Interfacing CCLRC's Atlas Data Store to the Grid Using the Storage Resource Broker

Bonny Strong, David Corney, Tim Folkes, and Peter Berrisford

CCLRC

Abstract

The Atlas Data Store (ADS) is a petabyte-scale mass storage system developed and managed at CCLRC. The San Diego Supercomputer Center (SDSC) Storage Resource Broker (SRB) is client-server grid middleware for connecting heterogeneous data resources. A driver for SRB has been developed to interface to the ADS. With this, the ADS becomes one of the SRB resource types, thus providing ADS users with advanced tools for data management and a choice of well-supported user interfaces. This paper describes why the driver was needed, how it was implemented, what problems and issues were encountered, how data transfer occurs, and what advantages this implementation has provided to the users of the ADS.

1. Background

The Atlas Data Store (ADS) at CCLRC is a petabyte-scale mass storage system, with front-end servers incorporating a disk cache. It includes software developed at CCLRC to handle basic data access functions. Many users would like to have access to the ADS for archiving or storing large amounts of data, but want more advanced grid-based data management functionality and user interfaces.

The San Diego Supercomputer Center (SDSC) Storage Resource Broker (SRB) is client-server middleware that provides a uniform interface for connecting to heterogeneous data resources over a network and accessing replicated data sets. By implementing an SRB driver into the ADS, this functionality has been provided to users of the ADS.

The original impetus to develop the SRB-ADS driver came from the CMS particle physics experiment. As they prepared in 2003 for their 2004 Data Challenge, they needed tools to manage large numbers of data files distributed around the world, and they needed the ability to transfer some of these files into and out of the ADS. They evaluated grid tools that were ready for production use at that time and found SRB most appropriate. To use SRB, an interface between SRB and the ADS was required.

Figure 1 shows a typical architecture for an SRB domain with an SRB-ADS server added for access to the Atlas Data Store.

2. Driver Implementation

The SRB code has been written with a multi-layered approach, so that low-level drivers can be added for different types of storage devices by implementing a clearly defined interface.

The 20 driver functions that define the interface for a file-type device are shown in Table 1.

create	
open	
close	
unlink	
read	
write	
sync	
stat	
fstat	
mkdir	
seek	- not supported in ADS
chmod	- not supported in ADS
rmdir	- not supported in ADS
opendir	- not supported in ADS
closedir	- not supported in ADS
readdir	- not supported in ADS
setstorattri	– not supported in ADS
migrate	- not supported in ADS
stage	- not supported in ADS
purge	- not supported in ADS

