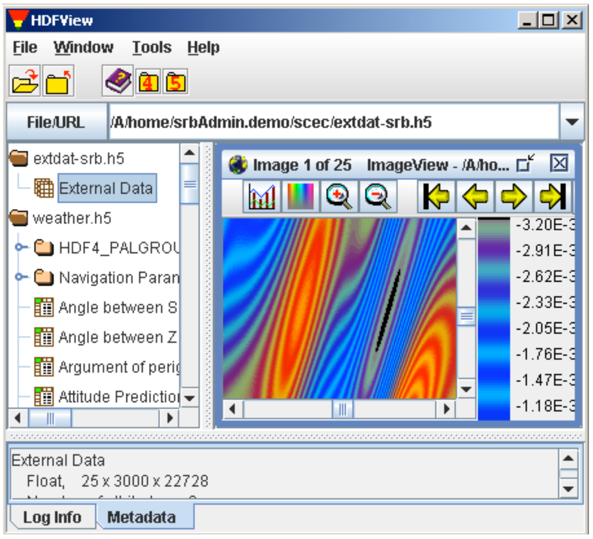
```
public Object read() throws Exception
{
    String srbInfo[] =
        ((H5SrbFile)getFileFormat()).getSrbInfo();
    if ( srbInfo == null  || srbInfo.length<5)
        return null;
    opID = H5DATASET_OP_READ;
    H5SRB.h5ObjRequest (srbInfo, this,
        H5SRB.H5OBJECT_DATASET);
    return data;
}
```

4.3. The GUI components

Since HDFView is built on modular fashion, the GUI components are transparent to data access. There is not much change to the GUI components. We added SRBFileDialog class to the GUI. SRBFileDialog class is used to make server connection by using the Java API for Real Grids On Networks (JARGON). JARGON is a pure java API for developing programs with a data grid interface. The API currently handles file I/O for local and SRB file systems, as well as querying and modify SRB metadata. For more information on JARGON, read http://www.sdsc.edu/srb/jargon/index.html.

Figure 2 shows two examples of accessing HDF5 files on the SRB server in HDFView. The first file, extdat-srb.h5, contains one dataset of size about seven gigabytes (25*3000*22728*4). With SRB support, we can have instant access to subset of the seven gigabyte dataset. Without SRB support, it would take hours to transfer the whole file to local machine.

The second example, weather.h5, shows how we can have instant access to the file structure. With SRB support, we can browse through the file structure without bringing the content of the file to local machine.



External Data
Float, 25 x 3000 x 22728

Figure 2 The SRB File Access in HDFView

4.4. Conclusions

The HDF-SRB model integrates two successful technologies, the SDSC SRB and the NCSA HDF, to create a new, object-lvel distributed data service. The SRB serves as standard middleware to transfer data between the server and client. HDF5 provides interactive and efficient access to datasets or subsets of datasets in large files without bringing entire files into local machines. A new set of data structures and APIs have been implemented to the SRB support such object-level data access.

A working prototype of the HDF5-SRB data system has been developed and tested. The HDF-SRB has been proved to be very efficient in large files. We have implemented SRB support in HDFView. Using HDFView, we can have instant access to file structures and fast access to subset of large dataset. Without HDF-SRB support, it might take hours to bring the file to a local machine.

This project has been a very successful team effort between SDSC and NCSA. Both SRB and HDF5 are very complex and the implementation of such a server/client system requires a full understanding of the two technologies. The SDSC SRB and NCSA HDF teams have worked together on all parts of the project, including designing, coding and testing.

References

[1] http://www.nladr.org
[2] http://www.npaci.edu/
[3] http://www.ncsa.uiuc.edu/
[4] http://www.sdsc.edu/
[5] http://www.sdsc.edu/srb/
[6] http://hdf.ncsa.uiuc.edu/

SRB Interfaces to the Antelope Environmental Monitoring System: The Antelope ORBserver, Datascope Database System, and Deployable ROADNet Point-Of-Presence

Kent G. Lindquist
Lindquist Consulting, Inc.
kent@lindquistconsulting.com

Arcot Rajasekar
Univ. of California, San Diego
sekar@sdsc.edu

Frank L. Vernon
Univ. of California, San Diego
flvernon@ucsd.edu

John Orcutt
Univ. of California, San Diego
jorcutt@ucsd.edu

Abstract

As part of the ROADNet project, we have developed two new interfaces for the Storage Resource Broker, one to the Antelope ORBserver and one to the Datascope relational database system (Antelope and Datascope are products of Boulder Real-Time Technologies, Inc.). The ORBserver is a real-time, event-driven node that can be used for highly reliable transport of packetized, labeled streams of environmental monitoring data. Network transparency allows the ORBserver to mediate near-real-time distributed processing of such packet streams. The associated Datascope database system allows near-real-time archival of such data streams into a fully generic relational database system. The two new Storage Resource Broker (SRB) interfaces allow ORBserver and Datascope resources to be registered, then accessed through SRB mechanisms. Recently we have begun development of a deployable Sun-Solaris machine and software stack called the ROADNet Point-Of-Presence (RPOP), which integrates Antelope, the Storage Resource Broker, and several other cyberinfrastructure components into a production-quality standalone server. This will give RPOP simultaneous, interconnected presence in several data- and sensor-grid paradigms. The RPOP can be seen as the fundamental, intelligent "atom" in a distributed real-time sensor-, data-, and processing-grid. The multi-signal-domain acquisition capability provided by the rest of the ROADNet and Antelope tools will optionally allow RPOP to be a leaf node in the acquisition system. RPOP may also be used as a data concentrator or data-access point at participating laboratory sites, and the processing capabilities inherent with Antelope and ROADNet tools will allow it to be used in distributed, near-real-time processing of the acquired data streams.

1. Introduction

Real-time, gridded sensor networks such as ROADNet [1] collect large volumes of multidisciplinary sensor data that must be buffered (held in accessible storage) for immediate analysis and redistribution, as well as archived for future re-examination for long-term trends and comparison of recent to historic events. In order to be useful to the wide range of users, data contributors, science teams and monitoring operations, these diverse datasets need to be accessible in some kind of centralized manner, though with component data sets often distributed amongst different research groups. Data access methods must be straightforward for end-users and for authors of analysis, processing, and display operations. Furthermore, any such system must be able to handle the diverse system and domain metadata, which may vary widely from subdiscipline to subdiscipline, often blurring the boundary between data and metadata. Finally, although the resultant system may be used for research-based prototypes and short-term (low investment) monitoring experiments, the need for constant availability and for support of large-scale, mission-critical scientific and monitoring operations requires a robust cyberinfrastructure characteristic of a hardened production system.

1.1. ROADNet

The ROADNet project is a testbed for real-time integration of multidisciplinary sensor networks, together

* This research was sponsored by a funding agency. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the Government, or any person or agency connected with them.

with development and integration of the cyberinfrastructure technologies needed to process and maintain them. The sensors collected include a wide range from seismic [2], infrasound, Global Positioning System [3], meteorological, high-frequency-radar ocean surface-current, remote cameras [4], and numerous others. The cyberinfrastructure components include the Antelope Environmental Monitoring System, custom additions to the above, the Storage Resource Broker (all described below), and the Kepler scientific workflow system [5].

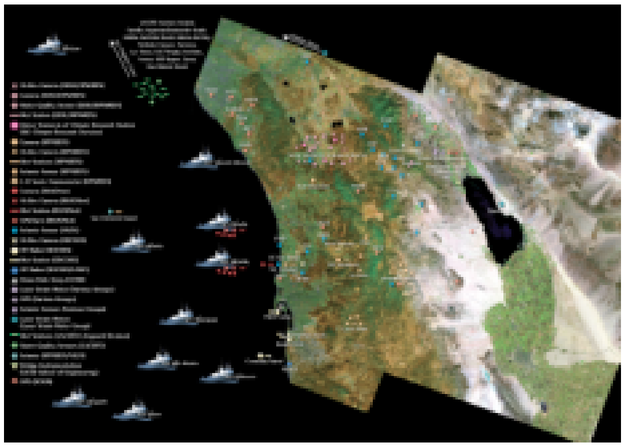


Figure 1. ROADNet sensor network.

A basic overview of the ROADNet real-time system architecture is shown in Figure 2. The left half shows Antelope-based components involved in data acquisition, processing, and initial archiving; the right half shows Storage Resource Broker-based deep archival, replication and redistribution.

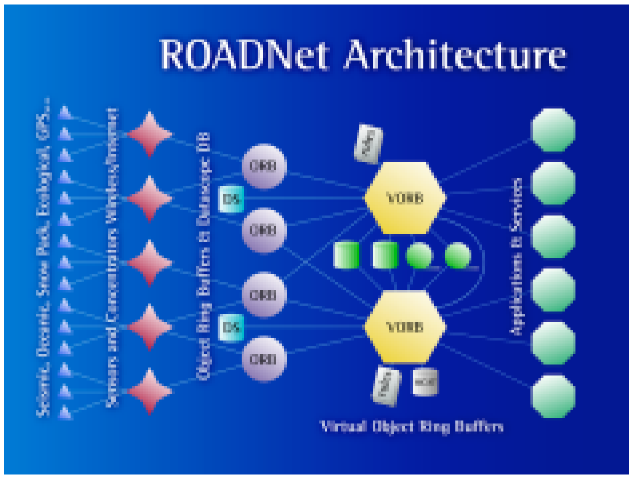

ROADNet Architecture
Seismic, Oceanic, Snow Pack, Ecological, GPS...
Sensors and Concentrators Wireless/Internet
Object Ring Buffers & Datascope DB
ORB
DS
ORB
ORB
DS
ORB
VORB
VORB
Applications & Services
Virtual Object Ring Buffers

Figure 2. ROADNet architecture

2. Drivers for Antelope and Datascope

The Antelope Environmental Monitoring System, a product of Boulder Real-Time Technologies, Inc. [6], forms the basis for much of ROADNet's real-time dataflow, archiving and processing. Antelope consists of several hundred Unix command-line utilities, as well as Application Programmer Interfaces in multiple languages, that generally divide into two classes: utilities for handling streams of real-time packets of environmental monitoring data (opaque binary objects labeled by stream identifier and timestamp); and a fully general Relational Database Management System called Datascope, which is optimized for real-time processing. An overview of the major classes of Antelope system components is shown in Figure 3.

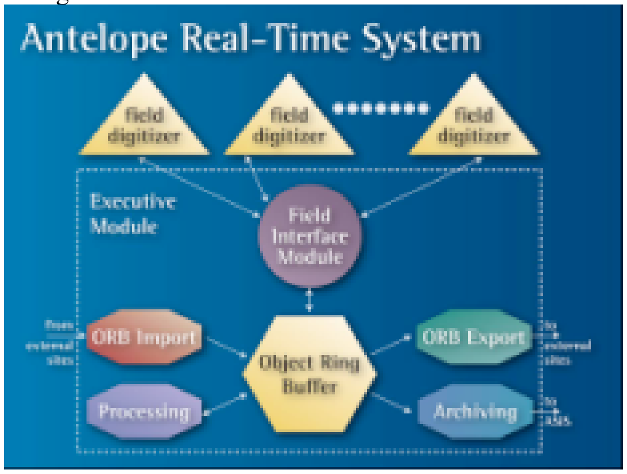
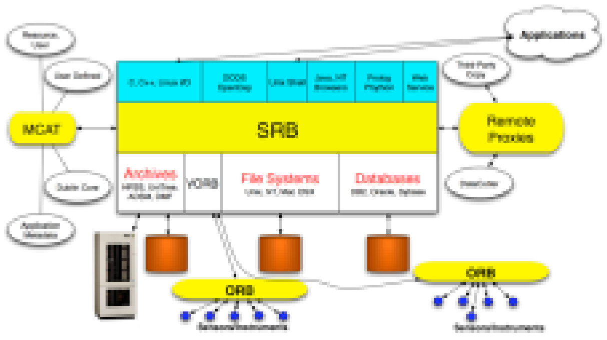


Figure 3. Antelope Components

The Storage Resource Broker supports several different classes of drivers including filesystems, databases, tape-stores, and more. The new Antelope drivers are implemented as SRB Miscellaneous drivers, as shown in Figure 4.



2.1. Datascope interface

The Storage Resource Broker drivers support several basic types of operations, including object Create, Read, Write, Open, Close and several others. The capability to add generic procedures to miscellaneous drivers through the "Proc" driver function is used to implement most of the Datascope interface.

The Datascope interface supports many of the standard calls to the Datascope database system, including database opening and closing, creation of views, joins, and groups, row-based queries, bulk ingestion/export mechanisms, and translation to XML.

2.2. Orbserver interface

The orbserver interface supports basic connectivity to Antelope real-time packet streams. An Antelope orbserver connection is best seen as a serial packet stream which can be continuously reaped, or rewound and probed as necessary to find individual packets. By far the most efficient use is to continuously read incoming packets in an event-driven loop. Several common operations are possible with a SRB-brokered orb connection: in addition to opening and closing orb connections, these include selecting or rejecting packet streams based on regular expressions, reaping packets and unstuffing their contents, and querying the orbserver for metadata on available packet-streams and connected clients.

3 .The ROADNet Point-Of-Presence (RPOP)

The cyberinfrastructure technologies involved in the ROADNet project form a fairly complex stack of data acquisition utilities, processing, archiving, and management tools, as well as access mechanisms for a variety of sensor- and data-grid paradigms. In a research-lab setting the complexities of configuring and managing such a conglomeration are usually manageable due to the presence of specialized research and support staff. However, because the grid-based architecture pre-supposes the ability to deploy intelligent field nodes, data concentrators and distributed access points, it is valuable to tame the software deployment and configuration issues by creating an operationally robust, easily deployable and easily updated and maintained platform. Towards this end we are building "ROADNet Point-Of-Presence" (RPOP) machines which encapsulate all necessary Antelope, ROADNet, and Storage-Resource-Broker components for the ROADNet real-time sensor and data grid. The RPOP developments are currently in prototype stage, with several test units deployed. An image of the Sun Fire servers currently in use for the RPOP is shown in Figure 4.

Figure 4. Sun Fire machine for RPOP

4. Conclusions

The SRB Antelope interfaces and the ROADNet Point-Of-Presence machines promise to be a significant step forward in the connection of real-time sensor networks to gridded environments. Please direct any questions to Kent G. Lindquist, kent@lindquistconsulting.com.

5. Acknowledgments

This work and the ROADNet project are sponsored by grants from the National Science Foundation (OCE-0121726), the Office of Naval Research (N00014-98-1-0772), and with matching funds from the California Institute of Telecommunications and Information Technology, Scripps Institution of Oceanography, San Diego State University, and the Cecil H. and Ida M. Green Institute of Geophysics and Planetary Physics. The ROADNet sensor network figure (Figure 1) is courtesy of Todd Hansen. The image of the Sun Fire server in Figure 4 is courtesy of Sun Microsystems, Inc., http://www.sun.com.

References

[1] T.S. Hansen, S. Foley, K. Lindquist, N. Cotofana, L. Ding, L. Hazard, M. Otero, E. Terrill, J. Orcutt, "ROADNet: A network of SensorNets", *Submitted to the Information Processing in Sensor Networks 2006 meeting*.

[2] K.G. Lindquist, F.L. Vernon, T.S. Hansen, A. Rajasekar, B. Ludaescher, J. Orcutt, J. Berger, H.W. Braun, Y. Bock, "Generalizing Seismic Processing Systems to Diverse Signal Domains." In *Abstracts from the Fourteenth Annual IRIS Workshop, The IRIS Consortium,* June 12-16 (2002), 141 pp.

[3] K.G. Lindquist, Y. Bock, F. Vernon, D. Honcik, J. Eakins. "Preliminary Integration of Real-time GPS and Seismic Data." In *Abstracts from the June, 2005*

IRIS/UNAVCO Meeting, Skamania, Washington (2005).

[4] K.G. Lindquist, T.S. Hansen, R.L. Newman, F.L. Vernon, A. Nayak, S. Foley, T. Fricke, J. Orcutt, A. Rajasekar. "Digital Image Support in the ROADNet Real-time Monitoring Platform." *Eos Trans. AGU* **85**, 47, *Fall Meet. Suppl.*, Abstract SF41A-0755 (2004).

[5] T.T. Fricke, B. Ludaescher, I. Altintas, K.G. Lindquist, T.S. Hansen, A. Rajasekar, F.L. Vernon, J. Orcutt. "Integration of Kepler with ROADNet: Visual Dataflow Design with Real-time Geophysical Data." *Eos Trans. AGU*, **85**(47), *Fall Meet. Suppl.*, Abstract SF41A-0762.

[6] http://www.brtt.com

Data Grid Services Based on SRB for National Digital Archives Program in Taiwan

Wei-Long Ueng
Grid Computing Centre,
Academia Sinica, Taiwan
wlueng@sinica.edu.tw

Hui-Min Lin
Grid Computing Centre,
Academia Sinica, Taiwan
huimin@gate.sinica.edu.tw

Eric Yan
Grid Computing Centre,
Academia Sinica, Taiwan
eric@sinica.edu.tw

Abstract

Objectives of the Data Grid for the National Digital Archive Project (NDAP) are to provide Grid services for long-term preservation and unified data access. These services will be built upon the e-Science infrastructure of Taiwan, by integrating the data management components of the underlying middleware. The services also link the digital archive management tools and applications to take advantage of the Grid infrastructure.

The Storage Resource Broker (SRB) Middleware enables users to create, manage, and collaborate with flexible, unified "virtual data collections" that can be stored on heterogeneous data resources distributed across a network. In early 2004, we deployed SRB nodes to various institutes in Taiwan to handle the massive storage, content management, and long-term preservation requirements of the National Digital Archive Project. As of January 2006, more than 30 TB and 1.4 million files have been archived in the distributed mass storage environment.

In this paper, the deployment of SRB in building a collaborative environment for the National Digital Archives Project are described in detail. The software tools developed to facilitate system management tasks are also introduced. In addition, many applications based on the same SRB-based Data Grid services developed in Taiwan are presented as well. For each application, the essential data virtualization services provided by SRB to manage the highly distributed and heterogeneous data sources are characterized.

Keywords
Data Grid, SRB, National Digital Archives Program, Applications

1. Introduction

Digital archives/libraries are widely recognized as a crucial component of a global information infrastructure for the new century. Research and development projects in many parts of the world are concerned about using advanced information technologies for managing and manipulating digital information, ranging from data storage, preservation, indexing, searching, presentation, and dissemination capabilities to organizing and sharing of information over networks.

Digital Archives demand reliable storage systems for persistent digital objects, well-organized information structure for effective content management, efficient and accurate information retrieval mechanisms, and flexible services for variant users needs. Hundreds of Petabytes of digital information have been created and dispersed all over the internet since computers were used for information processing, and the amount still grows at the rate of tens of Petabyte per year. Grid technology enlightened a possible solution for processing diversified heterogeneous Petabyte scale digital archives. Metadata-based information representation makes specific and relative information retrieval more accurate, makes information resources interoperable, and paves the way for formal knowledge discovery. By taking advantage of advancing IT, semantic level information indexing, categorizing, analyzing, tracking, retrieving and correlating could be implemented. Data Grid aims to set up a computational and data-intensive grid of resources for data analysis. It requires coordinated resource sharing, collaborative processing and analyzing on huge amounts of data produced and stored by many institutions.

