# Key Lessons in the Efficient Archive of Small Files to the CCLRC MSS Using SRB

Bonny Strong
*CCLRC e-Science, U.K.*
*b.strong@rl.ac.uk*

David Corney
*CCLRC e-Science, U.K.*
*d.corney@rl.ac.uk*

Peter Berrisford
*CCLRC e-Science, U.K.*
*p.berrisford@rl.ac.uk*

Tim Folkes
*CCLRC e-Science, U.K.*
*t.folkes@rl.ac.uk*

Chris Moreton-Smith
*CCLRC ISIS, U.K.*
*c.m.moreton-smith@rl.ac.uk*

Kerstin Kleese-Van-Dam
*CCLRC e-Science, U.K.*
*k.kleese-van-dam@rl.ac.uk*

## Abstract

*High volume data projects within the CCLRC and the wider UK academic community (ISIS, BADC, and BBSRC) are increasingly looking to implement access to a "limitless" data archive through an SRB infrastructure. This paper describes the recent development of SRB containers as an efficient solution to the "small file problem", and the benefits this has brought to these scientific communities as it opens up high volume archives and Mass Storage Systems as vital components of SRB-based data management infrastructures. By tracing the development of container implementation into the Atlas Petabyte Data Store (based at CCLRC in the UK), across three specific projects (CMS, BBSRC and especially ISIS), the paper identifies and describes key lessons learned, both for CCLRC's particular projects and also for archival systems in general.*

## 1 Introduction: Mass Storage Systems and the Small File Problem

In archiving data into a mass storage system (MSS), a common problem develops when users want to archive a large number of small files. Small files typically make very inefficient use of mass storage capabilities. Tape drives in particular suffer poor performance with small files. Each write operation requires a seek operation to the position on the tape to begin writing, and then writing a tape header. Furthermore, if files are not being written sequentially to the same tape, it may be necessary to load a new tape onto the drive for each file. For small files, the tape mount may take longer than the write itself.

Figure 1 shows an analysis of tape drive performance as a function of file size, using data gathered from files transferred into the CCLRC MSS during the 2004 data challenge performed by the CMS particle physics experiment. It can be seen that at file sizes of greater than 200 MB, the tape drive throughput levels off at 25-30 MB/sec, but at sizes below 200 MB, drive throughput remains at less than 15 MB/sec, and falls off dramatically with very small file sizes.

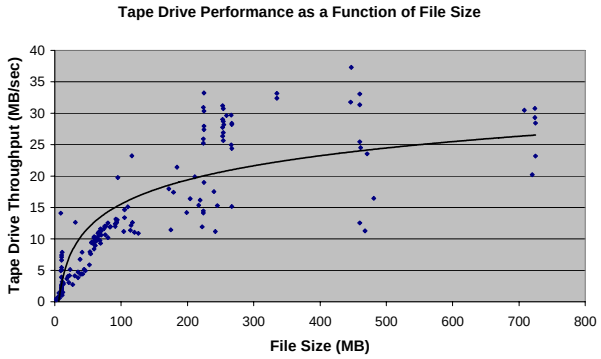


**Figure 1. Tape drive performance and file size.**

Large numbers of small files into an MSS can have a disastrous effect on overall system throughput. Managing this small-file problem becomes a critical part of managing a data archive system.

## 2 Brief history of the development of SRB and the Atlas Petabyte MSS at CCLRC

The CCLRC Atlas Data Store (ADS) [1] is a mass storage system built around an STK Powderhorn tape robot, with a current capacity of one petabyte, expected to grow to ten petabytes within five years. It is managed

