STORAGE RESOURCE MANAGER

T. Perelmutov, J. Bakken, D. Petravick, FNAL, Batavia, IL 60510, USA

Abstract

Storage Resource Managers (SRMs) are middleware components whose function is to provide dynamic space allocation and file management on shared storage components on the Grid[1,2]. SRMs support protocol negotiation and reliable replication mechanism. The SRM standard supports independent SRM implementations, allowing for a uniform access to heterogeneous storage elements. SRMs allow site-specific policies at each location. Resource Reservations made through SRMs have limited lifetimes and allow for automatic collection of unused resources thus preventing clogging of storage systems with "orphan" files.

At Fermilab, data handling systems use the SRM management interface to the dCache Distributed Disk Cache [3,4] and the Enstore Tape Storage System [5] as key components to satisfy current and future user requests [6]. The SAM project offers the SRM interface for its internal caches as well.

The Storage Resource Manager specification is a result of international collaborative effort by representatives of JLAB, LBNL, FNAL, EDG-WP2 and EDG-WP5 [7].

MOTIVATION AND FEATURES

SRM as a uniform grid storage interface

There is a proliferation of custom storage solutions at high energy physics laboratories. SRM interface provides a standard uniform management interface to these heterogeneous storage systems, providing a common interface to data grids, abstracting the peculiarities of each particular Mass Storage System. Designers of higher level grid middleware, such as Replica Managers [8], can concentrate on function rather then compatibility with all systems involved, as illustrated on Figure 1.

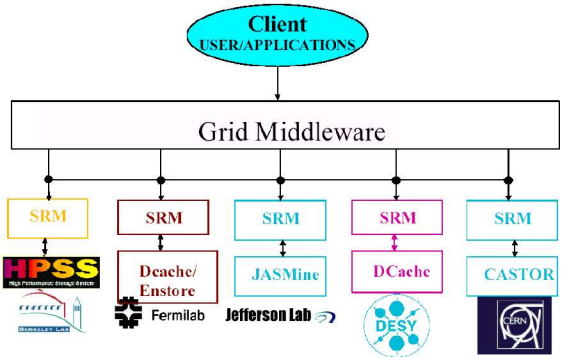


Figure 1: SRM as a uniform storage interface.

Protocol and TURLs negotiation

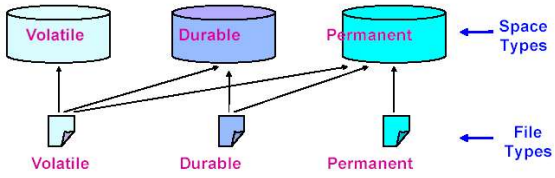
Protocol negotiation and dynamic transfer url generation allows to implement support for the multiple transfer protocols on the storage system, and still have a unique identifier of the data on the system.

Storage System provides the Transfer URLs (TURLs) to a client only after the data was brought to an accessible state, which might mean staging and pinning files within a hierarchical storage system. This feature provides the ability ability of the storage to marshal transfer requests, throttling client driven transfers.

Space types and space reservation

The SRM Protocol is general enough to be suitable to work on all types of storage systems. It specifically defines different types of Files and Spaces, supporting the permanent and temporary types of storage.

Along with the best effort Space and File Reservation, the SRM interface supports advanced Space Reservation. The storage systems can be classified on basis of their longevity and persistence of their data. Data can also be temporary or permanent. To support these notions, SRM defines Volatile, Durable and Permanent types of files and spaces (see Fig 2). Volatile files can be removed by the system to make space for new files upon the expiration of their lifetimes. Permanent files are expected to exist in the storage system for the lifetime of the storage system. Finally Durable files have both the lifetime associated with them and a mechanism of notification of owners and administrators of lifetime expiration. The notion is that when a lease on a durable file expires, there is a fault in some grid data handling system.



Property \ File Type	Volatile	Durable	Permanent
Archived	No	No	Yes
Lifetime	Yes	Yes	No
AutoDeleted	Yes	No	No

Figure 2: File and space types.

Direct Mass Storage System (MSS) to MSS transfers

SRM interface allows storage systems to be either the client or the server in a file transfer, enabling storage system to storage system transfers. This allows replication schedulers to take advantage of the distributed nature of some of the storage systems and of the knowledge of the internal system operating parameters, such as the current network load and availability of the system resources.