In Taiwan, a National Digital Archive Project (NDAP) was initiated in 2002 with its pilot phase started in 2001. Not only delicate and gracious Chinese cultural assets will be accessible thru the Internet, but also a new paradigm of academic research based on digital and integrated information resources will be devised and implemented. In this paper, ideas for utilizing Data Grid infrastructure for NDAP will be depicted and discussed.

2. Objectives

The major goal of the program is to conduct Grid-related R&D and integration tasks to help digitize and network the collections and resources of different institutes in NDAP. To achieve this goal, related Grid framework, application and development are conceived and designed in a comprehensive way, taking both the standardization and commonality of development environment and solutions into account. The digitized resources of institutes will be well preserved and re-used systematically. Different disciplinary institutes will be able to build and develop their resources on the same environment. Ultimately, the institute can share their digitized data.

The workflow for digitizing and networking the NDAP collections are summarized as follows.

Accumulation

Systematically accumulate media acquired during the process of collection, research and educational exhibitions. These media include photographs, recordings, films, slides, literature and manuscripts.

Material Organization and Description

Classify and organize different media and materials of all collection resources, conduct analyses on metadata, and provide annotation and interpretation of each kind of medium and material.

Digitization

The purpose of digitization is to use information technology to convert collected media into forms that can be stored, manipulated and edited.

Editing

Combine the digitized media and the description and interpretation by professionals with multimedia technology through graphical input interface; edit and assemble these materials into different information services and products to meet users' needs at all levels.

Accessing

Design multi-access methods, including browsing and query, for users to obtain information they need quickly.

Dissemination

Provide information services and products through computers and communication systems, such as Internet or CD-titles.

To achieve the aforementioned goals, an information structure that centers on integrated distributed resources should be established. Other extensive IT support and integration efforts are as follows.

* Standardization of digitization process and establishment of resource management system.

* Distributed data resource and editing organization interface

* Establishment of integrated associated index structure

* Extensible structure of distributed data resources

* Structure of integrated data storage system

* User-friendly design of browsing guide and query functions.

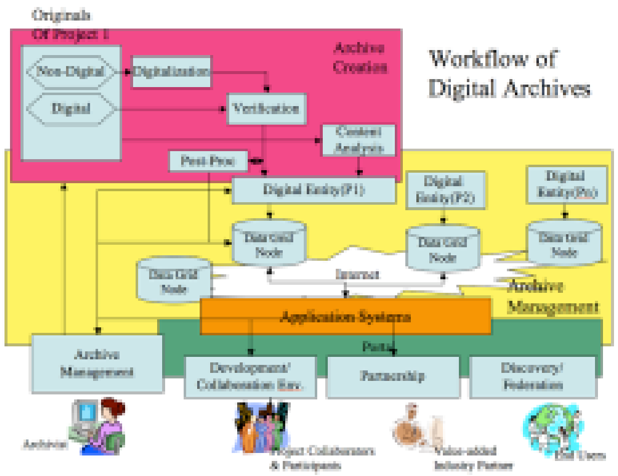


Fig 1 Workflow of Digital Archives

3. System Layout and Architecture

The Storage Resource Broker (SRB) Middleware enables scientists to create, manage, and collaborate with flexible, unified "virtual data collections" that may be stored on heterogeneous data resources distributed across a network. The SRB system in Academia Sinica, Taiwan is used for the long-term preservation of the digital contents produced by the digital archives projects, which are part of the National Digital Archive Projects in Taiwan. The system was deployed by the Academia Sinica Grid Computing Centre (ASGC) in early 2004, which was constituted from 7 sites in different institutes. Each site has an 8TB STK disk array, linked by a dedicated fibre campus network, and provides 68 TB capacity in total (before RAID-5). In early 2006, it will expand to 120 TB. A tape library server also has been incorporated into the SRB system. It provides 500 TB capacity in total. The system layout and architecture are illustrated in Figures 2 & 3.

ASGC is working on a new generation of Grid-based research infrastructure in Academia Sinica and in Taiwan, by using gLite and OSG as the Grid middleware. The Data Grid is a major part of this infrastructure, and the SRB is the first and the largest (in terms of the data volume) Data Grid in our academy right now. As of January 2006, more than 30 TB and 1.4 million files have been archived in the distributed mass storage environment. All files are also preserved in two copies on different sites.

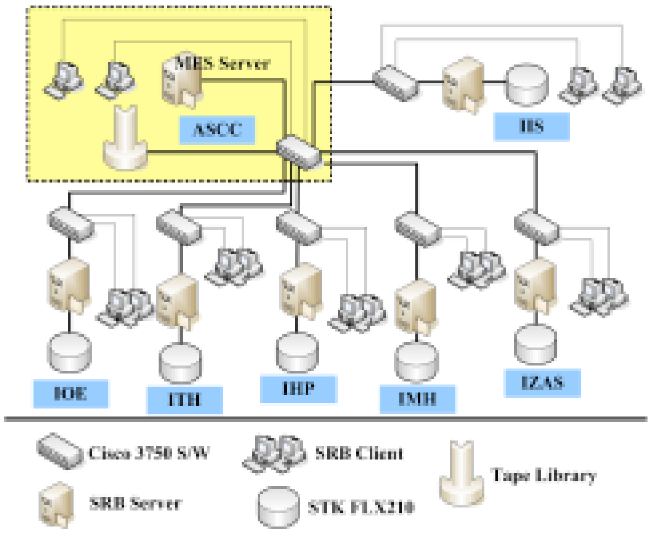


Fig 2 System Layout

.

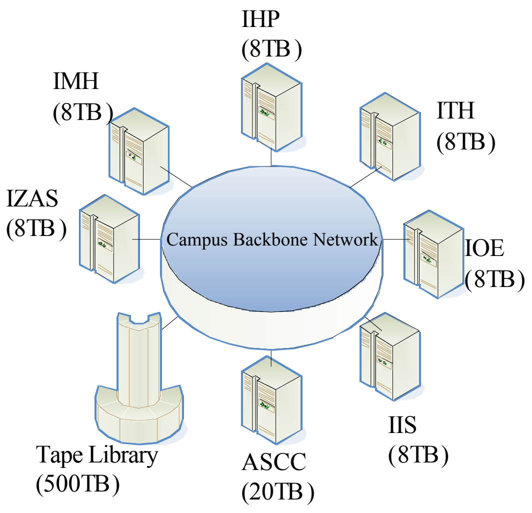


Fig 3 System Architecture

4. Development

From direct operations experience, we developed the SRB client and administration related software tools to meet the requirements of the end users and system manager. The software tools, including System Report Generator (SRG), Sync Package and PHPMySRB will be illustrated in this section.

4.1. System Monitoring (System Report Generator)

We faced a resource management challenge as the data grew rapidly after constructing the storage network based on SRB among the institutes in Academia Sinica. Therefore, we developed a software tool called "System Report Generator", which aims to facilitate resource management tasks for the system manager. The system manager can use the SRG to monitor the resource status of the huge backend disk pool. In other words, SRG can also be regarded as a monitoring component at the collective layer of the grid architecture, which is responsible for collecting the resource and user usage information on SRB. The use case for SRG is illustrated in Figure 4.

SRG is a set of programs based on the well known SRB command line interface "Scommands", written in PERL and bash scripts. The main feature is that it allows the users to run the command-line program with some options to produce the desired results, whose output format may be defined in the template. Another important feature is the load resource management design. The manager can set the threshold value for each resource usage to monitor the SRB resource more efficiently. They can receive a warning notification when the resource is running out of space.

With regard to other SRG applications, SRG can be easily integrated into other visualization programs. For instance, the advanced user may parse the historical results according to the format previously defined in the template and then plot charts and show the aggregated results as the system report. In addition, they may make use of the template feature to output the visualization codes directly.

SRG is a kind of general-purpose SRB related software tool, which has been successfully used in the national digital archive project. It will be applied to other SRB related data grid projects, such as the atmosphere databank project, as the resource monitoring tool.

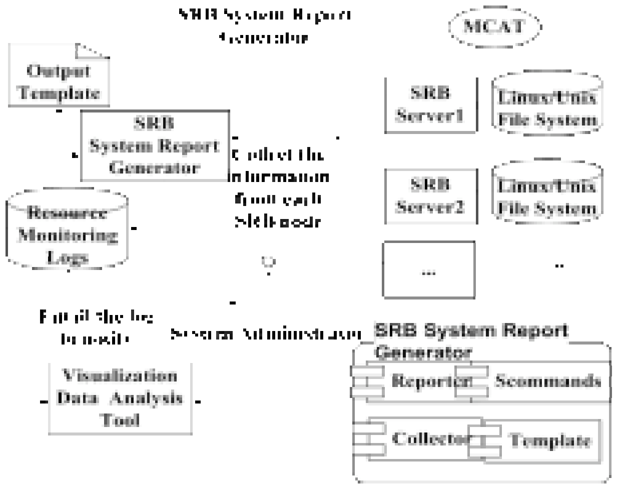


Fig 4 System Report Generator

Based on the System Report Generator, we also created a visualization web-base monitoring interface system. It can show all system and resource usage, watermark, condition, users' usage, data/files statistics and statistics charts. The SRG lets system managers clear all system conditions at a glance.

The interface is shown in Figure 5. We also integrate with Google Earth software to show the geographic distribution and usage of resources , see Figure 6.

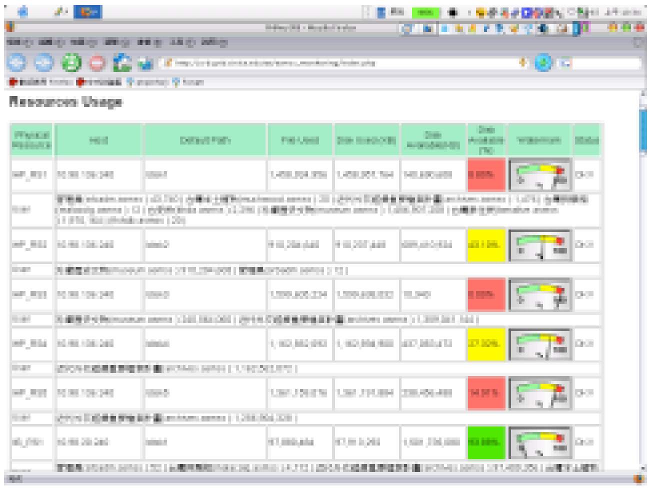

Resources Usage

Fig 5 Web-Base System Monitor Interface

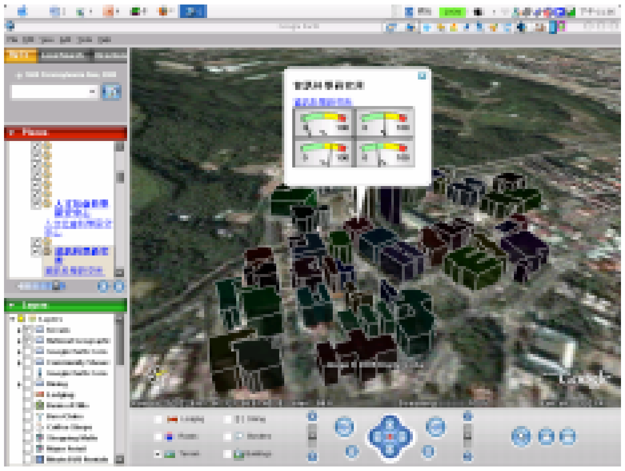


Fig 6 Resources Geographic Distribute on Google Earth

4.2. Sync Package

The Sync Package is a client utility program based on Scommands. It is designed to help the end users simplify their backup procedures and to enable workflow tracking. The entry-level users with limited computer knowledge and skills can easily use this utility to ingest their data in the backend SRB system as well. The Sync Package is illustrated in Figure 7.

For the design of the Sync package, we packed the powerful Srsync program in a more customized way with logging and booking capability. The Srsync program synchronizes the data between a local copy (local file system) and the copy stored in SRB or between two SRB copies. We deploy the Sync package to keep track of the sync process from the event logs automatically generated by Srsync program. Upon completion of the data ingestion, the sync report can be configured to be sent to both the users and managers. On one hand, the users will be notified of the data transfer results to be assured that the data transfer job is done. On the other hand, the system manager can make use of the logs to diagnose data transfer problems in the first place or do post-mortem analysis about the system reliability.

We currently release two versions of the Sync package for Windows and Linux users. The users are advised to utilize it along with the other system utilities, such as crontab and windows task scheduler, to execute the sync job seamlessly and on a regular basis. Both versions have been adopted and used by most institute users in Academia Sinica.

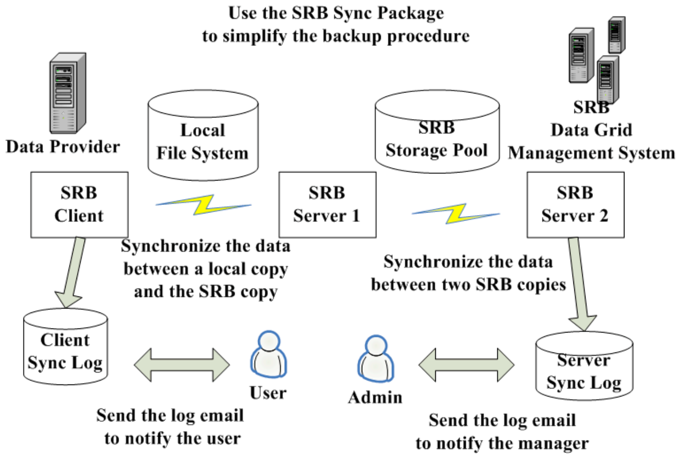
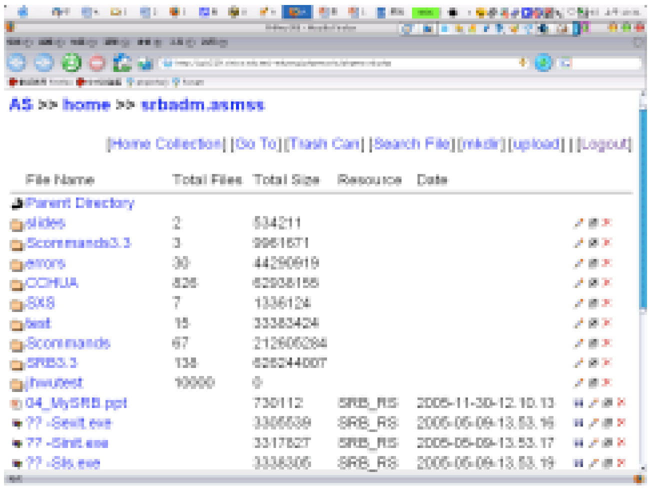

Use the SRB Sync Package
to simplify the backup procedure
Data Provider
Local File System
SRB Storage Pool
SRB Data Grid Management System
SRB Client
SRB Server 1
SRB Server 2
Synchronize the data between a local copy and the SRB copy
Synchronize the data between two SRB copies
Client Sync Log
User
Admin
Server Sync Log
Send the log email to notify the user
Send the log email to notify the manager

Fig 7 The SRB Sync Package

4.3. Web user interface

In contrast to MySRB, we developed a more friendly web user interface call PHPMySRB, written using PHP scripts. Users can use the PHPMySRB Web interface on Unix, Windows, Linux, and Mac platforms to define and manipulate collections of files. They also can create personal data collections and share, search, replicate, and more, all from a Web interface without installing any client software. The PHPMySRB is also a modularized application; developers can use the APIs of PHPMySRB in their own web applications. PHPMySRB is being tested within ASGC and provided to NDAP users for use.



AS >> home >> srbadm.asmss

[Home Collection] [Go To] [Trash Can] [Search File] [mkdir] [upload] | [Logout]

File Name	Total Files	Total Size	Resource	Date
Parent Directory				
slides	2	534211		
Scommands3.3	3	9961671		
errors	30	44290919		
CCHUA	826	62938155		
SXS	7	1336124		
test	15	33383424		
Scommands	67	212606284		
SRB3.3	138	628244007		
jhwutest	10000	0		
04_MySRB.ppt		730112	SRB_RS	2005-11-30-12.10.13
?? -Sexit.exe		3306539	SRB_RS	2005-06-09-13.53.16
?? -Sinit.exe		3317627	SRB_RS	2005-06-09-13.53.17
?? -Sls.exe		3338306	SRB_RS	2005-06-09-13.53.19

Fig 8 PHPMySRB Web User Interface

5. Applications

Based on the use of SRB Data Grid services for NDAP, we developed a data I/O interface to integrate applications such as Atmosphere Science and Geographic Information System.

5.1 Atmosphere Science Applications integration with SRB

Since 1994 the Live Access Server (LAS) has been providing visualization and subsetting of multi-dimensional scientific data for Web users. The LAS is a distributed "data fusion" system designed to support collaborative research. Users can co-plot and difference (with regridding as required) the comparative data sets. Binary access to remote data sets is provided transparently by the Distributed Ocean Data System (DODS). The DODS provides transparent access to remote data for existing applications. Thus the use of DODS makes it possible for Web servers such as LAS to provide access to reference data sets without incurring any of the costs associated with managing the data set. Sharing access to reference data sets is reduced to exchanging the small XML files that describe their metadata.

In the Atmosphere Databank Project, we developed a DODS-SRB file I/O interface which integrates some Atmosphere applications such as LAS and GrADS. The system architecture is illustrated in Fig 9.

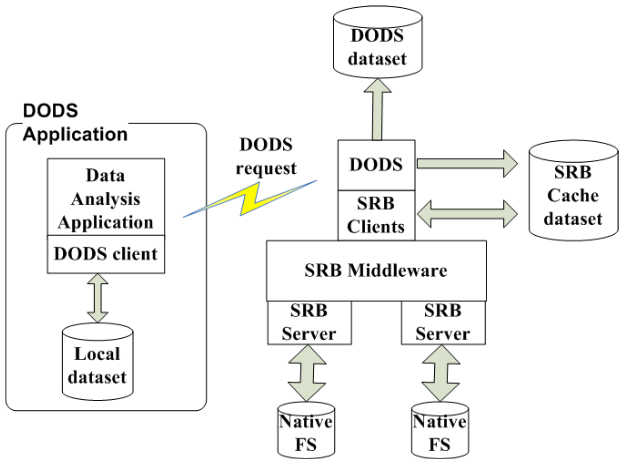


Fig 9 Atmosphere Databank Architecture

LAS supports collaborative research activities by providing 1) common access to reference data sets (without undue duplication of effort at each site); 2) shared, mutual access (visualization and subsetting) to distributed data sets; and 3) the ability to inter-compare distributed data holdings. Within LAS the use of a remote DODS data set differs from a local data set only in the filename, which begins with "HTTP://" for a DODS data set. The LAS uses data from the DODS-SRB interface to plot an image as show in Figure 10.

