On Building Trusted Digital Preservation Repositories

Reagan W. Moore, Arcot Rajasekar, Michael Wan, Wayne Schroeder, Richard Marciano

San Diego Supercomputer Center

Abstract

Trusted digital repository audit checklists are now being developed, based on assessments of organizational infrastructure, repository functions, community use, and technical infrastructure. These assessments can be expressed as rules that are applied on state information that define the criteria for trustworthiness. This paper maps the rules to the mechanisms that are needed in a trusted digital repository to minimize risk of both data and state information loss. The required mechanisms have been developed within the Storage Resource Broker data grid technology, and their use is illustrated on existing preservation repository projects.

1. Introduction

A trusted digital repository uses explicitly defined policies to manage records. These policies can be validated against criteria published by the RLG and NARA in "An Audit Checklist for the Certification of Trusted Digital Repositories" [1]. The checklist defines management policies that are organized into criteria for: The Organization; Repository Functions, Processes, and Procedures; The Designated Community & the Usability of Information; and Technologies & Technical Infrastructure. Each set of assessment criteria can be characterized as a set of rules applied to state information that define a specific management policy. The result of applying the rules can also be kept as state information within the trusted digital repository. An expression of the set of rules and state information is under development for preservation repositories built on the integration of DSpace [2] and the Storage Resource Broker (SRB) [3], and is available on request.

The rules assume that mechanisms exist within the preservation environment to ensure the integrity of both data and metadata. An essential component of a trusted digital repository is the ability to mitigate risk of loss of records and state information. We examine the types of mechanisms available within the Storage Resource Broker to assure the integrity of the digital repository. These mechanisms are available at all SRB installations, including SRB data grids installed outside of the San Diego Supercomputer Center.

1.1 RLG Assessment Criteria

The assessment criteria specify the policies that are needed for governance, sustainability, robustness, and use of the preservation facility. These policies in turn can be expressed as the set of expectations for assured data access and sustained linkage of preservation metadata to records. The RLG assessment criteria can be interpreted as categories of risk that must be managed for a preservation environment to be trustworthy.

1.2 Types of Risk

Managing data reliability requires protection against many types of risk, including:

Technology failure
- Storage media failure
- Vendor product errors
- Natural disaster
- Malicious user security breaches

Technology evolution
- Media obsolescence
- Format obsolescence
- Storage obsolescence
- Access mechanism obsolescence

Collection evolution
- Provenance metadata changes
- Name space degradation
- Semantics evolution (obsolescence of terms)

Organizational failure
- Storage operational errors
- Mismanagement of versions
- Loss of funding

The risks may be dynamic, requiring the ability to respond to faults that occur during data ingestion, data storage, and data retrieval. The faults may lead to data corruption (bad bits in the records), metadata loss, and data loss. A single copy of data or metadata is not sufficient to mitigate the risk of data loss. The use of a single facility also cannot mitigate against the risk of eventual data loss. Thus a viable preservation environment supports multiple copies of data and metadata, and provides mechanisms to synchronize the copies. By

choosing to replicate data across different types of storage media, across different vendor storage products, between geographically remote sites, and into deep archives, the technology failures can be addressed.

By supporting data virtualization, the ability to manage collection properties independently of the choice of storage system, most obsolescence issues can be addressed. Data virtualization enables the incorporation of new technology including new access protocols, new media, and new storage systems. Format obsolescence is managed through use of versions. Data virtualization also supports the evolution of the metadata schema, ensuring that new preservation authenticity attributes can be used over time. By choosing to federate independent preservation environments that use different sustainability models and different operational models, it is possible to address the organizational challenges.

2. Risk Mitigation Mechanisms

The Storage Resource Broker supports a wide variety of mechanisms that can be used to address the above risks. The mechanisms can be roughly divided into the following categories:

- Checksum validation
- Synchronization
- Replication, backups, and versions
- Federation

We examine the actual SRB Scommand utilities to illustrate how each of these types of integrity assurance mechanisms is used in support of the NARA Research Prototype Persistent Archive [4] and the NSF National Science Digital Library persistent archive [5].

2.1 Checksums

Checksums are used to validate the transfer of files, as well as the integrity of stored data. The SRB uses multiple types of checks:

- TCP/IP data transport checksum
- Simple check of file size
- Unix checksum (System5 sum command. Note that it does not detect blocks that are out of order)
- MD5 checksum (Robust checksum, implemented as a remote procedure)

The SRB uses TCP/IP to do all data transport. This implies that checksums are validated by the transfer protocol during all data movement. However the TCP/IP checksum is susceptible to multiple bit errors and may not detect corruption of large files. Also, the transport protocol does not guarantee end-to-end data integrity, from the submitting application to the final disk storage.