Table 1. Functons defined for SRB driver

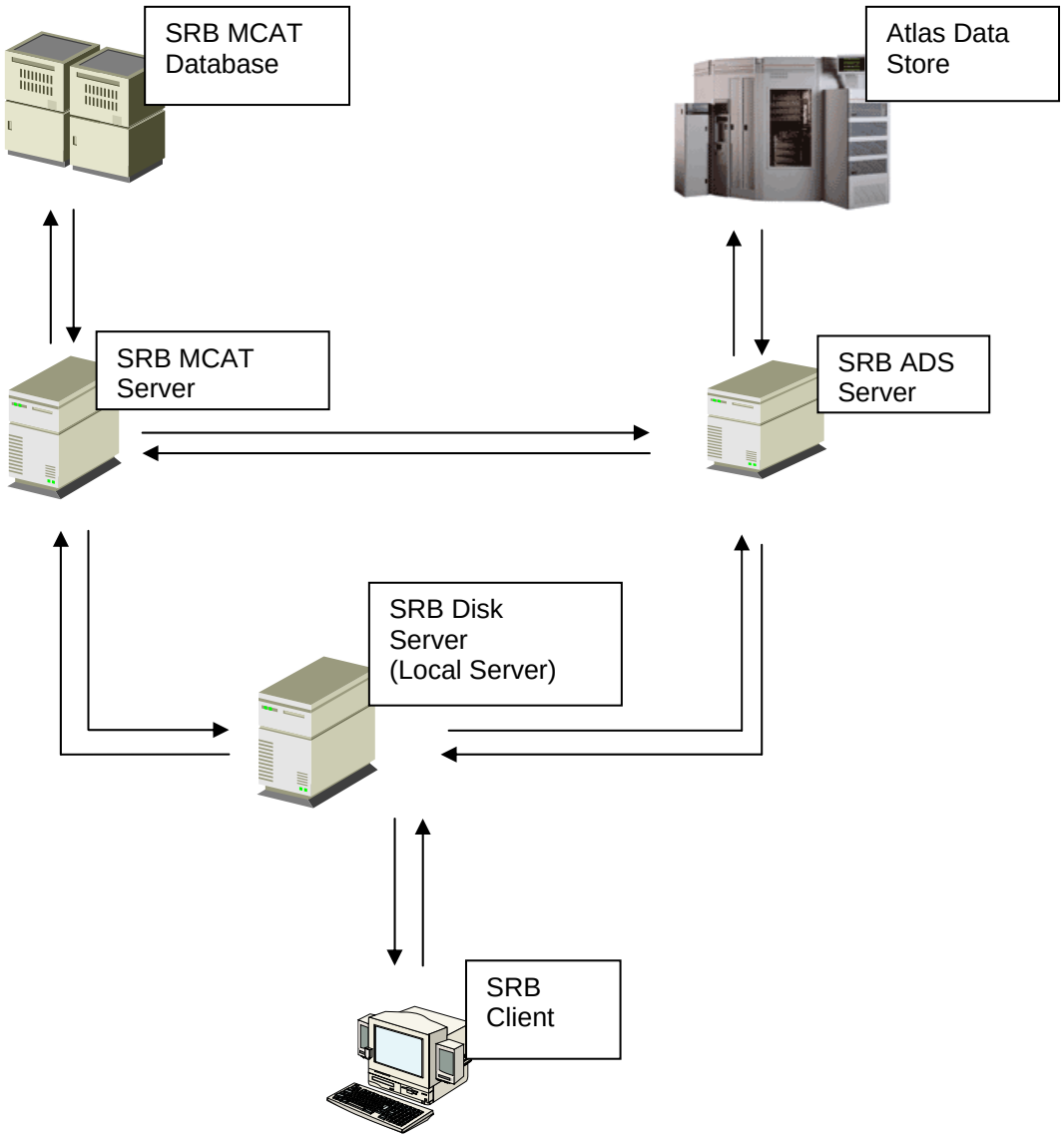

Figure 1.Typical SRB architecture with ADS server included

SRB provides for 2 types of drivers: database-type and file-type. The driver for the ADS is a file-type driver, which needs to support Unix-like I/O functions, such as create, open, read, write, and close. The ADS itself provides a low-level API, called the Virtual Tape Protocol (VTP) library, which also supports Unix-like functionality. Thus, mapping SRB driver functions to the ADS API was quite straightforward.

3. Issues and Problems

3.1. Seek funtion and parallel transfers

The 10 driver functions that were not implemented in the ADS presented little problem, since they were not critical to SRB functionality. The exception to this was the seek() function. Since a seek operation is not relevant on a tape device, the VTP API does not support it. SRB has the option to do parallel data transfer for “put” and “get” commands, where SRB will sub-divide the file to be transferred into segments and transfer each segment in a separate thread. This requires use

of the seek function, so is not available from the ADS driver.

3.2. SRB Containers

The seek function is also required to handle SRB containers, which group many small files into a single object. The lack of this functionality proved to be critical for users of the ADS. When users try to put many small files into the ADS as separate datasets, each one requires a tape to be mounted, written, then dismounted. Mount and dismount operations require about 30 seconds each, whereas the write operation may take only 2-3 seconds for a small file (where small is less than 50 MBytes). By grouping small files into containers, the overhead for tape mounts/dismounts is greatly reduced, and throughput greatly increased.

Initially, we anticipated that implementing containers would require additional development work in the driver. However, we found that SRB has the capability to handle a "logical resource" which consists of a disk cache device linked to a mass-storage device. Files can be written to the cache device, which can support the seek command and containers. Either when a container is full, or when the user issues a sync command, the container is copied to tape as a single dataset.

By providing a disk storage resource, we were able to configure an ADS logical resource that fully supported SRB containers, and required no additional development.

3.3. Mapping to ADS tape owner names

Another issue encountered in building the driver was how to map from SRB user and domain names to ADS tape owner names. The prototype driver was built to support only the CMS project, so all tapes were written into the ADS with an owner name of SRBCMS.