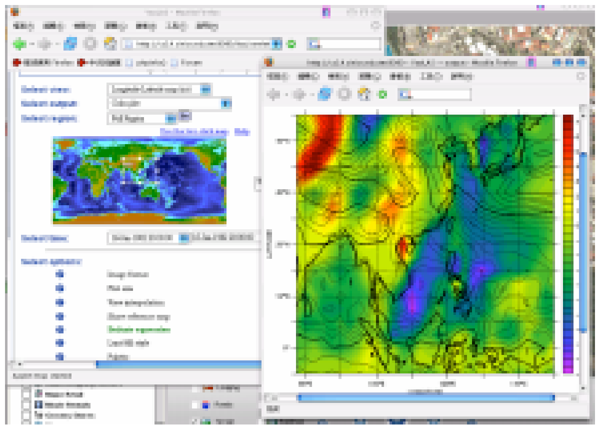


Fig 10 LAS access to a dataset from the SRB System.

5.2 GIS Applications integrate with SRB

Accessing, storing, and managing GIS images, maps, and GIS metadata has just become easier and more transparent for departmental and enterprise GIS users. The Light Weight SRB GIS Server is a dedicated system specifically used to access, manage, and store GIS image data. In this case, we integrate the SRBfs module, which was developed by BIRN, to enable client workstations to connect to the SRB data-grid and access the entire data-grid repository in a local VFS mount. This is designed to facilitate collaborative scientific workloads, grid-computing pipelines and distributed file system management. The system architecture is illustrated in Fig 11.

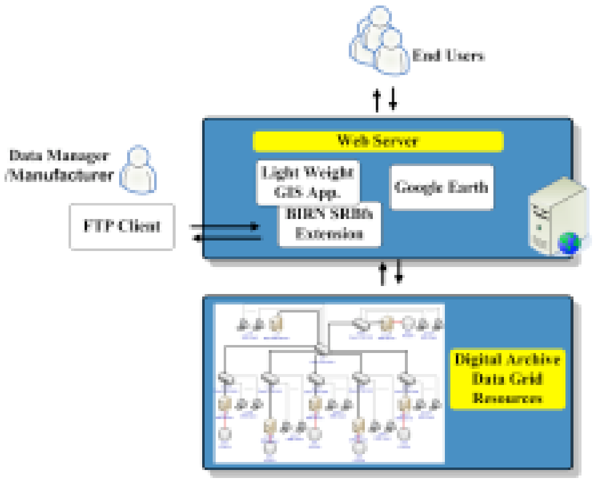


Fig 11 The System Architecture of GIS integration with SRB

The Light Weight GIS was developed by the GIS Team at the Academia Sinica Computing Centre, Taiwan. By integrating the Light Weight GIS Server, SRBfs and SRB, all GIS data can be archived in distributed resources. With the Light Weight GIS, the use of remote GIS data and access to map data becomes easier. Figure 12 shows the SRB-GIS integration.

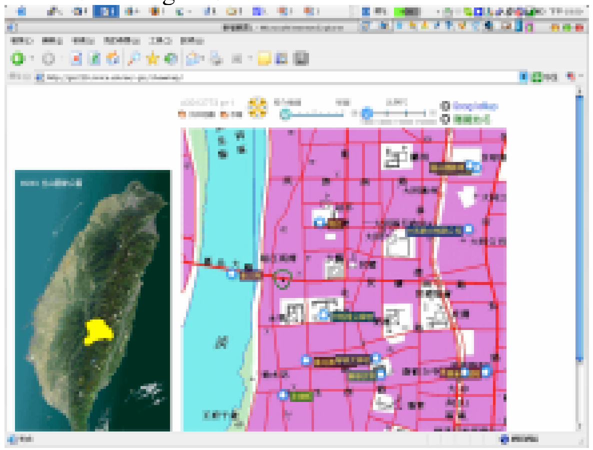


Fig 12 The Light Weight GIS integration with SRB

Using the same ideas, GIS map data can also be shown on Google Earth.

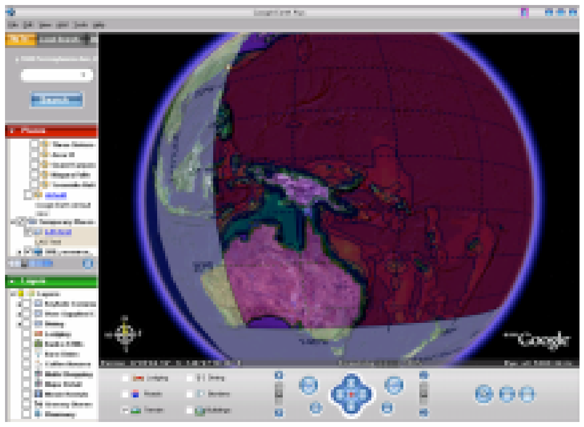


Fig 13 The SRB Map data show on the Google Earth

6. Future Works

We will keep improving the SRB related clients and administration software tools to provide better data grid services to the national digital archive program. In addition, we plan to develop some other tools to support content management.

As to the Sync package, we plan to add more features for logging and booking. Besides email notification to the manager, we will design a log repository on the server side to keep every data transfer event of end users, including the user identity, timestamp, data transfer rate and other relevant information. The logs should be uploaded into the repository through the Sync package. Sometimes the log may be too verbose and not user readable. Therefore, we require a log analyzer, which can be considered as an agent to parse the logs and extract some useful information in the log repository. Thus the log analyzer will also be the interface between the LB information system and the log repository, which is responsible for transforming the log events into a well defined user readable schema. Then users will be able to keep track of the data ingestion information via the LB information system.

The administration tool, System Report Generator, has been acting as the core monitoring component in our backend storage management framework. We will focus on improvement of the load resource management design. The load resource management is now available for monitoring the physical resources defined in SRB. We will support logical resource monitoring in the near future.

The content search is undoubtedly essential to the content management in the national digital archive program. The Metadata Catalog (MCAT) provides attribute based metadata discovery for the users to access their dataset, but so far it does not support the feature of full-text retrieval. Moreover, it may take a lot of effort to implement several interfaces for the users to define the attribute based metadata on a data item regarding its content prior to conducting the search.

Therefore, we plan to propose an index service for SRB as one of the content management solutions. We intend to develop a SRB robot to perform batch indexing tasks. The robot will be capable of extracting the content of certain document types and indexing the content of these documents as well as the system metadata attributes, such as data name, data owner and etc. As a result, the users can conduct full-text search via the search engine provided by the index service. In our experience, the index service often yields better query performance when multiple tables are involved in the query. With the index service, the content search for the long-term preserved data does not necessarily rely too much on commercial DBMS support.

7. Conclusions

Data Grid Services are becoming increasingly important in scientific communities for sharing large data collections and for archiving and disseminating them in National Digital Archive Project. The Storage Resource Broker provides transparent virtualized middleware for sharing data across distributed, heterogeneous data resources separated by different administrative and security domains. In this paper we saw brief descriptions of the use of the SRB infrastructure in the Data Grid Architecture for building distributed data collections, Digital Archives, and persistent archives. In the future, more and more applications will integrate with Digital Archives Data Grid Services based on SRB.

References

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

[2] 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.

[3] 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.

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

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

[6] Eric Yan, "Toward a Data Grid for Digital Archive," 2002 PNC.

[7] SRB system monitoring tool - System Report Generator, http://srb.grid.sinica.edu.tw

[8] Sommands – A Command Line Interface for SRB, *http://www.sdsc.edu/srb/scommands/index.html*l

[9] Arcot Rajasekar, Michael Wan, Reagan Moore, George Kremenek, and Tom Guptill, " Data Grids, Collections and Grid Bricks," *20th IEEE/ 11th NASA*.

[10] Steve Hankin, Jonathan Callahan, and Joseph Sirott, "THE LIVE ACCESS SERVER AND DODS: WEB VISUALIZATION AND DATA FUSION FOR DISTRIBUTED HOLDINGS", *http://ioc.unesco.org/oceanteacher/OceanTeacher2/06_OcDtaMgtProc/04_VirtCtrs&DistSys/LASoverview.htm*

[11] SRBfs, BIRN Grid File System, *http://www.nbirn.net/Resources/Users/Applications/SRBfs/*

[12] TWGrid. http://www.twgrid.org.

[13] National Digital Archives Program, http://www.ndap.org.tw

[14] GIS Team, Academia Sinica Computing Centre. http://gis.ascc.net.

SRB Image Archive with Cropping and Scaling for Environmental Niche Modeling

David Stockwell
University of California, San Diego
davids@sdsc.edu

Bing Zhu
University of California, San Diego
bzhu@sdsc.edu

Haowei Liu
University of California, San Diego
haoweiliu@gmail.com

Abstract

Here we describe the use of the Storage Resource Broker (SRB) to support new data intensive approaches to Environmental Niche Modeling (ENM) by providing access to cropped images from a remote SRB data store of almost 1000 global coverage data sets. The archive is currently used in conjunction with a data mining algorithm called WhyWhere (available at http://biodi.sdsc.edu/ww_home.html).

Introduction

The basic architecture of the system is illustrated below (Figure 1). ENM is a generic name for a range of geospatial modeling methods that model the probability of species occurrence with respect to environmental variables (the ecological niche), and then uses this model to project a distribution onto the landscape. Processing massive environmental data sources present a challenge to existing ENM techniques.

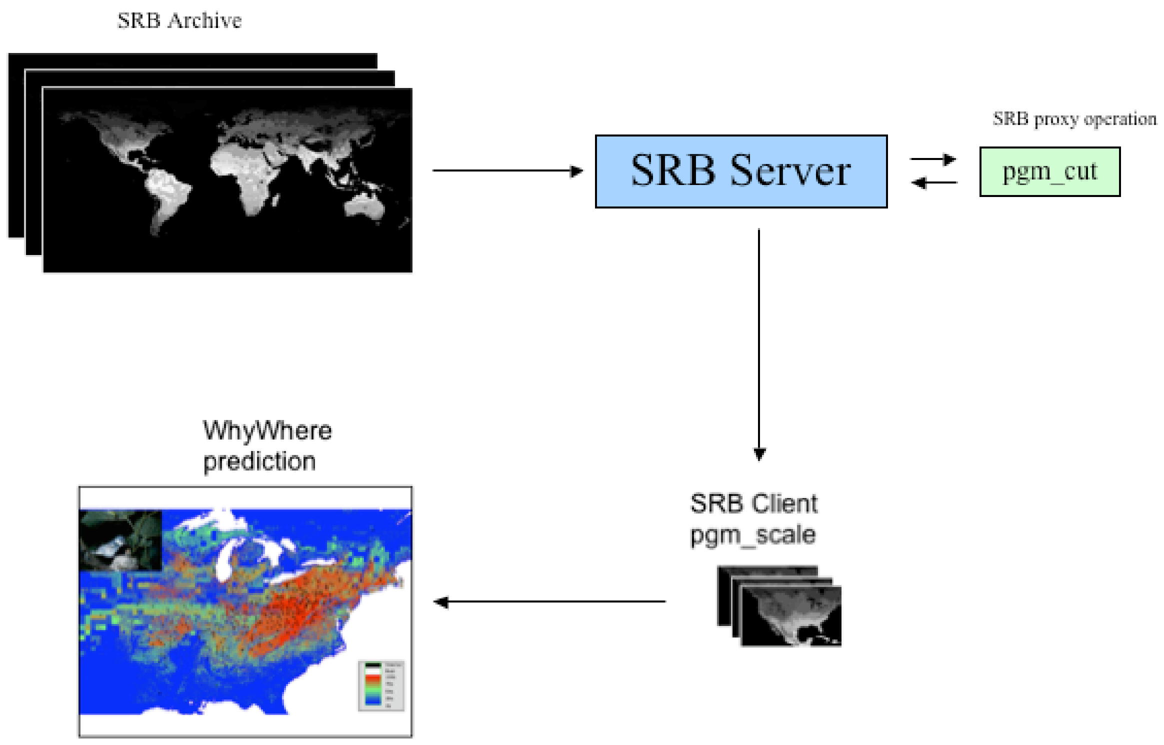


Figure 1. Illustration of the components and operation of the SRB data archive for ecological niche modeling. A large set of images and meta data are stored in a central archive. The client directs the server to crop an image in the archive using a server-side proxy operation. The cropped image is copied to the local directory and scaled by the client to the resolution required for the prediction algorithm. Illustrated is a prediction of a North American bird, the Cerulean Warbler.

Whereas in the past, ENMs were developed with small numbers of primarily climatic variables, such as annual average temperature and rainfall, this approach ignores a large number of potentially better correlates including monthly temperatures and rainfall, functions of these variables such as standard deviations, and those related to water availability and evapotransporation, soil and vegetation habitat conditions, and topography. Extended into the marine environment, each variable potentially exists in 3 dimensions. Combine this with remote sensing data, the existence of alternative versions of variables, different scales, and temporal factors such as time and duration and the number of variables that a researcher might want to examine for potential correlates expands rapidly.

The collection is called WhyWhere and can be viewed with inQ v3.3.1 (http://www.sdsc.edu/srb/software.html) using the following settings: Name: testuser Host: orion.sdsc.edu, Domain: sdsc, Port: 7613, Authorization: ENCRYPT1, Password: TESTUSER, or by downloading and using routines in the WhyWhere application [1].

Archive Architecture

The aim of this SRB archive was to provide a source of environmental correlates for ecological niche modeling (ENM) from a massive archive of data. The second element of the architecture is an algorithm for efficiently mining for environmental correlates, described in [1].

Here we describe the needs and solutions that drove the environmental data component supplying the spatial mining algorithms to use a server-side image cropping operation called 'pgmcut'. By putting the 'pgmcut' into server side we avoid the overhead of downloading the whole image into the client. This remote partial file transfer capability is accomplished by treating 'pgmcut' as a proxy operation within SRB server.

The approach of adding server-side operations to facilitate access to parts of images in archives is similar to that of another SRB user, the International Virtual Observatory Alliance (IVOA). While the IVOA have developed a more capable Simple Image Access Specification defining a cgi-based interface including cutting and mosaicing, the parameters use astronomical coordinates. The Open GIS Consortium (OGC) has a more appropriate geographic specification that could be used to provide a "wrapper" for accessing the archive via a GetMap request with latitude and longitude parameters.

Other benefits of using SRB as an archive place include location encapsulation for archived files and metadata, meaning that a 'pgmcut' client doesn't need to know where the actual image and data are stored. As a distributed data management system, SRB provides virtually unlimited storage space for geospatial data/images. The constraints were as follows and are described in turn, but the fourth required the extensions to the SRB that we describe in more detail: (1) single format for at least 1000 data variables, (2) all data sets of global extent but variable scale, (3) all data described with meta data, and (4) each variable supplied cropped and scaled to a specific size and resolution.

Format. The format used for storing the variables was a pgm image. This is a simple gray scale image with one byte per pixel and a simple text header giving the format, the extent and the number of shades of gray in the image. For example, below is the first few lines of a pgm binary file with dimension 3600_1800 and 255 colors.

```
P5
3600 1800
255
... (binary data)
```

While many variables such as categorical variables describing vegetation, landscape or soil types contained fewer than 256 values, continuous value variables were simply normalized between 0 and 256 according to their maximum and minimum values. While this trade off for efficiency resulted in loss of information, it is not as problematic as it might seem given the WhyWhere algorithm used in mining the data set is only looking for statistical associations. The WhyWhere algorithm categorizes all variables into less than this number of categories anyway, and the reduced size provides both storage and computational efficiencies.

Scale and Extent. It was also decided to restrict the database to variables with the same global extent and geographic projection to make extracting information consistent. Given all variables are global, all can potentially be used in an analysis. The scale varies between 1 degree per grid cell (i.e. 360_180 pixel image) to 1km per grid cell, approximately 1GByte total size.

Metadata. The meta data format consists of simple attribute/value lines as used in the Global Ecosystem Database (GED), one of the main sources of data. An example is shown below. While more complex meta data would be useful, but this format was adequate for our purposes. The meta data documents images, provides the dimensions to allow extraction by the cropping algorithm so that parts of the image file can be accessed by the SRB.

```
file title : Legates & Willmott Annual
Temperature (0.1C)
data type : integer
file type : binary
columns : 720
rows : 360
ref. system : lat/long
ref. units : deg
unit dist. : 1.0000000
min. X : -180.0000000
max. X : 180.0000000
min. Y : -90.0000000
max. Y : 90.0000000
pos'n error : unknown
resolution : 0.5000000
min. value : -569
max. value : 299
value units : 0.1 degrees Celsius
value error : unknown
```

Operations. Two main operations are needed, exemplified in the local image processing library netpbm as 'pgmcut' and 'pgmscale'. The 'pgmcut' operation produces a partial image labeled by 'B'. In the 'pgmscale' operation, we want to change the x and y extent to a given size, either increased (B to a) or reduced the size (a to B) (Figure 2).

```
aaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaa
aaaaBBBBBBBBaaaaa
aaaaBBBBBBBBaaaaa
aaaaBBBBBBBBaaaaa
aaaaaaaaaaaaaaaaaaa
```

Figure 2. Array of pixels for illustrating cropping and scaling operations. In a crop the array extent 'a' is cropped to 'B'. In scaling the entire array 'a' is reduced in size to 'B' or the array 'B' is increased in size to 'a'.

We wanted to develop a generic approach to accessing the geographic data sets, paying attention to increased usage in the future. There were a number of options to developing the right approach to balancing the client/server load. Based on our experience dealing with remote file transfer software, currently no such software can handle the requirement of a 'pgmcut' from SRB in a distributed environment.

One approach could be to download whole file into a local machine and operate on the image locally. This is not a good solution in terms of performance as many files are greater than 1GByte and we just need partial data.

Although grid-ftp provides partial file transfer, it cannot be done by one function call for the above case. So the client side (grid-ftp client) has to repeatedly calculate offsets and then make grid-ftp calls with new offsets and numbers of bytes for data transfer. In this case, we create a customized client that then assembles the resulting lines of the data to create the image. We tried this with the SRB using the SRB read library function. However initial experiments were very slow, presumably due to latency of the Internet connections that are made and broken for each line of the image.

Finally we extended SRB functionality by developing a cropping function on the server side, and kept the scaling function on the client side. The server side reads the header of the image file, retrieves and assembles a series of lines from the file corresponding to the area needed then passes it to the calling client. We found it to be reasonable performance and is the 'generic' solution currently used in the WhyWhere application.

The SRB connection is carried out as a proxy operation on the server through the use of a program in the client called rpgm. The WhyWhere application iterates along a list of image names to retrieve as selected by the user. For each image it retrieves a version cropped according to the latitude and longitude required. To do this, the client side calculates the pixel coordinates needed based on local metadata and makes calls of the form:

```
$ Sinit
$ Spcommad "Spgmcut 0 0 20 20
/home/whywhere.seek/ei/Data/Terrestrial/a
```

```
00sd1.9.pgm" > temp.pgm
$ pnm_scale -xsize $xwall -ysize $ywall
temp.pgm >$RDIR\$DIR\t_$i.pgm";
```

The above Spcommand sends the only command line argument to SRB server. The command line argument has the proxy program name, Spgmcut, parameters and file name. The result is sent to 'stdout' which, in above example, is directed to a local file, temp.pgm. The SRB S-commands in 'landscape.sdsc.edu' can be found in the following directory of the WhyWhere distribution – WW/cgi-bin/UTIL. While initial tests indicated adequate performance, once a number of people started using the service the performance became variable according to the size of the files (the biggest are 1GByte each) or the load on the server. Currently a call to Remote_All_Data is an overnight task.