UTILIZATION OF STORAGE RESOURCE MANAGERS BY COMPACT MUON SOLENOID LARGE HADRON COLLIDER EXPERIMENT

The Compact Muon Solenoid (CMS) experiment is designed to explore the full range of physics at the high-energy frontier up to TeV mass/energy scales made available for the first time at CERN's Large Hadron Collider (LHC) [9,10,11,12]. CMS raw data originate at the CERN laboratory, near Geneva, Switzerland. Collaborators worldwide will analyse this data. Fermilab will play the role of a Tier One center in the CMS experiment. When transferring data from CERN, data will be exchanged between different petabyte (growing to Exabyte) class storage systems (CASTOR at CERN, and Enstore at Fermilab). Data will be transferred between Fermilab and North American Tier Two centers, which typically are large university data centers. The Storage Resource Manager (SRM) protocol was selected as a management protocol for both of these transfers, and the development of the protocol is seen by CMS as an essential abstraction allowing for more uniform software and system development.

Among the benefits to CMS of adoption of the SRM as a major interface to the storage systems are the following: ability to use a single client to access multiple storage systems in multiple sites; security; reliable replication services, implementing pacing of transfer to prevent disk thrashing and network clogging by queuing and limiting the number of simultaneous transfers. Another benefit of SRM is its ability to address storage resources independently of a transfer protocol and a specific user. This allows the SRM URLs to be used as physical file names for storage resources in CMS replica catalogs.

SRMs as building blocks for the CMS datagrid

A goal of the US-CMS data grid is to make data available to the end user at multiple US institutions. In order to achieve that, data after being taken and stored at primary storage at CERN (Tier 0), is replicated to Fermilab (Tier1) and then to Tier 2 centers.

Once the us-cms data grid is completed, it might look like diagram on Figure 3. showing where SRM is already used or can be used to control the data movement and manage the storage.

The experimental data is generated at the CMS experiment at CERN, it is being stored at CERN CASTOR storage system (steps 1 and 2). SRM interface can be used to access the CASTOR storage system, to preallocate space for the data and to receive the transfer url. A large amount of simulated data is being generated in a distributed fashion as well, and is handled in an analogous fashion.

In order to be useful the data is registered in a Replica Catalog via Replica Registration Service (step3).

Data is replicated using srm copy functionality, when the agent initiating the transfer (Replica Manager) contacts one of the systems, and the storage performs the transfers on behalf of the agent (steps 4 though 9).

Finally, when data is stored in the Tier 2 center close to the final user, user may access it using srm get function. Depending on the user needs user might access data using file based or posix like protocol (step 10).

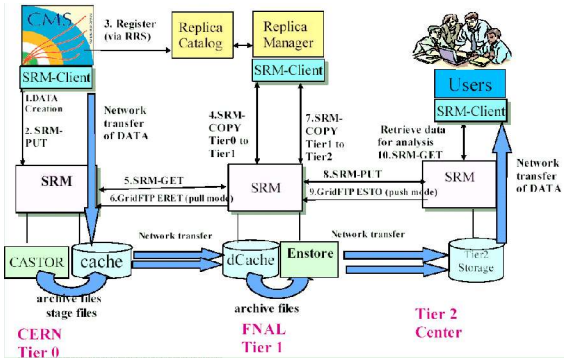


Figure 3: CMS Data life cycle and role of srm in the CMS datagrid.

SRM INTERFACE SPECIFICATION

SRM interface versions

There are two versions of SRM specifications [1,2] that are considered standards, ready for implementations. One is SRM version 1.1 and the other one is SRM version 2.1. SRM v1.1 was finalized in 2001, was implemented by LBNL, JLAB, Fermilab, and CERN. The 1.1 specification mostly consists of the definition of the data transfer functionality. Experience gained by the SRM collaborators during the implementation and production use of srm version 1.1 has allowed to define a more complete management interface SRM version 2.1. The interface specifies various space and files types and the explicit space reservation allowing advanced space reservation and space manipulation functions, in addition to the data transfer functions. It also defines the functionality for namespace discovery and manipulation and user permission modification functions.

SRM interface details

All srm functions can be divided in the following groups:

- Space Management Functions are used for explicit reservation of spaces of various types, modification of the lifetimes of the space reservations etc.

- Data Transfer Functions are used for negotiation of transfer urls, file reservations and system preparation for performance of the transfers, and, in case of srmCopy function, for the performance the transfers by the storage system itself.
- Directory Functions are used for listing of the directory context and for discovery of file metadata. There are also functions that allow the removal of the files and directory names from the storage system namespace.
- Permission Functions allow changing the ownership of the files and modification of various access permissions of the other system users.
- Status Functions are used to discover the status of the file transfer requests issued earlier via data transfer functions.

FERMILAB SRM IMPLEMENTATION STATUS

The major Fermilab storage system that has an SRM interface to it is dCache Storage System, developed jointly by DESY and Fermilab. In Fermilab dCache is mostly used as a disk cache in tertiary storage, consisting of dCache and Enstore tape storage system. DCache can work as a cache on top of various tape systems, including OSM and HPSS. It can also be used as standalone disk based mass storage system.