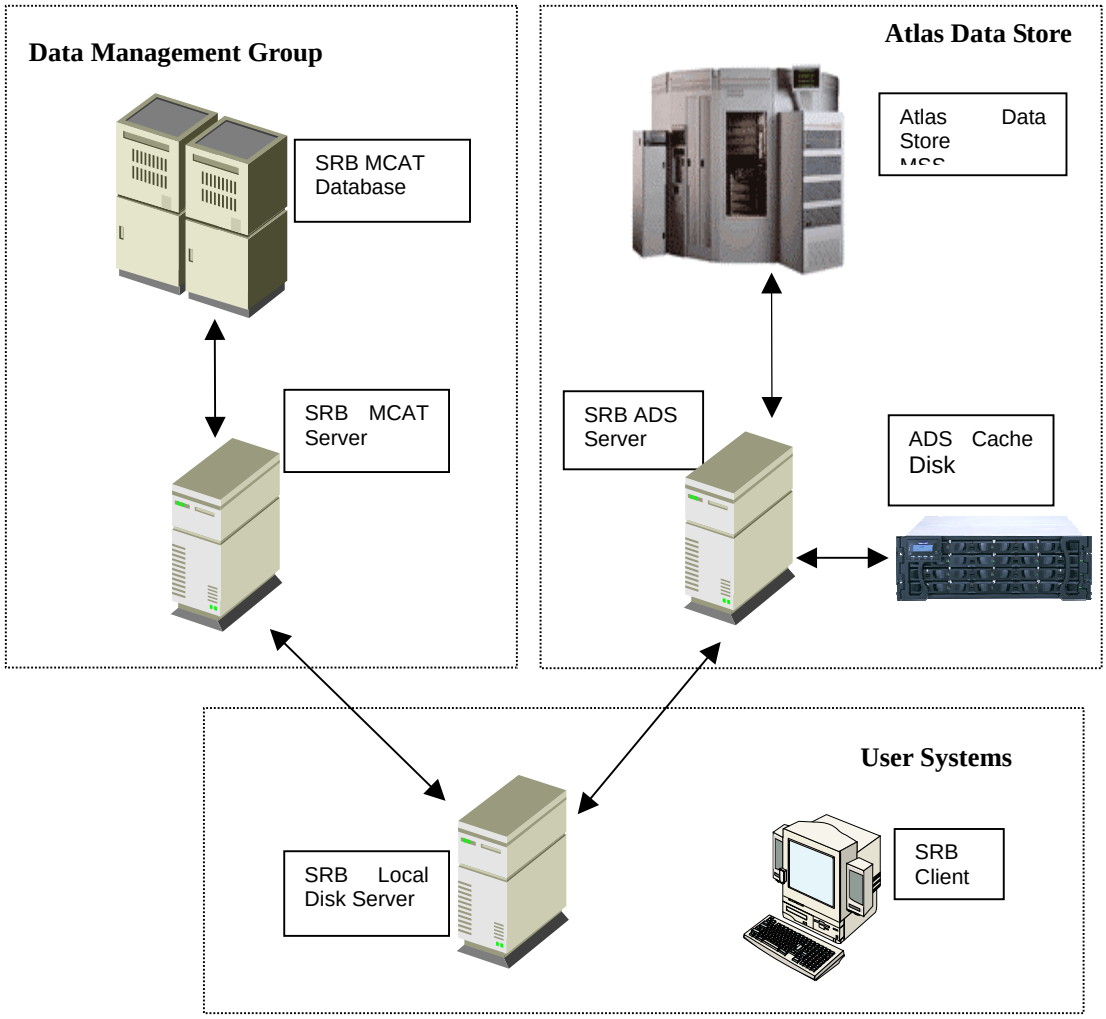


**Figure 2. CCLRC Data Archival Management Architecture**

with software developed at CCLRC, which is very strong on efficiency and reliability, but has only a very basic user interface. The ADS provides an efficient tool for scientific archiving, but not data management.

CCLRC Data Management Group (DMG) [2] has been researching issues around the management of scientific data and associated metadata, and has built and managed Data Access, Metadata Schema and Databases. They have, as well, managed services for the San Diego Supercomputer Center (SDSC) Storage Resource Broker (SRB) – a software package that provides a uniform interface for connecting to heterogeneous data resources over a network [3]. SRB has provided a critical tool for managing geographically distributed data across different computational platforms, and has provided much-requested desktop accessible tools for data management. But many users also need the capability to archive large quantities of data managed by SRB.

We have found that the integration of the data management using SRB services and archiving capabilities using the ADS provides a facility much more powerful than either facility alone, and one that has proven to be of major interest to scientific users with large-scale data archival needs. Figure 2 shows the typical architecture used for this managed archive facility. Currently the ADS and DMG groups are building or managing such an archival platform for ISIS (the CCLRC Centre for pulsed neutron and muon research), for the Biotechnology and Biological Sciences Research Council (BBSRC), for the British Atmospheric Data Group (BADC), and for the grid-enabling of other CCLRC facilities. We have had other groups indicate an interest in exploring this type of facility.

# 3 Developing the effective use of SRB Containers

In order to address the small-file problem, we have used an SRB facility for bundling data called "containers". From the SRB website (http://www.sdsc.edu/srb/faq.html):

"The SRB container is a like a tarball in the sense that it stores multiple files as one single file. It grows the container on the fly by adding new files as they are ingested into the container. Hence, unlike a tarball, the container can be grown as needed. Also, unlike a tarball, users can read individual files without downloading the container on to their desktops. Containers are normally assigned a logical resource that has two physical components: an archive resource and a cache resource such as a unix file system. Containers grow in size and are pinched off into physical pieces by the SRB so that a container might look really long, but is actually multiple files of smaller sizes. "

When containers are "pinched off into pieces" they form a family of containers in SRB nomenclature. If a file holding a container fills beyond its maximum size, it is closed and copied to another file specified as part of this container family. Then a new file container-file is opened. Thus, to users containers appear infinitely large, while at a physical level, containers are broken into sizes manageable by the system, while SRB maintains all the information needed to link the pieces of a container family together.