Archive Contents

The data sets were collected from various free sources on the web in a variety of formats and processed into pgm images. The following were major sources:

The Global Ecosystems Database (GED). The GED project began in 1990 as an Interagency project between the National Geophysical Data Center (NGDC) of the U.S. National Oceanic and Atmospheric Administration (NOAA), and the U.S. Environmental Protection Agency's (EPA) Environmental Research Laboratory in Corvallis, Oregon (ERL-C). In particular the following variables added, many consisting of multiple layers (e.g. monthly mean and standard deviations of temperatures).

A01: NGDC Monthly Generalized Global Vegetation Index from NESDIS NOAA-9 Weekly GVI Data (APR 1985 – DEC 1988).
A05: Olson World Ecosystems.
A06: Leemans Holdridge Life Zone Classifications.
A07: Matthews Vegetation, Land Use, and Seasonal Albedo.
A10: Wilson and Henderson Sellers Global Land Cover and Soils Data for GCMs.
B01: Fedorova, Volkova, and Varlyguin World Vegetation Cover
B02: Bazilevich Global Primary Productivity
B03: Bailey Eco regions of the Continents (reprojected)

WORLDCLIM. This is a set of global climate layers (grids) on a square kilometer grid supported by NatureServe. The bioclimatic variables represent physiologically relevant annual trends (e.g., mean annual temperature, annual precipitation) seasonality (e.g., annual range in temperature and precipitation) and extreme or limiting environmental factors (e.g., temperature of the coldest and warmest month, and precipitation of the wet and dry quarters) as follows:

BIO1 = Annual Mean Temperature
BIO2 = Mean Diurnal Range (Mean of monthly (max temp – min temp))
BIO3 = Isothermality (P2/P7) (* 100)
BIO4 = Temperature Seasonality (standard deviation *100)
BIO5 = Max Temperature of Warmest Month
BIO6 = Min Temperature of Coldest Month
BIO7 = Temperature Annual Range (P5-P6)
BIO8 = Mean Temperature of Wettest Quarter
BIO9 = Mean Temperature of Driest Quarter
BIO10 = Mean Temperature of Warmest Quarter
BIO11 = Mean Temperature of Coldest Quarter
BIO12 = Annual Precipitation
BIO13 = Precipitation of Wettest Month
BIO14 = Precipitation of Driest Month
BIO15 = Precipitation Seasonality (Coefficient of Variation)
BIO16 = Precipitation of Wettest Quarter
BIO17 = Precipitation of Driest Quarter
BIO18 = Precipitation of Warmest Quarter
BIO19 = Precipitation of Coldest Quarter

World Ocean Atlas 2001 (WOA01) Data for Ocean Data View. The objectively analyzed global ocean historical hydrographic data from the U.S. NODC World Ocean Atlas 2001. Data are on a 1_1 degree horizontal grid and at the following standard depths (in m): 0, 10, 20, 30, 50, 75, 100, 125, 150, 200, 250, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1750, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 5500. Data are available for the following variables:

Temperature [°C]
Salinity [psu]
Oxygen [ml/l]
Oxygen Saturation [%]
AOU [ml/l]
Phosphate [~$m~#mol/l]
Nitrate [~$m~#mol/l]
Silicate [~$m~#mol/l]

Continuous Fields 1 Km Tree Cover. This dataset was developed by DeFries, R. Hansen, M., Townshend, J.R.G., Janetos, A.C., Loveland, T.R. at the University of Maryland. It uses an alternative paradigm to describe land cover as discrete classes by representing land cover as continuous fields of vegetation characteristics using a linear mixture model approach. This data set contains 1km cells estimating:

Percent tree cover
Percentage cover for two layers representing leaf longevity (evergreen and deciduous)
Percentage cover for two layers estimating leaf type (broadleaf and needleleaf)

HYDRO1km. This dataset was developed at the U.S. Geological Survey's Center for Earth Resources Observation and Science (EROS), the HYDRO1k provides derivatives of hydrological data including:

Elevation
Aspect
Flow Accumulation
Compound Topographic Index
Drainage Basins
Slope
Flow Direction
Streams

Future Needs

Future needs are many if the archive is to transition into a production resource for the general research community. The following are some of the main challenges:

It is envisaged that traffic could expand considerably, as each analysis requires download and processing of a large section of the collection by each researcher each time they start analysis, so local caching is only a small saving. Meeting this need is initially envisaged via replication of SRB archives and a means in the client of selecting the most efficient (e.g. proximate) archive for connection and download.

If the popularity expands, attention will be paid to transitioning the archive to a community of developers for maintaining, updating, cleaning and improving the archive. Currently we are developing a model of modeling tools for R that would incorporate the utility. Other activities include outreach to the OpenModelling (http://sourceforge.net/projects/openmodeller/) project, and also maintaining a Weblog site to facilitate education and promotion about data intensive approaches (http://landscape.sdsc.edu/~davids/enm).

Finally, improved methods of access are needed, potentially based around the WMS OpenGIS specification to allow more flexible interaction with servers including selection of variables for analysis.

Acknowledgement

This work was partially funded by the National Science Foundation primarily through the National Science Foundation grant SEEK: Science Environment for Ecological Knowledge (DBI0225674).

References

[1] Stockwell D.R.B. (in press) Improving ecological niche models by data mining large environmental datasets for surrogate models, *Ecological Modelling*, [arxiv:q-bio/0511046].

VOSpace and VOStore Design

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

Abstract[1]

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

1. Introduction.

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

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

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

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

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

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

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

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

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

2. VOStore:

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

A simple VOStore interface would support:

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

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

3. VOSpace:

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

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

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

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

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

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

4. VOSpace implementation

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

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

The design principles on which the SRB is based are:

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

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

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

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

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

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

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

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

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

Near-real-time Backup of Large Seismic Waveform Datasets with the Storage Resource Broker

Kent G. Lindquist
Lindquist Consulting, Inc.
kent@lindquistconsulting.com

Jennifer Eakins
Univ. of California, San Diego
jeakins@ucsd.edu

Frank L. Vernon
Univ. of California, San Diego
flvernon@ucsd.edu

Arcot Rajasekar
Univ. of California, San Diego
sekar@sdsc.edu

Abstract

The Earthscope USARRAY project at full deployment is expected to generate more than four GigaBytes of data per day from approximately 400 seismic monitoring stations. The testbed as of October, 2005 is producing 1.2 GB/day from 102 stations (612 seismic channels) in the prototype Transportable Array. These data are being acquired via an Antelope near-real-time system into Datascope databases. The large data volumes, the need for deep-archival and up-to-date retrieval of waveform files, and the desire to have these data backed up and also accessible through internet portals make the Storage Resource Broker ideal for the application. A recently constructed driver for the SRB adds Datascope relational databases as one of the possible storage resources. A corresponding client library allows access to these data with Datascope operations mediated by the SRB. We have written Datascope database utilities (in this case based also in part on SRB S-commands) that routinely and automatically archive incoming USARRAY data into the Storage Resource Broker, tracked additionally through another Datascope database table linked into the main seismic database. This archival data collection starts with data from April of 2004, and as of October, 2005 encompasses over 265,000 individual files representing slightly over 1.3 million seismic waveform segments, with a total storage allocation of 0.6 TeraBytes.

[*] This research was sponsored by a funding agency. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the Government, or any person or agency connected with them.

1. Introduction

1.1. Earthscope

The Earthscope project [1] is a National Science Foundation multi-year initiative to investigate the structure and evolution of the North American continental lithosphere as well as the physical processes controlling earthquakes and volcanic eruptions. One of the major components of Earthscope is the USArray seismic observatory [2], a continent-wide deployment of seismometers designed to study the Earth's structure over many different physical length scales. In addition to permanent stations that will enhance the US Advanced National Seismic System (ANSS), a "Transportable Array" of 400 broadband, 3-component seismic stations will be deployed and systematically moved to cover subsections of a uniform grid across the United States, as shown in Figure 1. In addition, a "Flexible Array" will augment these for active and passive seismic experiments.

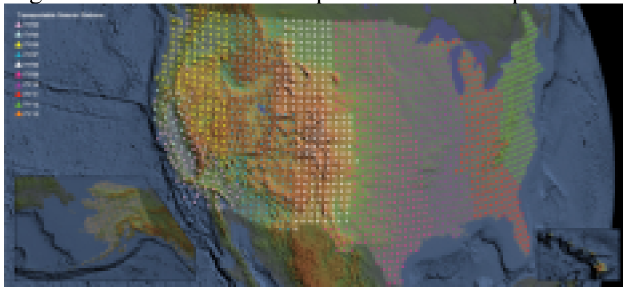


Figure 1. The Transportable Array

1.2. The Array Network Facility

Data received from the Transportable Array and Flexible Array seismic stations will be acquired and initially processed by the Array Network Facility (ANF) located at the Institute for Geophysics and Planetary Physics (IGPP) and Scripps Institution of Oceanography (SIO) at the University of California, San Diego [3]. The ANF will be responsible for maintenance of all station metadata, as well as quality control of incoming seismic data and control of the running stations.

1.3. Antelope

The Array Network Facility runs real-time acquisition and processing systems based on the Antelope Environmental Monitoring System produced by Boulder Real-Time Technologies, Inc. (BRTT) [4]. The Antelope suite of programs and programmer interfaces contains both real-time data transport, acquisition, and gridded distribution mechanisms (the "orb" utilities), and an embedded relational database system called Datascope, which is well suited to real-time processing tasks as well as the collection and archival of real-time data. The real-time utilities include signal detectors that examine the incoming data streams for candidate seismic energy arrivals, associator programs that can derive from these detections the approximate locations of the source earthquakes or explosions, and magnitude estimators to derive the approximate sizes of the events. All incoming seismic waveforms, station metadata, and processing results are saved in a real-time Datascope database in the Antelope "rt1.0" format, a derivative of the Center for Seismic Studies "css3.0" schema which is a de-facto standard in the seismic community [5]. Files of seismic waveform data are referenced in a database table called "*wfdisc*." This relation tracks station and channel names, start and end times of each segment, and the directory name, data file name, and byte offset in the file for each contiguous waveform segment. The actual files of waveform data are stored separately in a Unix filesystem.

2. Storage Resource Broker Connectivity

While the ultimate fate of seismic waveform data in the Earthscope program will be archiving at and distribution through the Incorporated Research Institutions for Seismology (IRIS)'s Data Management Center (DMC) in Seattle [6], the scale of ANF operations requires internal capabilities for virtual access and large-volume archiving and backup to sustain robust operations. This need provides a clear role for the Storage Resource Broker (SRB) [7].

The current connection between the USArray data being acquired and the Storage Resource Broker is established through extensions to the USArray Datascope database and through SRB S-commands. Two wrapper utilities have been written to provide straightforward tools for USArray system operators: *rtbackup_srb* and *get_archive_srb*. The former runs automatically in the USArray Antelope real-time system; the latter is used as needed to recover waveform data and their corresponding database tables out of the Storage Resource Broker archives.

2.1. Database extensions

We have expanded the Antelope rt1.0 schema by adding one new table, called "*wfsrb*", which tracks SRB objects into which waveform-segment files have been copied. Many of the features of the *wfsrb* table mirror those of the css3.0 *wfdisc* table, except that the "*dir*" and "*dfile*" fields used to reference external binary large objects in the *wfdisc* table are replaced by fields appropriate to the SRB in the *wfsrb* table. A simplified sketch of the *wfsrb* relation structure is shown in Table 1.

Table 1. wfsrb extension to rt1.0 schema

Field	Content
sta	Time-series station name
chan	Time-series channel name
Time	Unix epoch start time of time-series
Endtime	Unix epoch end time of time-series
Nsamp	Number of samples in time-series
Samprate	Sample rate in Hz of time-series
Foff	Byte offset in file of time-series start
Szone	SRB Zone name for stored BLOB
Scoll	SRB Collection name for stored BLOB
Sobj	SRB Object name for stored BLOB
…….	Various other data fields indicating data type, calibration, instrument, and units representation

2.2. Rtbackup_srb

The *rtbackup_srb* program is designed to run within the context of an Antelope real-time system. Antelope real-time systems support a utility for periodic execution of operational tasks, quite similar to the Unix cron(1) program. For the USArray systems, *rtbackup_srb* is set to run once per day. For each invocation, a database no-join operation is performed to determine all the rows of the *wfdisc* table (and therefore their corresponding binary-large-object waveform files) which do not have entries in the *wfsrb* table. These files are all loaded into the SRB, and appropriate entries are made in the *wfsrb* table. A variety of check mechanisms verify that the transfers are conducted successfully, that waveform files are not repeatedly loaded if they are referenced by multiple *wfdisc* rows, and that system operators are notified of any problems. The collection name under which each waveform is stored is constructed from the *dir* field of the *wfdisc* table, combined with the top-level SRB collection for the archive as specified at run-time to *rtbackup_srb*. For cases where many waveforms (months worth or more) must be archived to a new *wfsrb* table and new SRB archive, an incremental mode is provided in *rtbackup_srb* such that the run periods of *rtbackup_srb* may be interleaved with real-time data acquisition (these tasks must be coordinated to prevent corruption and/or significant latency in the real-time databases). In addition to backing up the raw waveform files, *rtbackup_srb* will copy any and/or all of the tables of station metadata, seismic processing results, and other parametric data to the SRB. Any tables that are backed up in the SRB are automatically replicated with a version string giving the current system time in UTC epoch seconds. This provides a convenient rollback facility for retrieving seismic processing results, since by nature those tables are updated continuously by the Antelope real-time system. Finally, the entire top-level collection into which *rtbackup_srb* saves its archives can be automatically replicated to one or more alternate SRB physical resources, thus immediately producing a much safer archive through the strengths of the SRB, as well as providing database copies in other convenient locations. This automatic replication, if enabled, is accomplished with the SRB *Sbkupsrb*(1) command. The architecture of *rtbackup_srb* is shown in context in Figure 2.

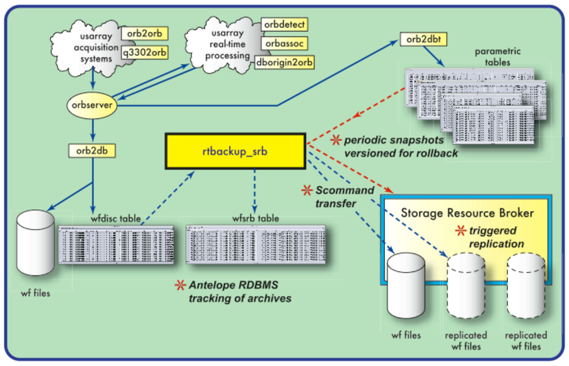


Figure 2. Context of rtbackup_srb.

2.3. Get_archive_srb

The *get_archive_srb* script retrieves waveform data that has been archived in the Storage Resource Broker, as described by a *wfsrb* database table, and extracts the raw waveforms files as well as recreating an appropriate *wfdisc* table. The standard mode in which to run *get_archive_srb* would be as a bulk extraction of waveform files. However, *get_archive_srb* also contains a fairly powerful and general mechanism to preprocess the database and create a complex input view on which to base the extraction. This may be used, for example, to extract waveforms for specific earthquakes, sets of arrivals, seismic-phase moveout patterns across the array of stations, and so on. Additionally, subsets may be applied to constrain the extraction to particular time ranges, stations and channels, etc. These constraints may be important in many cases since the SRB may contain far more data than a given local filesystem and *wfdisc* could ever hold. A sketch of the architecture for *get_archive_srb* is shown in Figure 3.

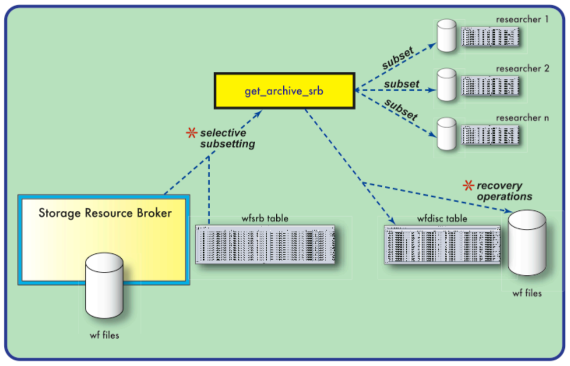


Figure 3. Context of get_archive_srb

3. Conclusions

The *rtbackup_*srb utility has been running at the ANF since October of 2004, with over 0.6 Terabytes of data (dating from April, 2004 onwards) backed up in the SRB. This encompasses over 265,000 individual files representing slightly over 1.3 million seismic waveform segments, as well as version-stamped daily snapshots of all the database tables representing seismic processing information (seismic arrival detections, located earthquake hypocenters, etc.). The archived files and databases, along with the *get_archive_srb* utility, have already proven useful on more than one occasion to contribute to robust ANF operations. Please direct any technical questions or feedback to Kent G. Lindquist, kent@lindquistconsulting.com.

4. Acknowledgments

This work and the ROADNet project are sponsored by grants from the National Science Foundation (OCE-0121726), the Office of Naval Research (N00014-98-1-0772), and with matching funds from the California Institute of Telecommunications and Information Technology, Scripps Institution of Oceanography, San Diego State University, and the Cecil H. and Ida M. Green Institute of Geophysics and Planetary Physics. We thank Breck Betts for assistance with the figures for *rtbackup_srb* and *get_archive_srb*. Figure 1 is courtesy of the USArray web-site [2]. Some of the material in this paper was presented in poster form at the 2005 meeting of the Seismological Society of America, Incline Village, Nevada [8].

References

[1] http://www.earthscope.org

[2] http://www.iris.iris.edu/USArray/

[3] http://www.anf.edu/

[4] http://www.brtt.com/

[5] J. Anderson, W.E. Farrell, K. Garcia, J. Given, and H. Swanger. "Center for Seismic Studies Version 3 Database: Schema Reference Manual", Science Applications International Corporation, Arlington, Virginia, Technical Report C90-01 (1990), 61 pp.

[6] http://www.iris.iris.edu/USArray/facilities.htm

[7] http://www.sdsc.edu/srb/

[8] K.G. Lindquist, J.A. Eakins, F.L. Vernon, A. Rajasekar "Near-real-time backup of large seismic waveform datasets with the Storage Resource Broker." *Seis. Res. Lett.* **76** (2005), 230.

Some Tools for Supporting SRB Production Services

R. Downing
CCLRC-Daresbury Laboratory

A. Weise, C. Koebernick
University of Reading

A. Hasan
CCLRC-Rutherford Appleton Laboratory

Abstract

Providing production-level services requires monitoring applications, performance and intercepting errors as soon as they occur. In this paper we describe some of the tools that have been developed to assist production SRB services. We describe the approaches used and how they can be more generally applied.