To extend the SRB interface to other users of the ADS, it is necessary to determine how SRB users and domains will map to ADS tape owner names. It is planned to implement 2 types of ADS-SRB servers. The first type is project-based and will map the SRB domain name onto an ADS owner name. For example, an SRB domain is planned for the British Atmospheric Data Centre (BADC). There will be an SRB domain for BADC, and this will map to an ADS owner name of SRBBADC.

The second type of server will be user-based. This will map the SRB user name to the same ADS owner name, with a prefix of "SRB". The prefix will guarantee that data written into the ADS via SRB will be identified by tape owner names unique from any non-SRB-owned data.

3.4. Access Control

Within the ADS, access is controlled by rules attached to owner names. Rules for SRBCMS have been set so datasets owned by SRBCMS are only accessible from machines designated as SRB-ADS servers, and where requests are submitted by a Unix username of "srb". It is anticipated to do the same for any future SRB owners.

The ADS supports other interfaces for data access, such as the EDG/LCG Storage Element and the ADS "tape" program. It would be possible to define ADS rules to allow access to data created within SRB from other interfaces, but a policy decision has been made to disallow this. In this way, the SRB system can enforce access control according to SRB policies, with assurance that files will not change out from under it, or be accessible outside SRB.

3.5. Multiple ADS-SRB servers

We anticipate that we may need to run multiple SRB-ADS servers, for example one which works as a project-based server as described above and one which is a user-based server. We have explored issues with running multiple SRB servers on the same machine, on different ports. The relevant issue here is that all SRB servers in a domain must run on the same port number. This implies that 2 projects, each with their own MCAT-servers running on different port numbers, cannot both access the same SRB-ADS server. This has had implications for architecture of SRB domains and projects.

3.6. Server redundancy

As we expand use of SRB as a user interface to the ADS, we would like to have SRB-ADS server redundancy, to insure access to users files will continue in the event of a hardware failure. Initially we will have backup hardware available as a cold backup. If an ADS-SRB server machine fails, the backup machine will be reconfigured with the same IP address and SRB software and be booted to replace the

failed server. Any data stored in the ADS from an SRB domain will again be available. No data files are stored on the SRB-ADS server.

We would prefer to run a farm of ADS-SRB servers, such that any of the machines in the farm can be used to access data stored in the ADS from an SRB domain. We are currently investigating issues associated with this. We need to determine if the SRB system uses the server id as part of the metadata that defines the location of an SRB object. If so, this would prevent a farm-type architecture. SDSC are willing to make modifications within SRB, if necessary, to support this architecture.

4. Usage and Data Transfer

To use the ADS within an SRB domain, a new ADS resource is created by the SRB MCAT-administrator, using the new token type "ads_uk". An SRB user can then issue an SRB put command to transfer data from a local system into the ADS resource.

The process of transferring data from an SRB client into the ADS is as follows:

- SRB client is initialized and connected to local SRB server.
- Put command is issued by client; local server makes call to MCAT-server to create an entry.
- Local server establishes a connection to the SRB ADS-server, and passes calls to it for a remote-create and remote-open.
- ADS driver makes calls to create dataset in ADS and executes VTP open. Connection established between SRB-ADS server and one of ADS VTP servers.
- SRB client writes buffers of data to local SRB-server. Each buffer is passed to SRB-ADS server, using SRB internal protocol.
- SRB-ADS server sends each buffer to VTP server, using the VTP protocol.

5. Advantages to Users

A number of projects have expressed interest in archiving large amounts of data in the ADS and using the SRB for managing this data. The ADS provides reliable and efficient petabyte-scale data storage, and transparently manages changes in data storage technology. However, the ADS only provides a low-level user interface with minimal data management functionality.

The SRB provides multiple, higher-level interfaces including a Windows-based GUI and a web-based interface as our users have requested. SRB also provides facilities for managing metadata, replicating data, and managing access control. By implementing the SRB driver for the ADS, we have given our users the hardware-independent petabyte-scale storage capacity of the ADS, together with the advanced data management functionality of SRB.

6. Further Information

For further information about the Atlas Data Store, see:
www.e-science.clrc.ac.uk/web/services/datastore

For further information about the Storage Resource Broker, see:
www.npaci.edu/DICE/SRB