# SAMGrid Integration of SRMs

R. D. Kennedy, A. Baranovski, G. Garzoglio, R. Illingworth, A. Kreymer,
A. Kumar, L. Loebel-Carpenter, L. Lueking, A. Lyon, W. Merritt, I. Terekhov,
J. Trumbo, S. Veseli, S. White, FNAL, Batavia, IL 60510, USA
M. Burgon-Lyon, R. St.Denis, Glasgow University
S. Belforte, INFN, Trieste
U. Kerzel, Karlsruhe University
V. Bartsch, M. Leslie, S. Stonjek, Oxford University
F. Ratnikov, Rutgers University
A. Sill, Texas Tech University

*Abstract*

SAMGrid is the shared data handling framework of the two large Fermilab Run II collider experiments: DZero and CDF. In production since 1999 at D0, and going into production at CDF, the SAMGrid framework has been adapted over time to accommodate a variety of storage solutions and configurations, as well as the differing data processing models of these two experiments. This has been very successful for both experiments. Backed by primary data repositories of approximately 1 PB in size for each experiment, the SAMGrid framework delivers over 100 TB/day to DZero and CDF analyses at Fermilab and around the world. Each of the storage systems used with SAMGrid, however, has distinct interfaces, protocols, and behaviors. This led to different levels of integration of the various storage devices into the framework, which complicated the exploitation of their functionality and limited in some cases SAMGrid expansion across the experiments' Grid.

In an effort to simplify the SAMGrid storage interfaces, SAMGrid is adopting the Storage Resource Manager (SRM) concept as the universal interface to all storage devices. This simplifies the SAMGrid framework, expecially the implementation of storage device interactions. It prepares the SAMGrid framework for future storage solutions equipped with SRM interfaces, without the need for long and risky software integration projects. In principle, any storage device with an SRM interface can be used with the SAMGrid framework. The integration of SRMs is an important further step towards evolving the SAMGrid framework into a co-operating collection of distinct, modular grid-oriented services. This report outlines how the SRMs are being integrated into the existing SAMGrid framework without disturbing on-going operations.

## THE BIG PICTURE

SAMGrid [1] is a general data handling framework originally designed for experiments with peta-byte-sized distributed data processing needs. It has been in production at DZero since 1999 and is going into production at CDF. SAMGrid is also being tested for use at MINOS and CMS.

To better support globally distrobuted data processing in these experiments, there are a number of efforts underway to extend the original SAM framework to the Grid, hence the revised project name SAMGrid. Related to SAMGrid at CHEP04, there are presentations on exploiting Job and Information Management (JIM) for DZero simulation production [2], adapting SAMGrid to a new Monitoring and Information Services framework [3], and the evolution of SAMGrid meta-data services to support more experiments [4]. In addition, there have been some major SAMGrid development projects unrelated to the Grid per se, such as an revision of the middle tier database access component (db server) [5] and deployment of SAMGrid in the CDF environment [6, 7]. The SAMGrid project has enjoyed the contributions of many, as well as the dedication of core developers, to evolve a working production data management system in situ.

While the SAMGrid framework has been adapted over time to accommodate a variety of storage solutions, each of these storage systems has distinct interfaces, protocols, and behaviors. In an effort to simplify the SAMGrid storage interfaces, SAMGrid has adopted the Storage Resource Manager (SRM) concept as the universal interface to all storage devices. After introducing SRMs and SAM, this paper describes the "SAMGrid integration of SRMs" project, its status, and plans for future work.

## INTRODUCTION TO SRM

The SRM is a uniform Grid interface to heterogeneous storage. It also negotiates transfer protocols between a client and a storage device. In the SRM model, client applications may manipulate and access storage through Grid middleware among a variety of existing storage systems with SRM interfaces. Implementations exist, for example, for HPSS, dCache, JASMine, and Castor. In the context of this paper, there are two distinct uses for the term SRM. First, there is the interface specification [8] for all SRMs, an effort in which Fermilab personnel are involved. Second, there is a Fermilab-developed SRM implementation [9] to the dCache storage system [10].

One can think of SRMs as abstracting basic file system

operations. For instance, there are space management operations including an srmReserveSpace(). There are basic data transfers operations like srmCopy(). There are directory operations like srmMkdir(). Permission operations are included, as well as status and metadata query operations.