1. Introduction

The Data Management Group (DMG)[1] is part of the Council for the Central Laboratory of the Research Councils (CCLRC) e-science centre [2] and provides data storage solutions for a large number of e-science projects. The DMG uses the Storage Resource Broker (SRB) [3] as the core component for many projects, tailoring the system to meet the needs of the project. Once a system is deployed the DMG also provides a level of support for the service ranging from troubleshooting to responding to further feature requests and upgrades.

Through the course of developing various SRB systems we have managed to identify a number of tasks that appear common and which greatly help in supporting a production system. In this paper we describe a few of the tools developed to aid this task.

2. Monitoring Production Servers

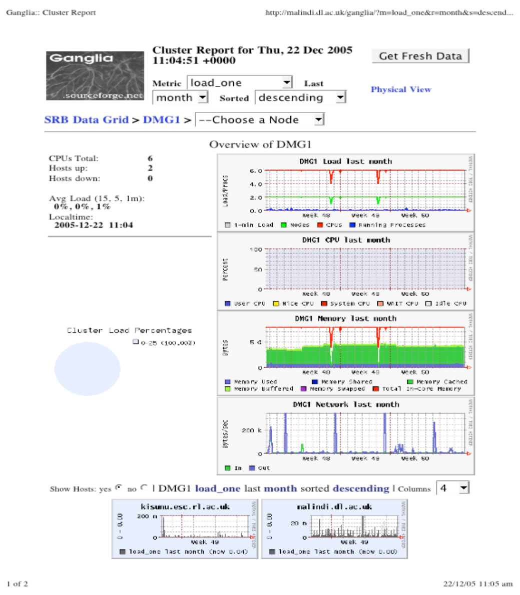
Careful monitoring of production servers provides a number of benefits: aids debugging, provides information on the distribution of load in the system and provides information for planning purposes. Troubleshooting and load balancing require both instantaneous information and also historic information whereas planning requires only historic information.

2.1. Ganglia and Nagios Monitoring

Since the SRB system is distributed any monitoring application must be capable of working with distributed systems. With this requirement in mind we have selected Nagios [4] to report instantaneous information on server properties, such as cpu, machine load, etc. The Nagios system emails a list of subscribers when any of the monitored properties of a server go beyond an acceptable threshold limit as well as reporting when a server is down.

For the collection of historic information we chose Ganglia [5]. The Ganglia monitoring system collects a set of system properties at regular intervals and stores them in a round-robin database. It is also possible to monitor additional properties by providing a script to extract these properties to Ganglia. The system also provides tools for presenting the information as a series of web-pages (see figure 1). As we run more than one SRB server on a given host we needed to make a minor kludge to allow the same host to appear in more than one group.

[*] This work has been funded by a range of UK agencies incl. the e-Science Programmes of the Natural Environmental Research Council, the Engineering and Physical Science Research Council, the Council of the Central Laboratory of the Research Councils, the Biotechnology and Biological Sciences Research Council and the Joint Information Systems Committee.


Ganglia:: Cluster Report http://malindi.dl.ac.uk/ganglia/?m=load_one&r=month&s=descend...

Cluster Report for Thu, 22 Dec 2005 11:04:51 +0000

Metric load_one Last month Sorted descending

SRB Data Grid > DMG1 > --Choose a Node

Overview of DMG1

CPUs Total: 6
Hosts up: 2
Hosts down: 0

Avg Load (15, 5, 1m):
0%, 0%, 1%
Localtime:
2005-12-22 11:04

Show Hosts: yes no | DMG1 load_one last month sorted descending | Columns 4

1 of 2 22/12/05 11:05 am

Figure 1: Ganglia web page displaying usage for a test SRB server.

3. Monitoring SRB Server Log Files

Each SRB server writes activity information to a log file. These log files contain information about which process, and from which machine, connected to the SRB server as well as error messages detected by the server when handling a request. These error messages along with the time that they occurred are an essential tool in troubleshooting. It is important to notify administrators as soon as an error occurs, it is also important to log the error messages in order to identify chronic problems and possibly identify patterns.

Any application to monitor the log files would need to be able to parse the log files for error messages, email to a subscriber list serious errors and collect in a central location the error messages for later searches. With these requirements we decided to build a system in Python to parse, log and notify when error messages occurred [6].

It is possible that Ganglia could be used to parse the log files and store the resulting error messages in a central round-robin database, but we found that the database was not flexible enough for our queries and we also required email notification when problems occurred.

The system essentially consists of three components: a Parser a Collector and a Displayer, figure 1 shows a simple diagram of how the application works.

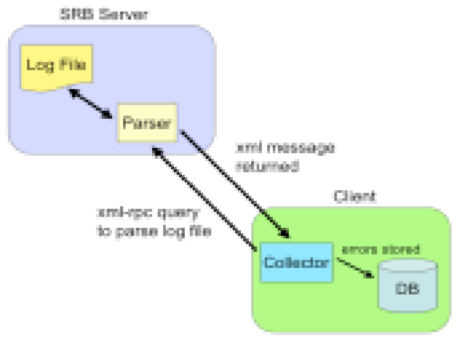
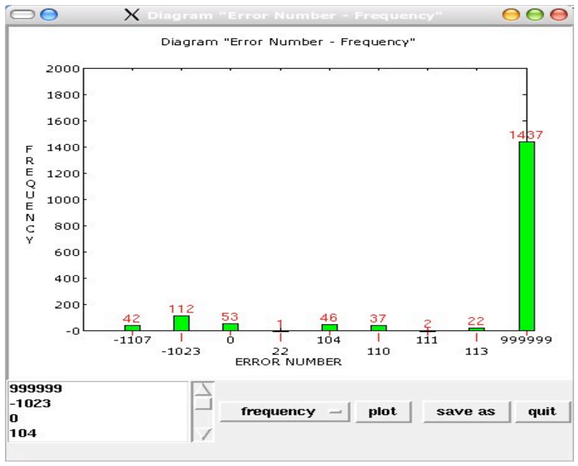


Figure 2: A simple schematic showing the log file parser system.

The Parser is actually an XML-RPC server that is started on the SRB server host and consists of a method to parse the SRB log file. The Collector is a daemon that sends an XML-RPC message to the Parser to parse the log file. The parser then returns an XML message containing the error message, line number, date, server and error message code to the Collector. The Collector then extracts the information from the XML message and stores the contents in an SQLite database and sends an email containing the error message information to a list of subscribers. The list of SRB servers that the Collector should contact and the frequency with which to contact them is read from a configuration file.

The Displayer is used to graphically display the error messages as a function of server that can help in identifying potentially chronic problems with a server. The Displayer can also plot error messages of a particular type as a function of time that may reveal interesting patterns that could help troubleshooting. Figure 2 shows a screenshot of a histogram of error messages for a given server.



999999
-1023
0
104

Figure 3: Screenshot of the error message numbers extracted from an SRB log file.

The numbers above the bars correspond to the actual occurrences of errors with that error number and error number 999999 corresponds to messages that do not have an SRB defined error number.
The Parser assumes all messages are error messages unless the user specifies in a configuration file a pattern contained in messages that should be ignored. The approach of assuming every message is an error ensures that we do not accidentally miss an unusual error message.

4. Tools for Measuring Performance

Measuring the performance of a system is important as it helps to determine the capabilities of the system, it helps to determine bottlenecks in the system and it provides a means of tuning a system. We have developed a framework that can be used to run performance tests [7] and a number of scripts that execute performance tests using Scommands on an SRB system.

The framework consists of the Ganglia monitoring system to monitor the SRB server and client application, an SQLite database to hold the measurements and Collector collect the results from Ganglia and store them in the SQLite database. The framework can also display, in real-time, graphs of the server properties as a function of time. A Displayer is also provided to graphically display previous data with the option to overlay previous performance tests. Figure 3 shows a simple schematic of the framework.

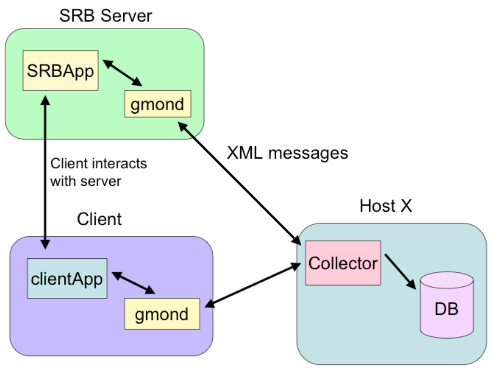


Figure 4: Schematic of the framework for performance measurements.

The Ganglia gmond daemons on the client and server machine are started by the Collector daemon before the performance tests start. The Collector collects the monitoring information in the form of XML messages at periodic intervals, extracts the information from the XML message and stores it in the SQLite database.
At this point the client application can be started and the performance measurements are recorded. The Collector reads from a configuration file the host names and applications that should be monitored as well as the interval at which the data should be collected. Figure 4 shows the cpu-load graph produced by the Displayer.In principle, the framework is not tied to the SRB and can be used for any application.

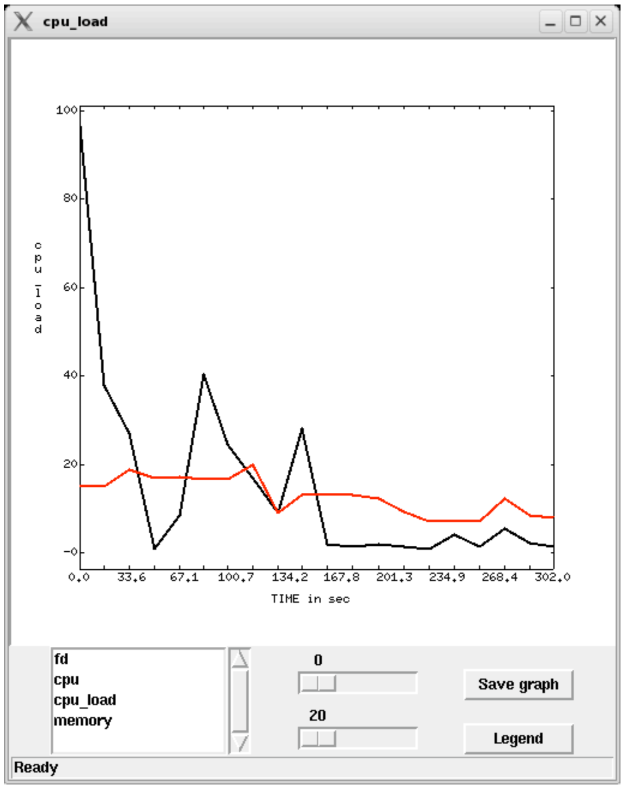


Figure 3: Graph of cpu-load produced by the Displayer application.

In order to measure the performance of an SRB system we have developed a set of tools based on the Scommands. The tools are capable of storing information in the SRB as collections, containers or simply files. The tools are configurable and can store large numbers of objects in flat or nested directory structures and are also capable of producing nested collections. The tools can also store variable amounts of metadata within the SRB.

5. Conclusion

Monitoring a production system is an essential aid in planning future extensions to the system, it can also be an essential aid in troubleshooting. Tools to carry out performance tests and collection, store and present the data are also important as they provide a means of providing a references against which the production system performance can be measured. Such tools can also help in troubleshooting problems either by comparing the performance against a benchmark, or simply by exercising a particular aspect of the system.

In this paper we have described a few of the tools that we have developed to help our production systems. All the tools we have developed are extensible as they have to accommodate new features or aspects of the production system.

References

[1] http://www.e-science.clrc.ac.uk/web/groups/Data-Management/Data-Management

[2] http://www.rcuk.ac.uk/escience

[3] http://www.sdsc.edu/srb

[4] http://www.nagios.org

[5] http://ganglia.sourceforge.net

[6] A. Weise, *M.Sc Thesis (in preparation).*

[7] C. Koebernick. *M.Sc Thesis (in preparation).*

Building a Demonstration Prototype for the Preservation of Large-Scale Multimedia Collections[*]

Arcot Rajasekar (PI)
Richard Marciano
Reagan Moore
Chien-Yi Hou
Francine Berman (co-PI)
San Diego Supercomputer Center, Univ. of California, San Diego

Lynn Burstan (co-PI)
Steve Anderson
Mellisa McEwen
Bee Bornheimer
UCSD-TV, Univ. of California, San Diego

Harry Kreisler
UCTV-Berkeley

Brian Schottlaender (co-PI)
Luc Declerck
Brad Westbrook
Arwen Hutt
Ardys Kozbial
Chris Frymann
Vivian Chu
UCSD Libraries, Univ. of California, San Diego

Abstract

The NSF-DIGARCH is building digital preservation lifecycle management infrastructure for the preservation of large-scale multimedia collections. The infrastructure consists of interfaces to TV production lifecycle systems, metadata definition and capture systems, and a persistent archive workflow which preserves the material in a SRB data grid. Kepler is used to build the workflow.

1. Introduction

The preservation framework development is viewed as a three part process which needs to interact constantly. The first part of this framework is the pre-existing video production lifecycle that should be preserved as much as possible; the second part is the metadata flow, capture and modeling framework that needs to be addressed in order to access, capture and finally preserve the additional material that is needed to complete the preservation packages, and; the third part of the framework is the persistent archive infrastructure and associated workflows [1]. Figure 1 provides a schematic view of this framework.

2. Video Production

For nearly 25 years now, Harry Kreisler has been conducting interviews as part of the "Conversations with History" series [2]. Over 230 guests have been interviewed, including diplomats, statesmen, soldiers, economists, political analysts, scientists, historians, writers, foreign correspondents, activists, and artists. These interviews are one-hour video-taped conversations.

This significant "at risk" collection includes video, audio, text transcripts, web-based material, databases of administrative and descriptive metadata and contains diverse types of data, created at multiple stages within the content production workflow.

Initial "archiving" of the video content has 230 programs in 3 formats:

- digital master files in DV format (.mov files) of typical size 12GB (compressed)
- UCTV broadcasting file in MPEG format of typical size 2GB
- Web archive files in Real Player format of typical size 200 MB

This makes the video content roughly 15GB per show or 230 x 15GB = 3.5TB for the complete collection. When preserving this content, we will replicate the collection in at least two locations, making this a 7TB persistently archived collection. The video metadata is stored in a FileMaker Pro 3.0 database at UCSD-TV [3]. Initially, a subset of this database will be exported for preservation. A dedicated Macintosh machine "eMac" was set up in the television laboratory at UCSD-TV. This machine will be used to convert the master tapes to digital DV format using Final Cut Pro (eventually, several platforms will be used to convert 2- or 3-way to speed up the process, as the conversion takes place in real-time). The eMac has a storage capacity of 183GB and was augmented with a 500GB external hard-drive. We also installed a version of Timbuktu Pro for remote access, the Kepler [4] software and a client version of the SRB [5].

[*] This research was sponsored the National Science Foundation and the Library of Congress. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the Government, or any person or agency connected with them.

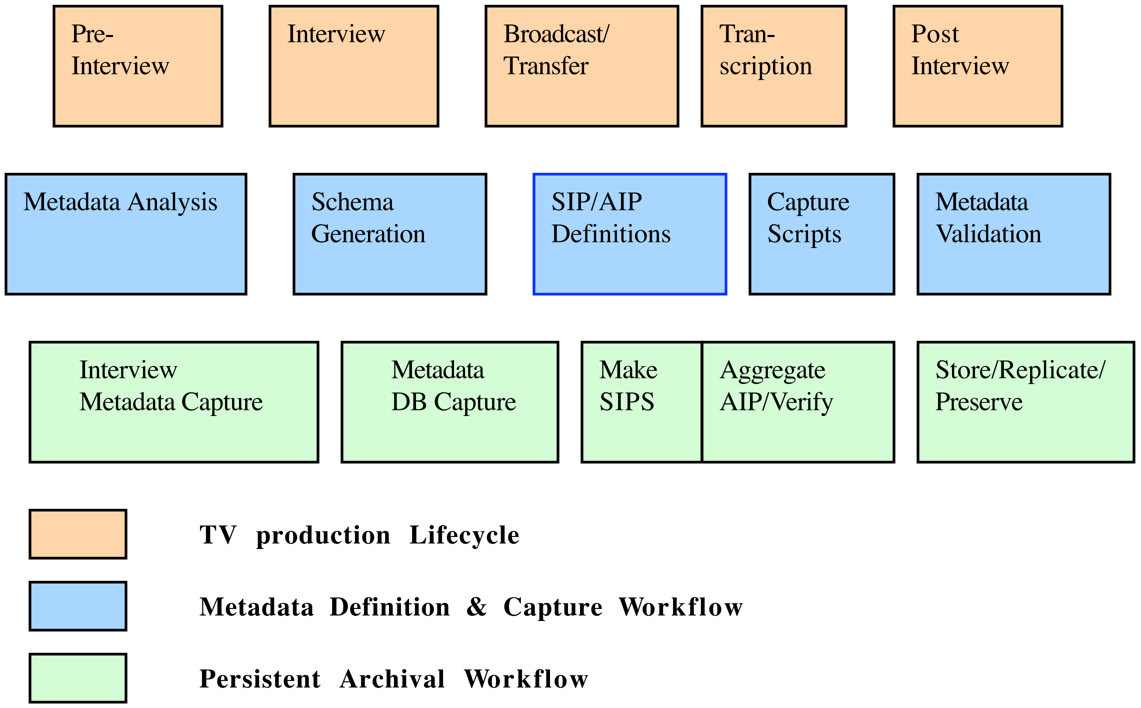


Figure 1. DIGARCH Preservation Framework

3. Metadata and Modeling

A series of modeling exercises of the existing production workflow were carried out. Two production workflows are described. The first is for creation and transfer of the video taped interviews of the CwH (Conversations with History) program at Berkeley, and its subsequent transfer to the UC/SD TV broadcast studio, which eventually broadcasts and webcasts the program. The second workflow is to model the flow of the audio transcript of the interview produced by CwH staff. Both workflows reflect the descriptive, technical, and rights metadata that needs to be created during the lifecycle of the interview to successfully manage the interview files for the long term.

3.1. First Work Flow: Capturing / Preserving Video of Interview

The interview is taped by UCB staff. The interview is then described, and the description is forwarded to UCSD-TV with a digital version of the original taping.

UCTV staff enters the description into their FileMakerPro database and augments it wherever useful. UCSD-TV also makes three versions of the original digital file. The files are an edited DV version, a MPG version, and a RealPlayer version.

UCSD-TV outputs an XML record containing preservation descriptive metadata for the interview and preservation technical metadata for each of the content files. This forms the first AIP for preservation.

3.2. Second Work Flow: Capturing / Preserving Transcript of the Video

The interview is transcribed from the video by CwH staff. CwH staff submits an rtf version of the transcript file, along with technical metadata for the file, to SDSC for inclusion in the AIP for the interview.

SDSC verifies the submission and, if all is valid, adds the transcript file and its technical metadata to the AIP for the interview.

These workflows are the basis for the following SIP and AIP models (OAIS-based Submission Information Package and Archival Information Package).