## 3.1 How it started: CMS 2003

We first encountered the small-file problem while working with the CMS particle physics experiment. They developed a prototype data management system using SRB, which transferred data between a number of internationally distributed sites and RAL. To support their work, we developed a driver to provide SRB access into the ADS.

One of the key problems that emerged during the data challenge arose from the unexpected number of small files. During their data challenge, from 15 March 2004 to 30 April20 04, CMS entered 434861 data files into the SRB system, with a total size of 4898.76 GB, giving an average file size of only 11 MB. The result was very poor performance in data transfer, and problems managing a large number of file indices in the SRB catalogue and in the internal ADS catalogue.

CMS were very keen to use SRB containers to improve their transfer performance, but at that time we were transferring data directly to tape, which could not support the use of containers.

## 3.2 Specific problems: development for ISIS

Following on from the work done for CMS, ISIS was interested in developing an archival data management solution using SRB. ISIS manages 20 instruments that provide pulsed neutron and muon beams for visiting teams of scientists to use in their own individual research programmes. While experiments are running they produce data 24 hours a day, which must be archived and made available to the scientists, both immediately during the running of their experiments and longer term after they return to their home institutions. Depending on the science of the experiment, a "run" of data is produced from the instrument anywhere from every 2 minutes to every 2 days. A run will produce one large file of data (about 100 MB) and possibly ten or more small files of descriptive information.

When testing began, it became evident that the number of small files was causing a critical problem with performance in archiving data into the ADS. In one test a sample 7-day period with 6 active instruments was analysed. This produced a total of 24.9 GB to be archived, contained in 14,614 files. Of these, 12,500 files were small files of less than 100 MB, containing a total of only 86MB of data. These were ingested into SRB using an Sput command, which was taking 6 seconds per file, giving a total time of 21 hours to archive just the small files from this sample.

This was judged far too slow for flexible management of data. ISIS wanted a ratio of real network copy time to "setup" time to be 10:1 or better. For example, if there were an unavoidable problem with transfer for a day or two from ISIS, they would need to be able to rapidly get the archive back up to date, which implies a much greater archiving rate than would normally be required (10x or one order of magnitude as a rule of thumb was deemed reasonable). Also people could easily restore a few GB of raw data without that much knowledge of structure and expect it to come back roughly as fast as the network would be able to provide it even if there were a lot of small files in there.

At this point, a decision was made that containers would be critical to using the ADS archive efficiently in conjunction with SRB.

## 3.3 A solution: containers implemented

Our initial understanding of containers led us to believe that some development work would be necessary to implement containers for the ADS. In fact, we found that the only steps necessary were to install an appropriate disk for cache space, and change the resource configuration. A 2-TB RAID disk was purchased, and attached to the host machine that was running the ADS server. Within SRB a logical resource was configured

which included 2 physical resources: the ADS tape storage system, and the new cache disk.

Using containers required some additional knowledge and management on the part of the users. Whereas in the direct-to-tape implementation, a file could be transferred into the ADS with a single Sput command, the use of containers requires 3 steps for a file transfer: 1) create a container on the ADS logical resource; 2) transfer data into the container, using Sput or a similar command, and 3) sync the container using the Ssyncont command, with an option to delete the data from the cache at the same time.

Users can access the file using the same commands as used for non-containerised files. When a file in a container is accessed, SRB checks to see if the container exists on cache. If not, the container is copied from tape to cache, and access proceeds from cache, with all the characteristics of any file stored on a disk resource.

SRB has the capability to do parallel data transfer, but this cannot be used when writing or reading directly to or from tape. With containers, parallel transfer can be used while copying files into the container, followed by an asynchronous transfer to tape, which speeds up transfer times into the ADS.

Extensive testing and use of containers by ISIS uncovered a few bugs in SRB, which were promptly fixed. At our request, SDSC also provided some enhanced functionality for information about container contents and container families, and for administration of containers.

With containers implemented, ISIS retested the transfer of small files. Whereas, without containers, archive time was taking about 6 seconds per file, with containers this was reduced to about 1 second per file.

### 3.4 Managing containers for users

Training was prepared by the CCLRC Data Management Group to teach users how containers work and how to use them effectively. Users need to understand how data moves between cache and disk, the importance of issuing the sync command to actually initiate transfer to tape, how families of containers work, and how to set the container size for optimal tape usage.

Administering containers required that we develop a few administrative scripts to run on the ADS server for the following functions:

1) Sync containers to tape, to insure that a copy has been written to tape.
2) Sync-and-delete any containers on cache to clear out the disk cache. Ideally, this should be done using an algorithm of deleting the oldest-accessed files only when additional cache is needed. Recently used containers are the most likely to be accessed again, and not deleting them can reduce tape usage.
3) Monitor that the cache disk is not filling up.