DCache srm implements SRM version 1.1 and provides all the data transfer functions (get, put and copy), which support load balancing and throttling of the transfers, fairness in the execution of the transfer requests, scalable replication mechanism via gridftp and http protocols and automatic directory creation. The srm functions achieve fault tolerance and reliability by providing persistent storage for transfer requests and retries on failures.

Fermilab implementation of SRM interface is now available as a standalone product, adaptable to work on top of another storage system through a well defined SRM-Storage interface. Fermilab also provides a reference implementation of the SRM-Storage interface to a Unix File System.

FUTURE PLANS

Fermilab SRM development team plans implementation of Space Management functionality, first as an implicit space reservation as a part of SRM Version 1.1, and then as a set of explicit functions, through the SRM V2.1 interface. Then the full implementation of SRM Version 2.1 interface will follow, which will allow the integration of Fermilab Storage into International Lattice QCD Data Grid (ILDG) [13].

Another important development will be the integration with Virtual Organization based Authorization Services and development of storage specific fine grain Authorization Services.

We also see the need of more standard Monitoring, Administration and Accounting interfaces.

Integration with Lambda Station interface to optical networks.

Storage systems managed by SRM interface are capable of direct Mass Storage System (MSS) to MSS transfers over wide area network. Lambda Station [14,15] is a research interface to optical network infrastructure, that dynamically provides optical links between sites. Fermilab plans to research the integration of Lambda Station interface into the SRM middleware. Through Lambda Station interface the SRM layer would be able to perform the network connection reservations, with the capability to discover the properties of such reservations such as the connection throughput and the length of the provisioning. Then SRMs will be able to adjust the number of the simultaneous transfers and the parallelisms of each transfer, thus providing a much batter utilization of the available network resources.

Integration of SRMs into SAMGrid .

"SAMGrid is the shared data handling framework of the two large Fermilab Run II collider experiments: DZero and CDF" [16]. SAMGrid [16] utilizes major Fermilab Mass Storage Systems such as dCache and Enstore, as well as its own SAM Cache software to manage local disc caches. Each type of the systems used has it own management interface and a set of supported transfer protocols. In order to simplify the system and to ease the integration of new types of storage systems into SAMGrid framework, SAMGrid project plans to adapt SRM interface as a universal interface to all types of storage.

REFERENCES

[1] Arie Shoshany at al, "SRM Joint Design v.1.0", http://sdm.lbl.gov/srm-wg/doc/srm.v1.0.pdf

[2] Arie Shoshany at al, "SRM Interface Specification v.2.1", http://sdm.lbl.gov/srm-wg/doc/SRM.spec.v2.1.final.pdf

[3] Patrick Fuhrmann, "dCache, Grid Storage Element and enhanced use cases", CHEP 04, Contribution 233, http://indico.cern.ch/contributionDisplay.py?contribId=233&sessionId=10&confId=0

[4] "DCache, Disk Cache Mass Storage System", http://www.dcache.org/

[5] Alexander Moybenko, "The Status of Fermilab Enstore Data Storage System", CHEP 04, Contribution

[6] "Fermilab SRM Project" , http://www-isd.fnal.gov/srm

[7] "The Storage Resource Manager Collaboration Pages", http://sdm.lbl.gov/srm-wg/

[8] Arie Shoshany, "Replica Registration Service", http://www.ppdg.net/mtgs/28jun04-wb/slides/PPDG-AH-0406-RMS-RRS.ppt

[9] US-CMS, http://www.uscms.org/

[10] Michael Ernst, “Distributed Data Management in US-CMS”, http://www.uscms.org/s&c/reviews/doe-nsf/2003-01/docs/USCMS-Grid-Storage.pdf
[11] Michael Ernst, “Managed Data Storage and Data Access Services for Data Grids”, CHEP 04, Contribution 190, http://indico.cern.ch/contributionDisplay.py?contribId=190&sessionId=7&confId=0
[12] Michael Ernst, “US-CMS Grid File Access Proposal”, http://www.uscms.org/sandc/reviews/doe-nsf/2003-07/docs/GFA-Proposal-Short-v1.0.pdf
[13] “LQCD at Fermilab”, http://lqcd.fnal.gov
[14] Don Petravick at al, “Lambda Station Proposal”, http://hppc.fnal.gov/wawg/omnibus-text.pdf
[15] Philip DeMar, “LambdaStation: A forwarding and admission control service to interface production network facilities with advanced research network paths”, CHEP 04, Contribution 359, http://indico.cern.ch/contributionDisplay.py?contribId=359&sessionId=11&confId=0
[16] Robert Kennedy, “SAMGrid Integration of SRMs, CHEP 04”, Contribution 460, http://indico.cern.ch/contributionDisplay.py?contribId=460&sessionId=7&confId=0