3.2.1. Submission Information Packages (SIPs):

The SIP for video data expresses in an XML non-standard format the metadata for the core production materials - an edited DV file, a MPEG broadcast file, and a RealPlayer streaming file.

There are two instances of SIPs. The first is for the digital video files metadata maintained by UCSD-TV in its FileMaker Pro database. This SIP includes the requisite descriptive metadata, technical metadata, and rights metadata in a non-standard expression for the interview captured in the corresponding content files. This SIP can be produced from the UCTV FileMaker Pro database and delivered, with the video content files, directly to the digital archive.

The second SIP is for the digital transcript of the interview, which Conversations with History staff will submit directly to SDSC. The submission typically will occur after the first SIP is submitted. However, there is no requirement that the transcript SIP be submitted within a certain period of time or that it cannot be submitted prior to the video SIP, or indeed, even that a transcript be submitted and included in the final AIP.

The transcript SIP contains the technical metadata for the transcript file and just enough descriptive metadata (e.g., UCTV program number) to insure it is properly integrated with the SIP constructed and submitted by UC/SD TV.

3.2.2. Archival information Package (AIP)

The SIPs for the metadata are captured in an AIP expressed as a METS document. The METS wrapper utilizes standard data formats. The descriptive metadata is expressed in the MODS schema, the technical metadata in the PREMIS schema , and the rights metadata in the MetsRIGHTS schema. The METS wrapper properly relates the metadata and content files and indicates which file is the original master, which file contains the broadcast version and which file contains the web (streaming) version. For the long view, the METS allows the content files and their metadata to be transmitted to other repository and more easily interoperate with other collection materials should that be desirable.

The SIP from UCSD-TV can be expressed in the basic METS schema if that is desirable for the digital repository. In that case, the CwH SIP would have to be integrated into it the METS as a PREMIS technical metadata record for the digital transcript.

At this stage, the SIPs and AIP are draft versions. There are several matters to address:

- expressing the format name, format version, and mime type accurately for each content file
- adding metadata to the SIP and / or AIP that the digital repository requires to fulfill its services
- determining if ARKs shall be used and what agency will be responsible for minting / applying them

4. Preservation

The preservation of CwH content is based on using data grid technology to manage distributed data. The Storage Resource Broker (SRB) is used as the preservation repository. A central metadata catalog (MCAT) manages preservation metadata for each video file. A dedicated MCAT instance called UCTVStudioArchive was set up. Two additional logical storage resources were registered to store digital video replicas on SAMQfs (uctv-fs1) and HPSS (hpss-sdsc). Also, SRB client software and Kepler scientific workflow software were installed on the eMac machine at UCSD-TV. Finally, a grid brick (srbbrick7) with 300GB of disk was configured for the DIGARCH project.

A grid brick [6] is a low-cost commodity disk system to store electronic records. Copies of the electronic records can still be kept on a tape archive at SDSC for minimizing risk of data loss. Grid Bricks are modular systems that are managed by data grid technology. As additional storage space is needed, additional grid bricks can be added to the data grid, and the electronic records can be automatically distributed across the new storage modules.

4.1. Kepler Scientific Workflows:

Kepler is an “executable Visio” type program, which allows you to build a workflow program by dragging and dropping components, glue them together, and execute the overall flow. Components can be grid-enabled and perform grid type operations (put, get, authenticate, monitor, report, filter, store, discover, etc.) [7].

Scientific Workflows:

- used to combine data integration, analysis, and visualization steps into larger, automated knowledge discovery pipelines and *grid workflows*
- allow the building of models of systems based on the assembly of pre-designed *components*

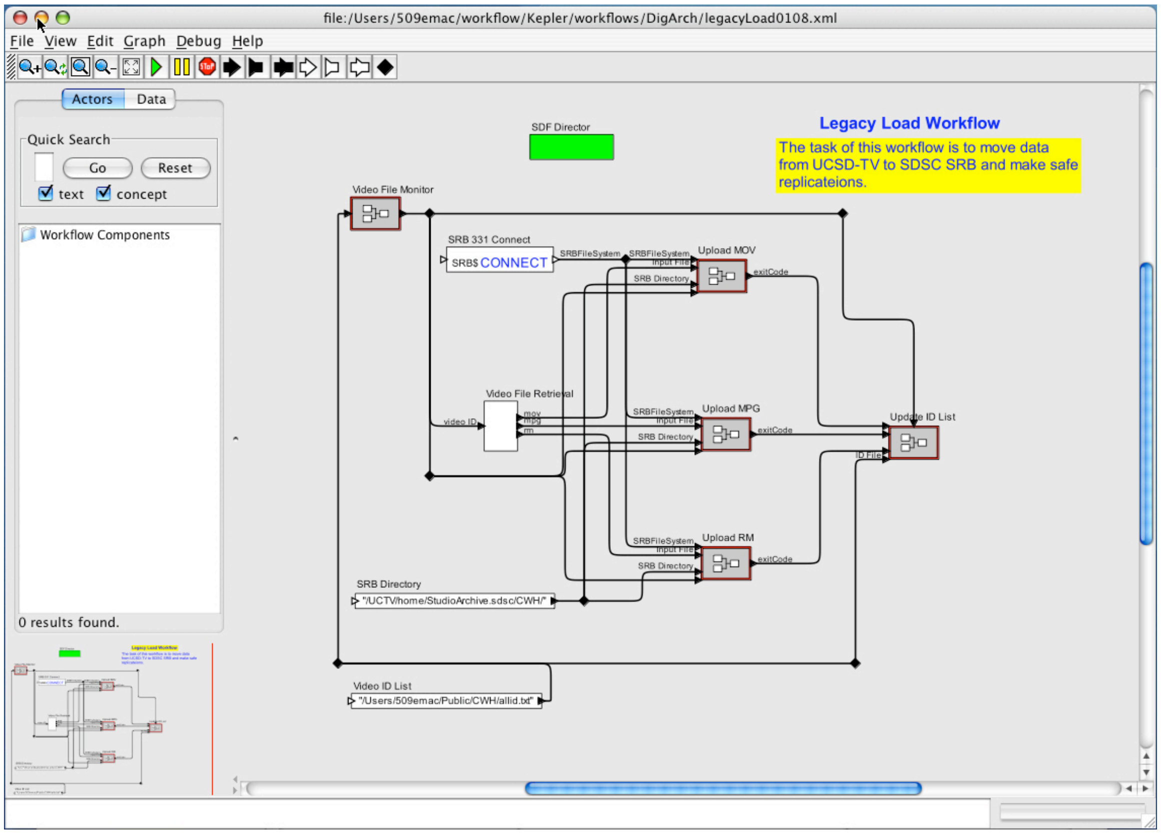
Components:

- ***actors***: encapsulation of parameterized actions performed on input data to produce output data (*parameters* configure and customize the behavior). Actors can be *simple* of *composite*
- ***ports***: actors communicate through interfaces called ports (ports are connected to one another via *channels*)
- ***director***: given an interconnection of actors, the director controls the execution

A Kepler workflow is being developed for transferring the legacy video programs from UCSD-TV to the SDSC Archive. The main transfer operation is an SRB “Sput” type operation, which accessions content into the receiving grid brick. Estimates are

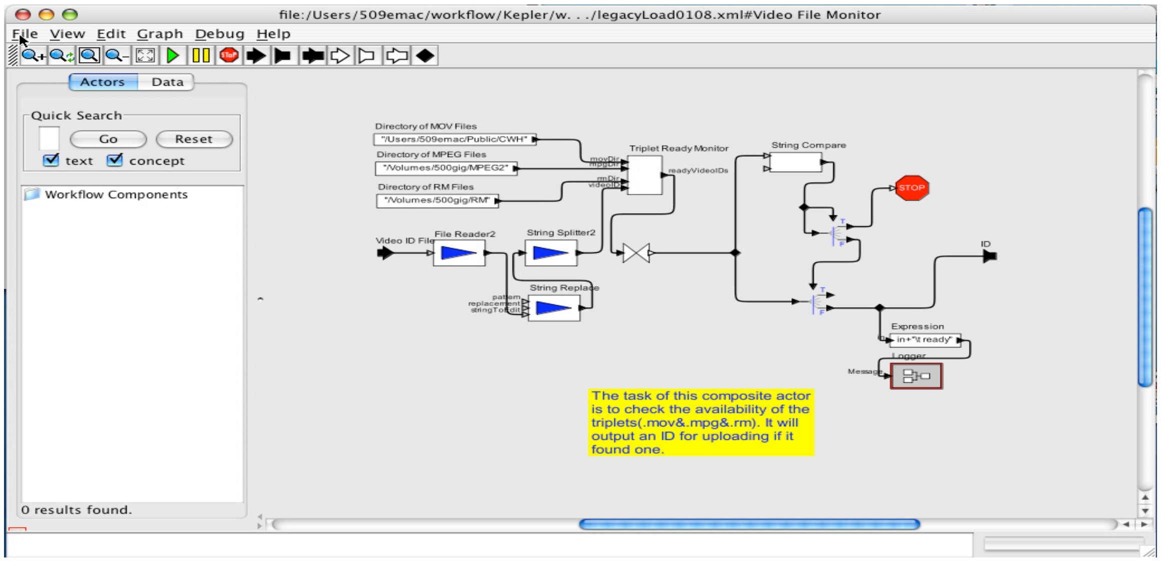
4.2. Preservation Workflows:

We have developed 4 legacy workflows for managing the preservation processes. The Legacy Load Workflow is shown in Figure 2, a Video File Monitor workflow is shown in Figure 3, and a Safe Delete Workflow is shown in Figure 4. The workflows are assembled using composite actors, allowing the creation of quite complex logic decision processes.



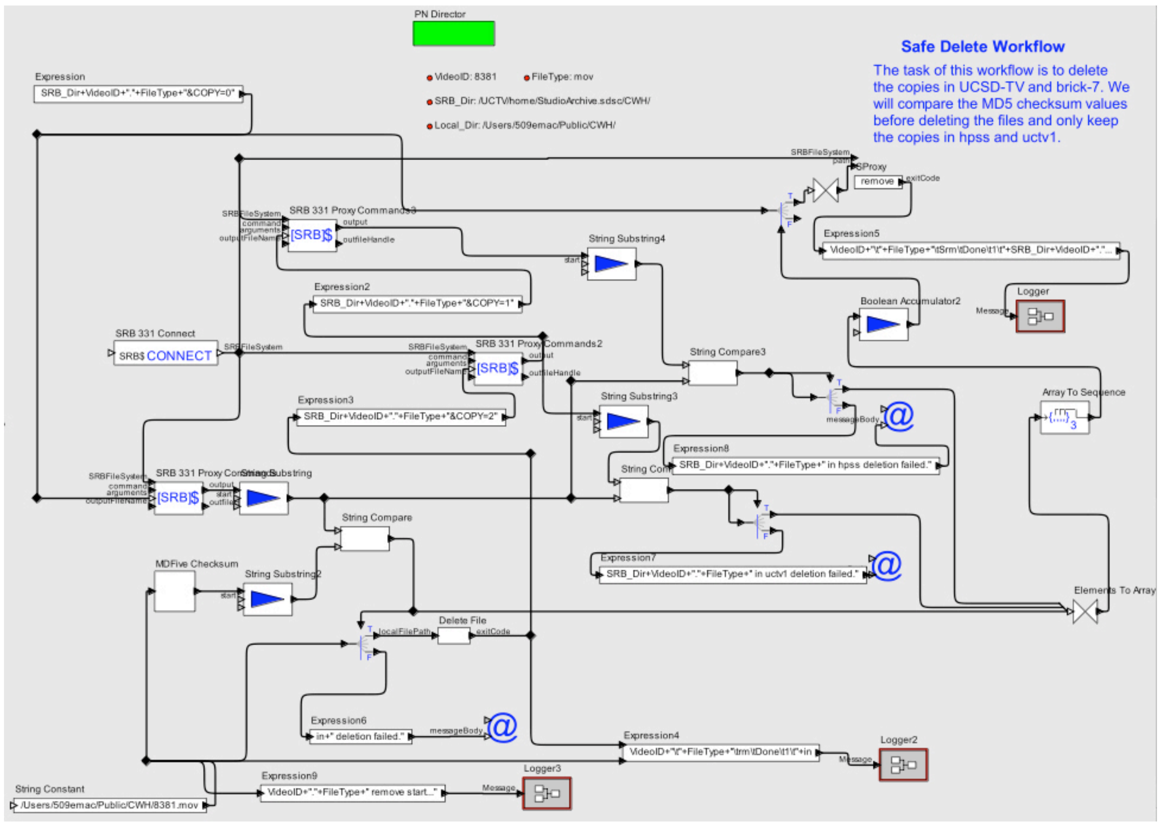
The task of this workflow is to move data from UCSD-TV to SDSC SRB and make safe replicateions.

Figure 2. Legacy Load Workflow



The task of this composite actor is to check the availability of the triplets(.mov&.mpg&.rm). It will output an ID for uploading if it found one.

Figure 3. Composite Actor for the Video File Monitor



Safe Delete Workflow

The task of this workflow is to delete the copies in UCSD-TV and brick-7. We will compare the MD5 checksum values before deleting the files and only keep the copies in hpss and uctv1.

Figure 4. Safe-Delete Workflow

5. Acknowledgements

We are working closely with Efrat Jaeger and Ilkay Altintas from the Kepler group and Lucas Gilbert from the SRB group (Jargon) and wish to thank them for their support.

6. Summary

The DigArch project integrates preservation processes on top of a SRB data grid for long-term preservation with a Kepler-based workflow system. The unique aspect of the project is the integration of the preservation processes into a production workflow.

7. References

1. ICADL 2005, The 8th International Conference on Asian Digital Libraries, December 12-15, 2005, Bangkok, Thailand, *"Digital Preservation Lifecycle Management for Multimedia Collections"*, Arcot Rajasekar, Reagan Moore, Fran Berman, Brian Schottlaender, http://www.icadl2005.ait.ac.th/program.htm
2. Kreisler, H. "Conversations With History", UC Berkeley, Institue of International Studies, http://globetrotter.berkeley.edu/conversations/
3. UCSD-TV, http://www.ucsd.tv/
4. Kepler: A System for Scientific Workflows, http://kepler-project.org/
5. SRB, Storage Resource Broker, Version 3.1, http://www.sdsc.edu/dice/srb, 2004.
6. **Data Grids, Collections and Grid Bricks**, Arcot Rajasekar, Michael Wan, Reagan Moore, George Kremenek, and Tom Guptill, *20th IEEE/ 11th NASA Goddard Conference on Mass Storage Systems & Technologies (MSST2003)* San Diego, California, April 7-10, 2003.
7. Developing Data Grid Workflows using Storage Resource Broker and Kepler, Tim Wong, UC-Davis. http://www.thwong.com/documents/SRB_Paper.doc

A Review of SRB Gridbrick Administration [*]

Geoffrey Avila
Univ. of California, San Diego
avlg@sdsc.edu

Abstract

We provide some observations about the Storage Resource Broker gridbricks as a production resource at SDSC, and speculate about future directions.

1. Introduction

We have run SRB gridbricks [2] at SDSC for nigh on four years now. Our original test systems, using the then-new 80GB disk drives and Linux 2.2 kernel software RAID, have given way to production-quality hardware using twenty-four disks of 250GB apiece, with dedicated hardware RAID done at the level of the disk controller.

2. Grid Brick Evolution

The intent of SDSC IT was to manage the gridbrick systems as we would any other server host; complete with configuration management via our in-house authorization system, configuration management via cfengine [1], a full set of mounts from our NFS servers (including user home directories, project workspaces, and third-party software), and centrally managed vendor-supplied OS patches. All of this was done to present a more homogenous environment to users and administrators. The rubrics were to be managed like any other server, save for the running SRB server process and the unusually large amount of local storage. At present, SDSC IT manages twelve SRB storage bricks for the DAKS group, with capacities between 1 and 5TB each, for a total of about 32TB in all.

In the past several years of running these systems as a production resource for the DAKS group, the performance & capacity of commodity hardware has increased tremendously. Our fourth–gen gridbricks have nearly five times the formatted capacity of the first-generation systems. This is a conservative reflection of the order-of-magnitude increase in hard disk per-spindle capacity, and the 100% increase in port capacity on high-end RAID controllers. Likewise, I/O throughput has increased fivefold by the PC industry move from PCI 2.1 to PCI-X and PCI-E. This raises the maximum capacity for a 10U rack mount system to nearly 20TB.

In addition to increases in the capacity of storage media, the current generation of SATA and SATA-2 disk drives supports many of the availability features that were previously common only to SCSI and FC-AL devices. Hot swapping defective spindles from a degraded array are one of those features that are essential to high uptime.

Outside of the configuration changes necessary for SRB use, our gridbricks must live in the same datacenter environment as any other SDSC server. This means using electricity, expelling waste heat, and taking up floor space. The commodity nature of many gridbrick components means that an aggressive packaging of system internals in not possible, but the high-end server case market is sufficiently competitive to ensure that no 6U system should be without externally accessible hot-swap disk drive pays, or triple-redundant hot-swap power supplies.

It should also be noted that the processor and motherboard for a gridbrick could be chosen to maximize the efficient use of power. Using a slower, less cache-heavy CPU, can save tens of watts. While having more memory is always nice, rarely do we find a gridbrick in normal use to be deep into swap.

At the motherboard level, the most important consideration is reliability. We buy from reputable manufacturers, and always validate the model and revision of critical subsystem components on a particular board (I/O controllers and Ethernet MACs) with our system software vendor before purchase.

3. Grid Brick Management

As our experience with gridbricks grew, we began to notice some subtle differences in the expectations that the DAKS group had for gridbrick reliability & availability, compared with the users of our other Linux systems, who

[*] This research was sponsored by the NSF SCI0438741 project on Delivering Cyberinfrastructure from Vision to Real. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, the National Archives and Records Administration, or the U.S. government.

are running edge servers or desktop workstations. We consider local disk on the latter machines to be scratch space, where if no special provision for backups has been made, local space can be overwritten or removed. The local SRB data partitions on the gridbricks are absolutely inviolate, and must be preserved at all costs. We can reboot or reinstall system software on most of our Linux servers and all of our desktops with at most a week's worth of warning to users, and often much less than that. As SRB users worldwide depend on the availability of the gridbrick-hosted data, downtime must be as short as possible and cleared weeks in advance.

The changes in system software have been occurring at a more gradual pace than the changes in hardware. Our choice of OS for all of our gridbricks has been RedHat Linux. This was a natural choice, given that the bulk of SDSC's production Linux systems are and were RedHat-based. The RH tools for installation, package management and patching were all known quantities from a sysadmin's perspective. We make heavy use of custom RedHat Kickstart for network installations, which allows us to get gigabit speeds from our install media to target systems, as well as allowing for per-host customization of disk layout and package selection.