In practice, as this is an archival system, usage patterns show that a container is created and immediately filled, then not accessed again for some time. So we currently have a single script that runs nightly to perform the functionality of (1) and (2) above, and this has to date proven quite adequate.

### 3.5 A few more lessons worthy of note – BBSRC

Following the implementation of containers for ISIS, another project has been initiated which takes the use of containers a step further. CCLRC is undertaking development for BBSRC to manage an SRB system for their data archival needs, and to do additional development to customise SRB for their specialised needs. This includes a special-purpose GUI to manage the end-to-end transfer and tracking of archive packages. Their data transfer must take place in 2 steps: from BBSRC local sites to a central site over slow network connections, and then nightly during specified hours from the central site into the ADS over a higher speed network. Users then require email notification when 2 copies of their data are resident on tape.

Achieving the best performance possible for data transfer was a key requirement for this project. After a series of comparative tests, it was determined that optimum transfer speeds could be achieved through the creation of containers on an intermediate disk resource at the central site, followed by the replication of the containerised archive package to the ADS SRB cache resource using the Sreplcont command. Once in a container, the package is effectively treated as a large file, enabling full use of SRB's parallel capabilities. Data transfer speeds that reflect maximum utilisation of the available network bandwidth have been consistently demonstrated.

It has been necessary to expand the logical resource model previously used with the ADS to cater for the additional physical cache resource. The Sreplcont command has been enhanced to allow the explicit specification of the target resource for the replica. The final step is to "synchronise" the containerised data to tape, using the Ssyncont command and remove all cached replicas.

Given that the data is staged on its way to the ADS, it is vital that performance is optimised for all stages of the data transfer process. While containers are used from the central site cache onwards, it is still necessary to get the data into the container. "Bulk" data transfer options had already existed within SRB for data ingestion and extraction, but not for data movement within SRB. The

bulk option allows for implicit temporary creation of containers for efficient data transfer, combined with a bulk metadata update. SDSC have now added bulk options to the Sphymove (physical data transfer between resources) and Scp (copy) commands. To illustrate the importance of the new command options, the transfer of a test SRB "collection hierarchy" was taking over twelve minutes – this has now been reduced to one minute.

One final area in which the BBSRC project has pushed previous limits is in the size of containers. The previous 2GB limit is no longer a constraint, with the new bulk Sphymove option allowing the creation of a container that will hold a complete archive package far greater than 2GB in size. The only limitations relate to the underlying storage system and the practicality of restoring colossal containers to cache if only a few files are required.

# 4 Conclusions

## 4.1 Lessons learned

Overcoming the small-file problem is critical for an efficient interface to a mass storage system. Ignoring it leads to poor utilization of tape, network, and database resources, and can bring MSS access to a near standstill.

Containers provided by SRB have proved relatively easy to implement and extremely valuable in managing the small-file problem and improving transfer rates. SRB provides the tools necessary to install a cache disk in front of the MSS and manage the linkage. This gives users the capability of interacting with the MSS as if it were a disk file system.

Training users on how to effectively use containers is important. Some thought about what logical structure to use when grouping files into containers can lead to significant improvements in access times.

SRB has a wealth of commands that allow different approaches to how data is ingested, transferred, and accessed. It is worthwhile to give some thought and testing to which of the SRB commands is most appropriate for a particular project. We have found the Sreplcont command very useful in improving data transfer rates.

Our users have dictated to us that the combination of SRB services together with ADS large-scale archiving capabilities is required to provide effective tools for their data management needs.

## 4.2 Implications for the future

Containers are now an integral part of planning for new data archiving projects. In order to manage data transfers and tape handling efficiently, we may make this the only acceptable path into the ADS from SRB.

Projects that require archival of many small files will be encouraged to use the SRB interface for containerisation.

## 4.3 Where from here?

Some additional development and refinement of SRB container commands would be useful. For example:

- A facility to automatically sync a container to tape when it becomes full, and a new container family member is opened is planned.
- Viewing of container families could be more user- friendly.
- SRB effectively hides container families from users. However, from an administrator's or developer's point of view, it would be useful if they were not quite so effectively hidden. Identifying container families from internal SRB transfer names is not always straightforward.
- Better user documentation about containers is needed.

Improvements in our administration scripts will probably be required to better manage the cache space associated with the MSS as usage of containers scales up.

As containers have proved invaluable to us in managing small files, we will continue to work with SDSC to test, debug and document containers, and to expand their capabilities.

# References

[1] Atlas Data Store (CCLRC e-Science), http://www.e-science.clrc.ac.uk/web/services/datastore
[2] Data Management Services (CCLRC e-Science), http://www.e-science.clrc.ac.uk/web/projects/data_storage_and_management
[3] SRB: Storage Resource Broker, http://www.sdsc.edu/srb