The Fermilab dCache-SRM team has parameterized the internal storage system interface to allow re-use of the "Grid layer" of the implementation with only modest effort. This should greatly simplify the development of alternative SRMs that are immediately compliant with the SRM interface specification. As a proof of concept, this team has implemented an SRM on top of a plain Unix file system. Other implementations are planned as well, in part to support the integration of SRMs into the SAMGrid project.

## INTRODUCTION TO SAM

The original SAM framework was an early DataGrid [11]. However, since SAM's design predates modern Grid design, its nomenclature and design principles differ with what many might be familiar. The physical architecture of SAM is built around a *station*, which is a collection of resources that are managed together. One can think of a station as DataGrid node. A station might consist of a single host computer with disk, or it may span many hosts in a reconstruction farm. SAM uses a *project* to manage file delivery for one of more *consumers*, or user applications. The original intent for the project concept was to allow multiple consumers to read the same files in a coordinated fashion. That has proven with experience to be a rare use-case, but the concept is retained to shield consumers from file delivery error handling. A *dataset definition* is a meta-data-based specification of the data files a consumer intends to use. A *snapshot* is the actual list of files that match the criteria of a dataset definition at a specific point in time. A *stager* is an agent of the station that actually performs file transfers, in particular to local disk cache for access by consumers. *SAM cache* is a quota-enabled local disk cache managed by the SAM station software. Finally, the replica database component of SAM is a catalog of file replica *locations* in the entire, global SAM system.

The data processing model of SAM is as follows: a user starts a project appropriate for his/her consumers, as well as the consumers, on a station. The project requests delivery of a dataset snapshot (list of files) to local SAM cache disk accessible to interested consumers. Stagers arrange the actual file transfers to that cache. When files are available in local cache, the interested consumers are notified and they may begin processing the files. Information is maintained in the SAM meta-data system to track what files are consumed, what consumers were run, the versions of code used, and so on, to make this process both reproducible and robust. For instance, after running a large job consuming thousands of files, one can query the database to see if any part of the job failed or if any files could not be delivered. If so, then one can run a "clean-up" job which attempts to re-try any part of the job that failed.

## UNIFORM STORAGE INTERFACES

While SAM works well, with real-world experience we have recognized a few issues that the adoption of SRM use may address. SAM was originally designed to support only the local SAM cache as the file "consumption" storage system, and that limitation is deeply ingrained in the implementation. Files can have locations in other storage, in tape-based mass storage systems for instance, but such files have to be transferred to local SAM cache before a consumer can access them. Over time, customers have requested station configurations supporting other "consumption" storage, such as NFS mounted disks, HPSS, dCache, and the AFS file system. An *External Storage Mechanism* (ESM) was developed in SAM to allow these configurations to be supported, but that mechanism led to code and configurations issues that are specific to each storage system.

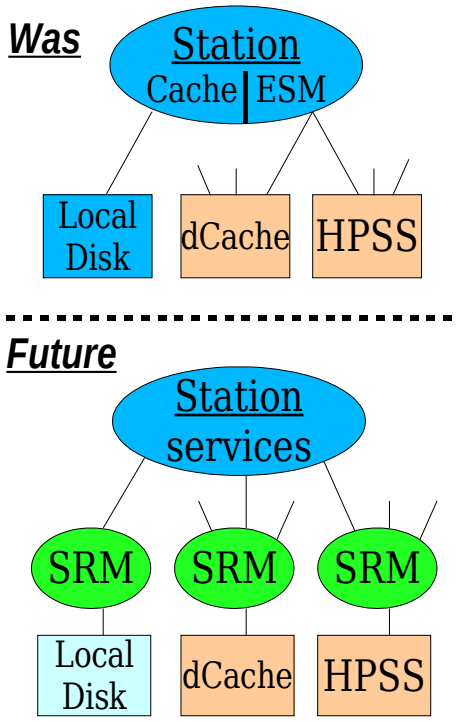


Figure 1: A SAM station and its storage - before and after SRM integration

The SRM provides a uniform interface to all storage, as represented in Fig. 1 on page 2. The adaptation of SAMGrid to use SRMs eliminates specialized code for each storage system, and simplifies the adaptation to new storage systems in the future. It also permits centralized configuration of storage systems, rather than each SAMGrid station having to maintain storage system configuration information. Without SRMs, each CDF station using dCache must be configured with the the host name and port numbers of the CDF dCache dcap doors. If these doors were to change, and they have before, every station would have to

be stopped, reconfigured, and restarted.