Most of effort involved in maintaining our reference systems at SDSC is directed towards providing a secure computing environment. This doesn't just mean the absence of intruders or malicious worm software; it is the assurance that all computers under our control will behave predictably. In order to guarantee this, we employ a robust and flexible means of configuration management, a central database of all user accounts, the heavy use of secure access technology like ssh and Kerberos, and active security measures like host-based firewalling, NAT and Tripwire.

Our means of configuration management is through cfengine, a high-level language describing the on-disk state of a system. The cfagent interpreter copies files, changes permissions, makes symbolic links, removes temporary files, and runs scripts. The platform-independent and modular nature of cfengine allows for detailed yet replicable configuration of machines, from the most general of categories down to a specific host. Gridbricks are a class of systems in cfengine, a class that has some specific customizations unique to its role (such as the locations and names of the SRB data file systems), but also inherit from the class of managed hosts in general and RedHat Linux machines in particular.

SDSC account management ensures that only the users who need to have access to a system are able to log in. The system for distributing the /etc/passwd file is centrally managed, with remote updates to ensure that only our controlling authority can allow individuals access to a system.

Patch early and patch often, is IT industry best practice from a security standpoint. The Linux install on the gridbricks is liable to almost all of the kernel and userland vulnerabilities common to such software. RedHat, in the previous few years has moved from rolling out patches as they are released to a quarterly schedule for noncritical patches to software. Critical security patches can be and often are released at a frequency greater than four times a year. These patches often require a system restart, forcing us to choose between running unpatched systems and compromising the availability of SRB data.

To avoid this, we have made several changes to the software configuration on several of the gridbricks, and, should it be successful, we shall make it the standard for all such systems.

For starters, we only NFS mount a single file system for the purpose of running configuration management tools. While cfengine can be started and run as a daemon, the regular nightly per-machine run mounts, reads from, and then dismounts the filesystem containing the configuration data. We have greatly reduced the number of users with login privileges on the gridbricks, cognizant of the fact that most critical vulnerabilities found on Linux systems are "local" exploits, which can only be employed by someone who already has a valid user account. IT has made a great investment in hardware One Time Password devices, which we will issue to users who need shell access to the gridbricks.

We are employing tcpd and considering the use of ipfilter on gridbricks, especially ones that may need to live outside of the predictable SDSC network environment. Restricting the number of services run on a gridbrick allows us to trim potential routes of access down to a minimum. Much of the time, the only two services that need to be both running and visible to the outside network is the SRB server itself, and sshd for remote access.

Every gridbrick IT has maintained since our original proof-of-concept has deprecated software RAID in favor of dedicated hardware RAID, in the form of 3ware Escalade controllers. These controllers require a driver specific to the running Linux kernel version, as well as a matching on-card firmware specific to the driver version, and a utility (web-based or CLI) that schedules maintenance and alerts administrators of possible problems.

3ware maintains the driver themselves, often rolling in bug fixes ahead of the driver version in the currently shipping kernel. It is important to keep the version numbers of 3ware drivers/firmware/utilities in sync, something that can easily be undone by upgrading to a new kernel that may not have the latest vendor driver included. The symptoms of a mismatch between the driver and the management utility can at first resemble a bad disk or array controller, and can lead to I/O timeouts or file system corruption. Proper installation of the driver, firmware and utility programs through cfengine has been critical to maintaining stability.

The default file system as used by RedHat can also lead to difficulties. For ext3 on our RedHat Linux 2.4 servers, the maximum file system size is 4TB; already too small for the largest possible combinations of disks and controllers. The selection of different file system types, while possible, is discouraged. RedHat will only support ext3 without added cost when using their Enterprise v.3 product.

We have tested the included, but as-yet unsupported ReiserFS 3.x file system on some of the gridbricks, with mixed results. ReiserFS seems to recover from an unexpected shutdown instantaneously, far better than the 72-hour fsck one can expect when using ext3.

The downside is that ReiserFS is much less tolerant of misconfigured hardware than ext3. Should a RAID controller show too much latency on writes, file system metadata can be irretrievably lost. IT has since begun a move back away from ReiserFS, as it looks to be removed completely from shipping versions of RedHat Linux.

While it is not usually necessary to have conventional disaster-recovery or archival backups scheduled for data stored on gridbricks (the SRB software handles this function internally by replication), it is sometimes necessary to do bulk data transfer when SRB is not available. In these cases, we have discovered that many tools, like tar, cpio, or dump, have limitations on file size or name length than can make the transfer of data off gridbricks outside of an SRB session troublesome. For instance, both the POSIX and GNU *tar* utilities have hard limits of 256 characters or less in the pathname of file to be stored in an archive, and a limit of 8GB on the final size of an archive[3]. These limits are not usually reached on desktop systems, but can be hit when dealing with SRB data volumes with tens of thousands of directories and gridbricks with hundreds of gigabytes of space to put them in.

At present, SDSC IT is evaluating new system software and hardware for future gridbrick systems. We are looking at new RAID controllers from LSI Logic and Adaptec. The RAID-6 capability of many new controllers, where an additional parity block is added to the array, allows for the simultaneous failure of two spindles without the loss of data. On a similar note, IT is also testing Sun's new Solaris 10 OS and ZFS file system. The design of ZFS allows for RAID volumes that can be interrupted during parity write, but not lose stripe data[4].

We would also like to examine possibilities for storage clustering through the use of network block device like functionality, which has been mature on Linux and certain BSD variants for some time now. While there are limits to the number of disks one can enclose in even the largest single PC case, it should be possible to merge via a dedicated high-performance network a vast cluster of gridbricks, up to petabytes in size. A builder would be limited not so much by the brick hardware itself, but by the speed of his or her network and the environmental and electrical capacity of the datacenter.

4. References

[1] M. Burgess and R. Ralston, Strategies for Distributed Resource Administration Using Cfengine, Software-Practice and Experience 27, 1083 (1997)

[2] Data Grids, Collections & Grid Bricks, Arcot Rajasekar, Michael Wan, Reagan Moore, George Kremenek, and Tom Guptill, 20th IEEE/ 11th NASA Goddard Conference on Mass Storage Systems & Technologies (MSST2003) San Diego CA, April 7-10th, 2003 pp.5-8

[3] *tar*(8) manual page

[4] Sun Microsystems, ZFS Administration Guide, Sun Part#817-2271-2, January 2006, pp.17

Performance Optimization of SRB Hardware Configurations

Peter Ashford
Ashford Computer Consulting Service
ashford@accs.com

Abstract

The configurations required for large production installations of the Storage Resource Broker (SRB) are discussed. Small installations can use scaled-down versions of the configurations that will be discussed.

1. About the author

Peter Ashford has been working in the computer field since 1979. He currently builds custom computer systems based on off-the-shelf commodity components. He has worked for Cray Research supporting both hardware and software maintenance activities at multiple customer sites. He has been involved with every phase of the software life-cycle from initial concept to termination.

ACCS has supplied the SRB project with ten SRB servers and a test SRB MES. In addition, ACCS has supplied several other projects and organizations with SRB servers. ACCS also supplies custom systems to other organizations for use as compute servers, file servers and high-performance workstations.

2. SRB overview

SRB consists of three components. There's an SRB client, an SRB server, and an MCAT-Enabled SRB Server (MES). The first step in optimizing performance is that the MES not be used to store datasets, but only to store and work with the MCAT database. The rest of this paper will assume that the MES is not used to store SRB datasets.

The client communicates with both the server and the MES. When processing user queries, the MES and server each communicate with the client, and with each other. In addition, the MES communicates to a database engine, where the actual query processing is performed.

Due to the limited abilities of current performance analysis tools under Linux, this paper discusses the requirements in fairly general terms.

3. Redundancy

The SRB is used to perform many functions. All of these functions rely on the ability of SRB to store data, and retrieve it in the original form. This can only be done reliably if redundancy is used. For maximum availability and accuracy of production data, some thought needs to be put into redundancy of the systems and the data. In a typical large installation, there would be one or two MES systems, and several server systems. Due to the different tasks that the SRB systems are required to perform, different levels of redundancy are required.

Due to the low number of MES systems, they should have a high level of internal redundancy, in order to provide maximum availability of user data. An MES should have hot-swap redundant power supplies, and all file-systems on it should reside on redundant arrays.

Since the SRB architecture allows a dataset to reside on multiple SRB servers, the redundancy in those servers need not be as high as on the MES to maintain excellent data availability. The power supply and the disk containing the operating system don't need to be redundant. Only the stored data needs to have redundancy. This is not so much to keep from losing the data, but to reduce the chance of having to recover the data from another server. Even with a Gigabit Ethernet network, transferring the multiple terabytes that often reside on an SRB server can require a significant amount of time and network bandwidth (at least 2.5 hours per Terabyte).

When redundant disks are used, a hot-spare is suggested. This will decrease the window of opportunity where a second disk failure can cause loss of data.

To reduce down-time of the systems, hot-swap disks are strongly suggested for any disk.

All SRB servers and MESs should use ECC memory. This is to reduce the impact of the occasional memory errors that will occur.

4. Client hardware requirements

The user communication with the SRB is through a client system. The hardware requirements for this system are minimal. The system must have a network connection, be able to run the user interface software, and be able to perform any processing of the data that is required by the user. The hardware requirements for the user interfaces are similar to the hardware requirements for the web browser interface.

5. Server hardware requirements

An SRB server receives file requests from a client or an MES, reads the requested file from disk, and returns the file. The type of system that the activities on an SRB server most closely resemble is a file server. We will discuss this architecture in this section.

In an SRB implementation, we're interested in how quickly an SRB server can deliver stored content to clients. Several factors limit the ability of an SRB server to deliver content to clients. These include the I/O bus, network, CPU and disks.

When a single-processor motherboard is used in a file-server, the PCI bus often limits the rate at which content can be delivered to clients. A 32-bit/33MHz PCI bus has an effective bandwidth of approximately 100MB/S. This bandwidth is usually shared between the network interface and the disk controller, giving a limit of approximately 50MB/S of delivered content. A system with a PCI-X bus for the disk controller and network will have a total effective PCI-X bandwidth of at least 400MB/S (single 64-bit/66MHz PCI-X bus), and as high as 1600MB/S (dual 64-bit/133MHz PCI-X buses). Even higher bandwidths are available when a PCI-Express bus is used, but the extra bandwidth is rarely required.

Another limit to the rate at which content can be delivered is the network. Currently available file servers normally have a single Gigabit Ethernet connection which limits overall content delivery to approximately 100MB/S. If a higher rate of content delivery is required, it is possible to use a faster network connection, and hope that the network infrastructure can keep up, or to use multiple systems to deliver the content. Single transfers will not be able to use the full network bandwidth, due to network latencies. To saturate a Gigabit Ethernet network connection will normally require at least three concurrent transfers. Higher bandwidth connections will require additional connections to saturate them.

Most of the system memory will be used to buffer disk data. Due to the way SRB data is retrieved and used, there will normally be little or no reuse of data in the system buffers. This gives us a relatively small memory requirement. In general, 1GB of memory is adequate for an SRB server. Larger memory (up to 2GB) may be of use if high transfer rates are required.

For maximum performance, the CPU and disk subsystem need only be fast enough to keep up with the network. The principal usage of CPU cycles is TCP/IP encapsulation, which can be done only as fast as memory can deliver data to the CPU. Because SRB uses packet sizes of up to 1MB, a cache of less than 2MB will have little impact on peak network performance. If the cache size is 4MB or larger, the encapsulation process can be performed entirely within cache, significantly speeding up the process.

In addition, when the data is encrypted prior to transfer, additional CPU time is required. The encryption process is almost as fast as the encapsulation process, as the CPU cache provides a significant speedup to the process, allowing it to run at CPU speed, instead of at memory speed.

It should be noted that any modern high-performance CPU will be able to encapsulate data at the speeds required for a single Gigabit Ethernet connection, requiring no special processing. When more network bandwidth is available, or where a large portion of the data is encrypted prior to transfer, it might be necessary to use special processors, or multiple processors, to maximize data delivery.

The disk sub-system is the most likely real limit that will be encountered. When multiple transfers are being performed, the disk subsystem must seek between the individual datasets. With small datasets, seeks are likely to limit the overall throughput. For large datasets, the SRB server software reads in enough data in a single read to counteract the seek time.

In order to provide reliable data storage, SRB server system data should be stored in a redundant manner. The I/O pattern (large I/Os, mostly read) on an SRB server system is well suited to use with RAID levels 5 and 6. To decrease the interference between multiple transfers, it may be useful to have multiple arrays in the system.

Currently available SATA disks are fast enough to provide reasonable throughput on an SRB server. The reliability of the latest data-center versions of the SATA disks is similar to the reliability of the latest SCSI disks. Specifically, the Seagate NL35 series and the Western Digital RAID Edition series both have a 5-year warranty. In addition, both of the above series are specified with an MTBF of at least 1 million hours, and an error rate of 1 per 10^{14} bits. This can be compared to an error rate of 1 per 10^{15} bits for the newest SCSI and Fiber Channel disks, and the Western Digital Raptor series of SATA disks.

Although the SATA drives are reliable, the SATA interconnects aren't. The standard SATA connector is easily pulled out of it's socket. It can come loose during shipping, while the system is being worked on, and it can even come loose due to the system vibrations. This problem can be corrected by using MultiLane backplanes and cables. The connector on the end of a MultiLane cable has a positive latching mechanism, so that the cable won't come loose easily. The MultiLane cable also carries the signals for four disks, reducing the cable clutter within the system.

6. MES hardware requirements

An SRB MES receives SRB queries from a client, locates the appropriate matches in the SRB servers that it controls, and sends that metadata, including the file name and SRB server identification, to the client. The type of system that the activities on an MES most closely resemble is a database server, and is the architecture that will be discussed here.

In an SRB implementation, we're interested in the number of queries that can be satisfied in a given period of time. Since an SRB MES typically controls many SRB servers, it is possible to have a significant number of queries coming into an SRB MES. An SRB query, like any database query, can be simple to satisfy, or can require a large amount of work to satisfy. In order to satisfy these queries in a reasonable period of time, the SRB MES must be able to respond to the average complex query in about a second. If this level of response can be achieved, and there are no undue delays in the SRB server, the SRB system as a whole will appear to the user to be very responsive. Any delay in the ability of an MES to generate a response will translate directly to a delay in the responsiveness of the SRB system as a whole.

Like other database servers, there is a significant memory requirement for caching the database. With a 32-bit CPU, the limit to the memory size of the database engine is normally 3GB. When 64-bit CPUs are used, the memory that can be used by the database engine expands significantly. Fortunately, the database engine used by the SRB rarely requires more than 1GB of memory for optimum performance.

When a query is searching the contents of the metadata for a matching string, a large amount of CPU power is required. Unfortunately, the maximum performance of a CPU in searching through data is limited by the memory interface bus, as the CPU has virtually no reuse of the searched data before the cache is flushed. For this reason, multi-CPU systems, with the fastest possible memory interconnect, are appropriate for an SRB MES. Multi-core processors are of marginal use here, as the memory bandwidth is not increased with the second core.

When choosing a CPU architecture, there will be some temptation to use a 64-bit processor, as the database benchmarks are usually significantly better than with 32-bit processors. Unfortunately, some of the code in the 64-bit releases of Linux isn't 64-bit clean yet. This situation is currently improving, and should be resolved shortly. In addition, the MES hasn't yet been tested on either a 64-bit Linux of 64-bit Windows platform. What this boils down to is that although it's safe to use a 64-bit commodity processor, it's not yet safe to run them in 64-bit mode as an MES.

Like all database servers, an SRB MES will respond well to a fast disk subsystem. A large portion of the disk I/O consists of small random reads. Unfortunately, there is also a significant amount of small random write activity. When choosing the RAID level for the disk subsystem, it is the write I/O pattern that determines the best possible choice. In this case, RAID-10 is best for storage of the MES database and temporary files. Increasing the number of disks in the RAID array will give a performance improvement. For the operating system and applications, RAID-1 is the appropriate choice. For the database transaction log files, RAID-5 may be used, but due to the low storage requirements for these files, RAID-1 would also be a good choice. Due to the I/O pattern, the use of RAID-5 or RAID-6 to store the MES database will have a significant performance penalty.

In order to maximize the benefit of a good disk configuration, fast disks should also be used. The Western Digital Raptor series of SATA disks (10,000 RPM) is an economical choice, but the 15,000 RPM SCSI or Fiber Channel disks will perform significantly better, especially in large installations.

When using a commercial DBMS with an MES, it might be necessary, for fiscal reasons, to have the database reside on a different system than the MES. If this is the case, the CPU and memory requirement will be slightly lessened, and the disk sub-system need not be as fast. Additionally, it might be useful to use a dedicated network interface on the MES for communications to the DBMS system.

7. Lifespan

Experience has shown that a computer system built from commodity components has a maintainable lifespan of between three and five years. This is because compatible replacement components are typically only available for that period of time. Recent changes in the market are tending towards shorter availability times for CPUs and motherboards and longer availability times for disks.

Budgets should allow for replacement of systems on a schedule similar to the above mentioned component availability.

When low-quality components are used in a computer, the lifespan of the computer is likely to be shorter. An example of this is that a few years ago, several motherboard manufacturers changed the brand of capacitor used in the CPU voltage regulator. The result was motherboards with a life expectancy that was significantly shorter than normal, but at a slightly lower price.

There are three environmental factors that can help to maximize the lifespan of a computer. These are: clean air, cool air, and good power. If the air coming into a computer isn't clean, it will cause dust buildup on the components. This dust will insulate the components, and increase their operating temperature. In general, the dust level in the average office is high enough to cause problems within six months to a year.

Cool air is also useful for extending the life of a server. If the air temperature entering a computer climbs, the temperature of all the components in the computer also climbs. The rule of thumb is that for every five degrees that the temperature of a computer climbs above optimal (normally 70-75 degrees), the life expectancy of that computer will be cut in half. Lowering the intake air temperature below optimal will have little or no impact on the life expectancy of a system with reasonable airflow.

Good power is devoid of surges, spikes, brownouts and dropouts. These can all be removed by a good UPS. The reason that these power issues are bad for the lifespan of a computer is that they stress the power supply. In addition, power surges and spikes can cause a short-term increase in the voltage coming from a power supply. This increase will stress the components of the computer, shortening its lifespan.

8. Summary

In summary, an SRB server may be configured with one CPU, small memory (1GB) and a Gigabit Ethernet network interface. The data storage should be in a redundant array (RAID-5 or RAID-6), preferably with a hot-spare. Larger numbers of disks will improve the performance. Multiple arrays may be used for additional storage space, and improved performance.

An SRB MES should be configured with multiple CPUs, large memory (2-4GB), and a Gigabit Ethernet network interface. The database storage should be in a redundant array (RAID-10), preferably with a hot-spare. Larger numbers of disks will improve performance.