The preferred method is to checksum a file before transport, register the value of the checksum in the MCAT metadata catalog, and then verify the checksum after the transfer.

Sput –k *localfilename SRBfilename*
: Put a file into a SRB collection. Client computes simple checksum (System5 sum command) of the local file and registers with MCAT. No verification is done on the server side.

Sput –K *localfilename SRBfilename*
: Put a file into a SRB collection. After the transfer, the server computes the checksum by reading back the file that was just stored. This value is then compared with the source checksum value provided by the client. This verified checksum value is then registered with MCAT.

Sget –k *SRBfilename localfilename*
: Get a file from a SRB collection. Retrieves simple checksum (System5 sum command result) from the MCAT and compares with the checksum of the local file just downloaded.

The SRB provides mechanisms to list size and checksums of files that were loaded into a SRB collection. The utilities identify discrepancies between the preservation state information and the files in the storage repositories.

Sls -V
: Verifies file size in a SRB vault with file size registered in MCAT. Lists error when file does not exist or the sizes do not match.

Schksum -l *SRBfilename*
: Lists the checksum values stored in MCAT including checksums of all replicas.

Schksum *SRBfilename*
: If a checksum exists, nothing is done. If the checksum does not exist, it is computed and stored in MCAT.

Schksum –f *SRBfilename*
: Force the computation and registration of a checksum in MCAT.

Schksum –c *SRBfilename*
: Computes checksum and verifies value with the checksum registered in MCAT

Spcommand –d <*SRBfile*> "*command command-input*"
: Discovers where *SRBfile* is stored, and executes *command* at the remote site. The MD5 checksum is implemented as a proxy command.

2.2 Synchronization

The SRB supports the synchronization of files from system buffers to storage, between SRB

collections, and between a SRB collection and a local directory. The synchronization commands are used to overcome errors such as loss of a file, or a system halt during transfer of a file. In addition, the SRB server now incorporates mechanisms to validate the integrity of recently transferred files.

srbFileChk server

The srbFileChk server checks the integrity of newly uploaded files. By default, it runs on the MCAT enabled host. It can also be configured to run on other servers by changing the fileChkOnMesa and fileChkOnThisServer parameters in the runsrb script.

A fairly nasty problem with a loss of data integrity can exist in recently uploaded files if the UNIX OS of the resource server crashes during the transfer. Data that are uploaded successfully may still be in a system buffer and not on disk. This problem is particularly bad because the SRB system has already registered the files in the MCAT and is not aware of the integrity problem until someone tries to retrieve the data. A typical symptom of this mode of corruption is the size of the corrupted file is not the same as the one registered in MCAT. The srbFileChk server performs a file check operation for newly created SRB and lists errors. By default, it wakes up once per day to perform the checking.

SphyMove –S *targetResource SRBfilename*

Sysadmin command to move user's files between storage resources. Since the original copy is removed after the transfer, the UNIX fsync call is executed after each successful SphyMove to ensure the files do not remain in a system buffer.

Srsync –s *Localfilename s:SRBfilename*

Synchronize files from local repository to SRB collection checking for differences in file size.

Srsync –s s:*SRBfilename Localfilename*

Synchronize files from SRB collection to local repository checking for differences in file size.

Srsync –s s:*SRBsourcefile s:SRBtargetfile*

Synchronize files between two SRB collections checking for differences in size.

Srsync *Localsourcefile s:SRBtargetfile*

Synchronize files from local repository to SRB collection using checksum registered in MCAT.

Srsync –l s:*SRBsourceCollection s:SRBtargetCollection*

List all the files that have different checksums between the source and destination SRB collections.

Synchronization commands are also used to ensure that the multiple copies of a file have the same bits.

Ssyncd *SRBfilename*

Synchronize all replicas of a file to be the same as the "dirty" or changed version.

Ssyncd –a –S *SRBresourcename SRBfilename*

Synchronize all replicas of a file, ensuring that a copy exists on each of the physical storage systems represented by the logical storage resource name *SRBresourcenam.*

Ssyncont *SRBcontainer*

Synchronize the permanent copy of a SRB container (residing on tape) with the cached copy residing on disk.