SRB Portlet Development for the Grid Portals[*]

Mary P. Thomas
San Diego State Univ.
mthomas@sciences.sdsu.edu

Tarun Bansal
San Diego State Univ.
tbansal@rohan.sdsu.edu

Tushar Gupta
San Diego State Univ.
tgupta@rohan.sdsu.edu

Akhil Seth
Univ. of Texas at Austin
akhil@tacc.utexas.edu

Dave Thomas
San Diego State Univ.
dthomas@rohan.sdsu.edu

Abstract

This paper describes the design and architecture of a set of simple portlets that interface to the SRB services using the Jargon library and the GridPort toolkit, and the GridSphere framework. The client side interface has been enhanced with AJAX architecture. These portlets have been designed to use either SRB account and passwords, or the GSI credentials. They have been demonstrated to work on both TeraGrid and DOE Fusion Grid SRB Collections.

1. Introduction

Portals are well-established as useful interfaces to complex, distributed services and are in use by both commercial and scientific applications [MPT-1]. More recently, they have emerged as important components in large scientific applications known as Science Gateways [TG]. Science Gateways enable entire communities of users with a common scientific goal to use grids (such as the TeraGrid) through a common interface and sharing common information. It is not uncommon for a science portal to provide access to resource information, job submission and control, help users build job control scripts, and provide access to files and to view or visualize data. As the resources, services, and computation on grids increases, the need to organize, classify, search, and store in a distributed manner becomes an important aspect of most projects.

The Storage Resource Broker (SRB) project from the San Diego Supercomputer Center (SDSC) is a system of services that integrate diverse storage resources (e.g. databases, workstations, archival and NFS systems) behind a virtualized interface [SRB]. Users can customize the metadata used to query and search their collections. Portals that interface to data collections via services such as SRB have the potential to reduce the complexities associated with data investigations and hence increase productivity and information distribution and collaboration.

The architecture, design, and coding examples of a simple portlet that interfaces to the SRB services are described in the sections below. The client side interface has been enhanced with AJAX architecture. These portlets, based on the Grid Portals Toolkit (GridPort) [GP4], have been designed to use either SRB account and passwords, or the GSI credentials. They have been demonstrated to work on both TeraGrid and DOE Fusion Grid SRB Collections.

2. Architecture

Portals are deployed on science grids using Java, Perl, Python and other languages. The Java portlet framework [PB] has emerged as a key development environment for grid portals for several reasons. It is component based, so that user interfaces and experiences can be customized and web pages can be composed of several portlets. The portlet components can be shared among projects (since portlets are standardized). The portlet component architecture maps well to the services oriented architecture (SOA) on which the current grid and Web are being based. Additionally, portlet frameworks allow developers to utilize the large number of existing libraries, tools, and services being developed in the commercial, open source world, which enhances the capabilities of a portal.

The SDSU SRB Portlets discussed in this paper are written using the GridPort Toolkit (GridSphere Framework), and follow the classic portlet lifecycle (initialization, handling of requests/actions, rendering a

[*] This research was sponsored by the National Science Foundation (grant #12345) and the Department of Energy (Grant #12345). Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of the Government, or any person or agency connected with them

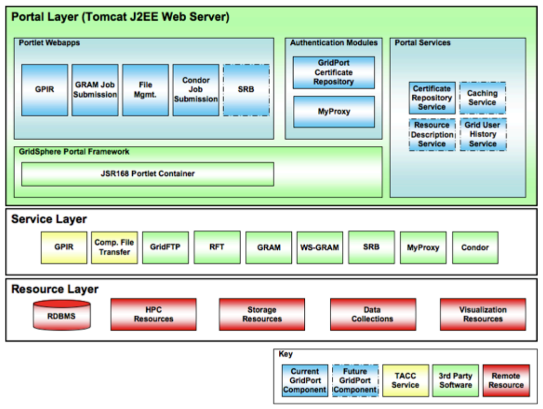


Figure 1. GridPort 4 Toolkit diagram depicting the layered architecture and component services approach.

view, destruction) and employ the JSP (Java Server Pages) Portlets pattern [JSP-1]. AJAX is used to obtain local updates to the user interface (no new data/remote service required) and to help to reduce web page update lifecycles.

2.1. SRB Technologies

The current set of portlets are based on the NMI R5 release, which contains versions of the Globus Toolkit and the SRB client software that are interoperable. The reasons driving this choice are discussed in the 'Challenges' section. The portlets use the SRB Jargon API [JAR] which interfaces to the SRB services using either SRB authentication or Grid/GSI.

2.2. GridPort Toolkit

The SDSU SRB portlets are designed to work within the GridPort Toolkit [GP4], which is a major component of the NMI OGCE software project [OGCE]. The vision for GridPort 4 is a software package that includes core portlets, application specific portlets, and portal services (see Figure 1). GridPort is intended for use by developers of grid-enabled portals, portlets, and applications. GridPort is based on the GridSphere framework, and hence inherits all of the GridSphere capabilities as well [GS]. In GridPort, the portlets expose backend services via customizable web interfaces that enable personalization of grid portal user interfaces. GridPort has 3 basic layers: a Portal Layer, a Service Layer, and a Resource Layer.

In the *Portal Layer*, there are core Webapp portlets, which are interfaces to core grid technologies such as Globus, Condor, SRB, GPIR, etc. Application specific portlets provide interfaces to specific applications such as Fusion Grid, Gaussian, NWChem, and NAMD.

The *Portal Services Layer* is intended to be the connective glue between the portlets and resources, and help to increase cohesion between the portlets. Internal portal services support the portlets at the portal layer by augmenting their capabilities in an extensible and reusable way while tying the portlets together in order to make them more cohesive. The GridPort Information Repository(GPIR) service is an example of the connective glue. This is a Web service that allows content/information providers to insert data into a database, and web clients to perform queries and pull data from the service.

The *Resources Layer* consists of both hardware (clusters, databases, etc.) and remote grid services such as the GPIR, SRB or GRAM services. Some of these services are also standard Web services, etc.

2.3. SRB Portlet Security

GridPort maps portal user accounts to their grid credentials by using the Open Grid Computing Environments (OGCE) proxy credential management service, which retrieves proxy credentials from a MyProxy service. The internal service then makes that credential available to all portlets.

The Jargon API includes a GSI authentication mechanism. The SRB portlet code pulls the credential and passes this information to the appropriate Jargon method. Optionally, Jargon can be used to authenticate SRB users via the proprietary SRB authentication scheme.

2.4. AJAX

The SDSU SRB Portlets follow a modified Model-View-Controller (MVC) architecture. The use of AJAX modifies the View aspect of MVC. AJAX stands for Asynchronous JavaScript and XML [AJAX]. It has an advantage over standard HTML because it allows information to be transmitted dynamically between the client and the server without having to refresh the page.

Standard HTML sends a request to the server, which processes the request and either returns a static HTML page or sends the request to a framework (such as a portlet), or another service. The request is processed, returned to the server, which creates and returns an HTML page for the browser to render. When the client has to retrieve new data from the server the browser has to reload another HTML file. This can often take time if the network is slow, the server is overloaded, or the web page is date intensive. Often, only a small portion of the new HTML page varies.

AJAX is not based on new or novel technologies, but is working because of advances and standardization in browsers, servers, networks, services, etc. AJAX uses

XHTML for the data presentation of the view layer, DOM (Document Object Model) to dynamically manipulate the presentation, XML for data exchange, and XMLHttpRequest to create a direct connection between the browser and the Web server. With AJAX, the web pages do not need to be updated, just some of the data on the page. A powerful example of an AJAX application is the Google Map Search engine page [GM].

[1] SRB Portlet Design & Operation

In this section, the life cycle of the SDSU SRB Portlets for the case of file upload is described, including code examples to help clarify the operational states of the portal. Presenting an in-depth discussion of how portlets work is outside the scope of this paper, however a textbook reference on portlets for the interested reader is [JP]. In this scenario, we assume that the client has logged in, authenticated to the grid (obtained a credential), has authenticated to the SRB server (e.g. the TeraGrid), and is currently sitting in some collection or directory. Via the browser interface (view), the client chooses "File Upload."

Typically, when a new function is selected, a request to the Web server will result in a new web page being downloaded. However, with the use of AJAX, in the SDSU SRB Portlets, this type of interaction is minimized to only occur when the data being displayed requires an update. We use AJAX to update and retrieve new form elements and additional Web page display information (text, icons, etc). Because AJAX is still a "new" approach, we chose to conservatively utilize this feature, and hence limited the scope to updating simple, common, web page elements.

```
function fileUploadForm(relativePath)   {
  var url = '/srb/AJAXServlet?relativePath=' +  relativePath +
      "&requestAction=fileUploadForm&formAction=" +
     document.dummyForm.action;
     loadURL(url);
 }
```

Next, the request is sent to the AJAX servlet service running on a Web server:

```
function loadURL(url)    {
// branch for native XMLHttpRequest object
if (window.XMLHttpRequest)   {
  req = new XMLHttpRequest();
  req.onreadystatechange = processRequest;
  req.open("GET", url, true);
  req.send(null);
  // branch for IE/Windows ActiveX version
  }   else if (window.ActiveXObject)   {
      req = new ActiveXObject("Microsoft.XMLHTTP");
      if (req)  {
          req.onreadystatechange = processRequest;
          req.open("GET", url, true);
          req.send();
 } } }
```

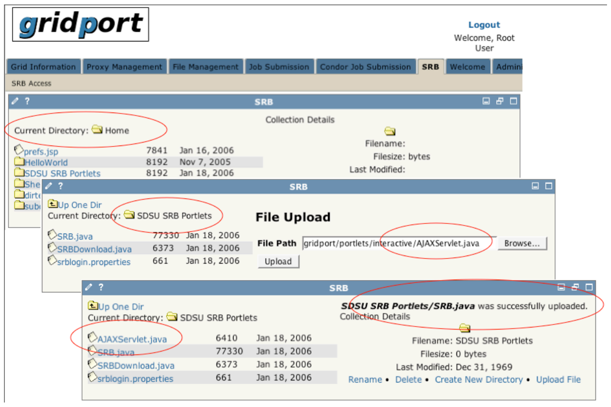
Inside the AJAXServlet class, the request for 'fileUploadForm' will generate the required HTML to be returned and to appear on the client's Web page. Since the Web page on the browser does not need to be reloaded, the HTML Form elements appear to be written as if from local cache:

```
public class AJAXServlet extends HttpServlet    {
public void doGet(HttpServletRequest req,
   HttpServletResponse resp)
   throws ServletException, IOException    {
 …..
 if ( action.equals("fileUploadForm") )   {
    res.getOutputStream().println("<h2>File Upload</h2>");
    res.getOutputStream().println("<form
        name=\"fileUploadForm\" action=\"" +
        formAction + "&gs_action=&requestAction=fileUpload\"
        enctype=\"multipart/form-data\" method=\"post\">");
    res.getOutputStream().println("File:  <input type=\"file\"
        name=\"fileUploadPath\" size=\"40\">");
    res.getOutputStream().println(" <input
        name=\"submitUpload\" type=\"submit\"
        value=\"Upload\">");
    res.getOutputStream().println(" <input type=\"hidden\"
        name=\"relativePath\" value=\""+ relativePath + "\">");
    res.getOutputStream().println("</form>");
 }
}
```

The "File Upload" view is a familiar interface to all web browser users. Once the client has entered the file name (typically by selecting the 'Browse' button which allows file selection from the users machine), he hits the 'Submit' Button. The Request is sent to the Web server, which registers the event. The event causes the portlet to invoke its ProcessAction methods (in MVC model, this Control). Within this method all of the Request data is filtered and interpreted and a switch block is used to invoke the correct action and set the mode for the next request:

```
public void processAction(ActionRequest request,
         ActionResponse response)
         throws PortletException, IOException
{
  getPrefs(request);     //get users SRB prefs
  PortletSession session = request.getPortletSession(true);
  String requestAction  =
      request.getParameter("requestAction");
  If ( requestAction.equals("fileUpload") )  {
     try {
            doFileUpload(request, relativePath);
     }
     catch(Exception e)  {   …   }
     String relativePath =
           (String)session.getAttribute("relativePath");
     session.setAttribute("requestAction", "afterFileUpload",
           PortletSession.APPLICATION_SCOPE);
     response.setPortletMode(PortletMode.VIEW);
    }
}
```

In the method, doFileUpload, the basic filenames and paths are set up. The file is uploaded from the clients'



Collection Details
Current Directory: Home

```
prefs.jsp             7841   Jan 16, 2006
HelloWorld            8192   Nov 7, 2005
SDSU SRB Portlets     8192   Jan 18, 2006
```

Filename:
Filesize: bytes
Last Modified:

Current Directory: SDSU SRB Portlets

```
SRB.java              77330  Jan 18, 2006
SRBDownload.java      6373   Jan 18, 2006
srblogin.properties   661    Jan 18, 2006
```

File Path: gridport/portlets/interactive/AJAXServlet.java

SDSU SRB Portlets/SRB.java was successfully uploaded.
Collection Details

Current Directory: SDSU SRB Portlets

```
AJAXServlet.java      6410   Jan 18, 2006
SRB.java              77330  Jan 18, 2006
SRBDownload.java      6373   Jan 18, 2006
srblogin.properties   661    Jan 18, 2006
```

Filename: SDSU SRB Portlets
Filesize: 0 bytes
Last Modified: Dec 31, 1969

Figure 2. Composite image of the SDSU SRB Portlets demonstrating Directory Navigation and File Upload from local client to remote SRB server.

browser when a message is sent to a GridPort common interface method called FileUploadManager, which gets and stores the file on the local server:

```
private void doFileUpload( ActionRequest request,
   String relativePath) throws Exception
{
  // code to set up paths and file names appears here
  String portalFullFileName =
     FileUploadManager.uploadFileToPortal(request);
  ….
  /*  upload to srb server */
  boolean actionResult = copyFrom(absolutePath,
        portalFullFileName);
}  /* End doFileUpload
```

Finally, doFileUpload sends a message to the copyFrom method, which will copy the file from the local web server system to the remote SRB server using the Jargon method, copyFrom(file):

```
private boolean copyFrom(String srbAbsolutePath,
               String localFileName)
{
   try   {
      SRBFileSystem srbFileSystem = new
              SRBFileSystem(srbLogin());
      SRBFile srbFile =  new SRBFile(srbFileSystem,
              srbHome + "/" + srbAbsolutePath);
      LocalFile file = new LocalFile(localFileName);
      srbFile.copyFrom(file);
      file.delete();
      return true;
  }    catch (IOException e)  {  … }
  //clean up
} // END copyFrom()
```

Once the file has been transferred to the remote SRB server, new data for the display must be obtained and so a new view of the web page is generated. This is invoked because the 'afterFileUpload' action was set earlier.

2.5. Portlet Status

Currently, the SDSU SRB portlets allow users to perform the following SRB S* commands and functions:

- Collection Browsing (list, change directory)
- File manipulation (rename, delete, copy).
- Directory manipulation (change directory, create new dir, remove directory.
- View file/document contents
- File upload from local host to SRB server
- File download from SRB server to local host
- Third party file transfer between SRB collection or FTP servers.

2.6. Installation

The SDSU SRB Portlets are part of the GridPort Toolkit which uses the GridSphere framework. It can be downloaded from the GridPort website [GP4]. The download can either be a full GridPort demo portal, or a separate WAR file that you install into an existing portal. The Jargon & GSI jar files can be pulled from SDSC [JAR]. Since GridPort uses Maven to build projects, the files will be pulled automatically. To run the portlets you fire up the portal, obtain a valid credential for the portal (run the MyProxy portlet), and set the SRB user preferences (which are saved for later sessions).

2.7. Challenges

Because there are so many versions of the Globus Toolkit and the SRB servers and clients, installation and configuration of clients and servers tended to be empirical and lengthy (note: we did get very good support from the SRB team). The portals pull certificates from MyProxy credential services, and those certificates are based on a certain version of the Globus Toolkit. The SRB servers run a particular version of the SRB server, and the server may or may not be GSI enabled. If a project installs the SRB and then changes to GSI versions, new SRB servers need to be hosted, which changes the port configuration, etc. If a portal is accessing multiple SRB's, then managing the configuration data can be complex. In addition, the most current versions of SRB hosted on the project website is not necessarily compatible with your grid. As a result, we chose to install the latest NMI stable release, which included SRB client, and once we did this, everything worked fine. Although not required to install the portal, we do recommend that developers install both Globus and SRB clients, and to develop test scripts to confirm that basic functions work, that certificates are valid, etc.

3. Conclusions & Future Work

A general, simple SRB portlet has been developed which provides basic functionality (collection navigation, file upload/download/transfer, create/deletion, etc.) to any SRB server. These portlets work on several grids, including the TeraGrid and the FusionGrid, and use either the SRB or GSI authentication mechanisms. These portlets will be integrated into the NMI Open Grid Computing Environments (OGCE) project [OGCE].

Future plans include an admin interface and inclusion of more S* commands, as well as an MCAT browser. These portlets will be released in Feb, 2006 as part of the GridPortal Toolkit, which is part of the OGCE NMI release and is scheduled for deployment on the Fusion Data Grid project [MPT-2]. In addition, the user interface and design used by the GridPort GridFTP portlets will be adopted so the GridPort demo portal has a uniform look and feel. Another area of research will be the use of Java Server Faces, which will allow clients to make simpler web pages that will access portal services running on the Web server. Finally, plans are underway to develop a portlet that abstracts the nature of the data system behind the portal so that the user is unaware of whether or not they are using FTP, SSH, SRB, etc. to access or migrate files.

References

[PB] J. Linwood, D. Minter. “Building Portals with the Java Portlet API.” Published by Apress, 2005.

[TG] TeraGrid Gateways Project. Website last accessed on 1-Jan-06 at http://www.teragrid.org/programs/sci_gateways/

[GP4] The GridPort Toolkit Project. Website last accessed on 1-Jan-06 at http://www.gridport.net.

[GS] The GridSphere Project. Website last accessed on 1-Jan-06 at http://www.gridsphere.org

[GM]: Google Map Search Page. Website last accessed on 1-Jan-06 at http://maps.google.com.

[JSP-1] Jetspeed-1 Portlet Tutorial. Website last accessed on 1-Jan-06 at http://portals.apache.org/jetspeed-1/tutorial/8/jsp.html

[MPT-1] D. Gannon, G. Fox, M. Pierce, B. Plale, G. von Laszewski, C. Severance, J. Hardin, J. Alameda, M. Thomas, J. Boisseau, Grid Portals: A Scientist's Access Point for Grid Services, GGF Community Practice document, working draft 1, September 2003.

[SRB] SDSC SRB Project. Website last accessed on 1-Jan-06 at at http://www.sdsc.edu/srb.

[JAR] SDSC Jargon Project. Website last accessed on 1-Jan-06 at at http://www.sdsc.edu/srb/jargon/index.html.

[OGCE] NSF NMI Open Grid Computing Environments (OGCE) project. Website last accessed on 1-Jan-06 at http://www.ogce.org.

[MPT-2] M. P. Thomas, J. Burruss, L. Cinquini, G. Fox, D. Gannon, L. Gilbert, G. von Laszewski, K. Jackson, D. Middleton, R. Moore, M. Pierce, B. Plale, A. Rajasekar, R. Regno, E. Roberts, D. Schissel, A. Seth, and W. Schroeder. Grid Portal Architectures for Scientific Applications. Accepted for publication in Journal of Physics: Conference Series, 2005.