Ssyncont –z *mcatZone SRBcontainer*

Synchronize the SRB container residing in the data grid Zone *mcatZone* with the cached copy residing on disk in data grid Zone *mcatZone*

Synchronization between the metadata catalog and the storage repository is also needed, either to delete state information for which no file exists, or to delete files for which no state information exists.

Schksum -s *SRBfilename*

Does a quick check on data integrity using file size. Any discrepancies are listed as errors, including files not present on the storage system. This identifies bad state information.

vaultchk utility

This is a sysadmin tool that identifies and deletes orphan files in the UNIX vaults. Orphan files (files in SRB vaults with no entry in the MCAT) can exist in the SRB vault when the SRB server stops due to system crashes or server shutdown at certain critical points (such as during a bulk upload operation).

2.3 Replica, backups, and versions

The synchronization commands rely on the existence of multiple copies of the files. We differentiate between:

- Replica, a copy of a file that is stored under the same logical SRB file name.
- Backup, a copy of a file that has a time stamp.
- Version, a copy of a file that has an associated version number.

Replicas can be synchronized to ensure that they are bit for bit identical. A Backup enables the storage of a snapshot of a file at a particular time. Versions enable management of changes

to a file. Commands are needed to create each of these types of copies and to verify that identical copies are indeed identical.

Sreplicate *SRBfilename*
Create a copy of a SRB file and register as a replica of the existing SRB file. Each time the command is executed another replica is made.

Sreplicate –l *localfilename SRBfilename*
Load a local file into a SRB collection and register as a replica of an existing SRB file called *SRBfilename.*

Sreplcont –S *SRBresource SRBcontainer*
Replicate a SRB container onto the named SRB resource.

Sbkupsrb –S *SRBresource SRBcollection*
Backup a SRB collection to the SRB logical resource. First check whether a copy of the data already exists and that the copy is up to date. If not, create a copy on the resource.

Sbkupsrb –K–S *SRBresource SRBcollection*
Verify the backup copy is indeed copied correctly by creating a checksum on the original, and then computing a checksum after the copy is made and comparing with the original.

Sput –n *replicanumber Localfilename SRBfilename*
Load a local file into a SRB collection as a specific replica number or version.

SmodD –V *replicanumber oldVersionString newVersionString SRBfilename*
Set a version identifier for the SRB file copy identified by *replicanumber.*

2.4 Federation

The SRB supports synchronization between two independent data grids. This provides a way to ensure that a copy of files and their metadata exist in a geographically remote data grid that can be under separate administrative control, using different storage resources, using different storage technology, and governed by a different sustainability model. The ability to create a copy of a collection ensures that the choice of storage technology or sustainability model does not constitute a risk.

Szonesync.pl –u –z *SRBZone*
Sysadmin command to synchronize user information between the local data grid and the specified remote data grid *SRBZone.*

Szonesync.pl –d –z *SRBZone*
Sysadmin command to synchronize data information between data grids. This is equivalent to the registration of files from the local data grid into the remote data grid.

Scp –S *Zoneresourcename SRBsourcecollection SRBtargetcollection*
Copy each SRB file in the local zone source collection to the remote zone target collection and store files on the specified remote zone logical resource. Each collection name is written as:
/zone/home/user.domain/collection
Thus copying between data grids just requires specifying the zone name as part of the collection name.

Srsync s:*SourceCollection s:TargetCollection*
Synchronize the files in SourceCollection
/localzone/home/user.domain/sourcecollection
with the files in TargetCollection
/remotezone/home/user.domain/targetcollection
using the checksums registered in the MCAT catalogs to detect differences.

3. Summary

A surprisingly large number of commands are required to implement the replication, synchronization, and federation mechanisms that are needed to meet assessment criteria for Trusted Digital Repositories. These mechanisms have been implemented in the Storage Resource Broker.

Acknowledgements

This project was supported by the National Archives and Records Administration under NSF cooperative agreement 0523307 through a supplement to SCI 0438741, "Cyberinfrastructure; From Vision to Reality". 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.

References

1. *An Audit Checklist for the Certification of Trusted Digital Repositories*. RLG, Mountain View,CA, August 2005. http://www.rlg.org/en/pdfs/rlgnara-repositorieschecklist.pdf
2. DSpace, http://www.dspace.org/
3. Storage Resource Broker, http://www.sdsc.edu/srb/
4. NARA Research Prototype Persistent Archive, http://www.sdsc.edu/NARA
5. National Science Digital Library, http://nsdl.org/