While there are clear benefits, SAMGrid integration of SRMs is a heavily constrained project. The SAMGrid system is a production system, in use around the world, and is undergoing much unrelated development at the same time. Any adaptation must be compatible with a gradual deployment plan, allowing old and new station versions to interoperate for an extended period of time. Finally, there is a challenge to identify distinct interfaces of station components since SAM was not originally developed with well-defined internal service interfaces. The data routing service, for instance, does not query the local cache service for its content through a well-defined method or message. Rather, it directly reads the cache content data structure in memory.

## ABSTRACT LOCATIONS

The SAM team has also recognized a subtle problem with the treatment of locations which the SRM may help address. Locations of files in SAM cache include the hardware hostname and file system directory of the file in that SAM cache. These locations are stored in a global replica catalog, depended upon potentially by consumers around the world. Since SAM caches are located on a wide variety of hardware, from robust file servers to low-end IDE disks on farm CPU nodes, from centrally managed Fermilab systems to remotely administered university desktops, this approach leaves the SAMGrid system vulnerable to meta-data inconsistencies due to simple equipment failures, directory re-organization, local system administration policy changes, and NFS mount point variations. Projects may attempt to transfer a file from a no longer valid location, until that location is fixed or removed from the replica DB. Another consequence of the ESM implementation is that a single physical file can have more than one location (for the same transfer protocol) if there is more than one way to access a file in a storage device. This is possible for example where dCache systems have more than one dcap door, as is the case in the CDF dCache system. While this "N locations to 1 replica" is not problematic in general, it causes some problems for SAMGrid at present since SAM does not yet fully separate the concepts of locations and Grid transfer URLs.

The adaptation to use SRMs can help provide more abstract locations by providing something like a layer of indirection in the replica location specification. Instead of the specific hardware host and directory of the file, one specifies an SRM-oriented location by identifying the SRM and an abstract specification of the file in that SRM's namespace. In principle, the physical location of the file *inside* the SRM can change, and yet the file still be accessible to consumers, since the SRM is responsible for internally tracking the physical location of its files. In addition to specifying the SRM and the abstract file reference, the client also specifies what transfer protocols it can utilize. With this, the SRM translates the abstract "namespace-filepath" into a protocol-specific transfer URL, as demonstrated in the example below. In addition to the constraints mentioned before, there are subtleties in how locations affect data management decisions that have to be addressed in the SAMGrid adaptation to use SRMs.

### *Old-Style Location*

cachehost.fnal.gov/hardware/directory

### *New-style Location*

srm://srmhost.fnal.gov:8843/namespace-filepath

### *Resulting TURL*

protocol://storagehost.fnal.gov/transfer-filepath.dat

## SRM INTEGRATION

To manage the risk involved in reworking a production system, we have chosen to integrate the use of SRMs in SAMGrid in several stages. Initially, we have reinterpreted existing features and re-used existing functionality to provide a proof of concept in the SAMGrid context. The question is not whether SRMs work (they do), but whether their benefit will be fully realized in the technical and historical context of SAMGrid. In this approach, we configure SRM-managed storage elements as if they were station "consumption nodes" with a virtual cache disk per access method of some size. These pseudo-station disks and nodes describe a physical cache element (like dCache) accessible by consumers with common data access requirements. The former data processing sequence still occurs, but instead of moving files into local SAM caches, a station configured in this way "moves" the file at the meta-data level into its pseudo-cache and give the consumer a transfer URL pointing back into the actual physical cache element (for instance, a dCache dcap TURL). One can then tune the "size" assigned to the pseudo-disks to control the number of simultaneous transfers or the total volume of meta-data stored for the physical cache element. In the large CDF dCache system, for instance, we may prefer the pseudo-disk size to be less than the actual dCache size (see Fig. 2 on page 4) to improve some aspects of performance at the expense of the station having incomplete knowledge of what is actually "in cache" at any one moment.

## STATUS AND FUTURE WORK

We have the initial stage of SRM integration working with dCache gridftp transfers. We are ready to try other storage elements and transfer protocols (dcap for instance). There does appear to be a small mis-match in the error handling "world-view" between SAM and SRM for handling the case of instantaneous high loads which is being worked on before more extensive deployment. While this work has proceeded more slowly than expected, it is also proceeding

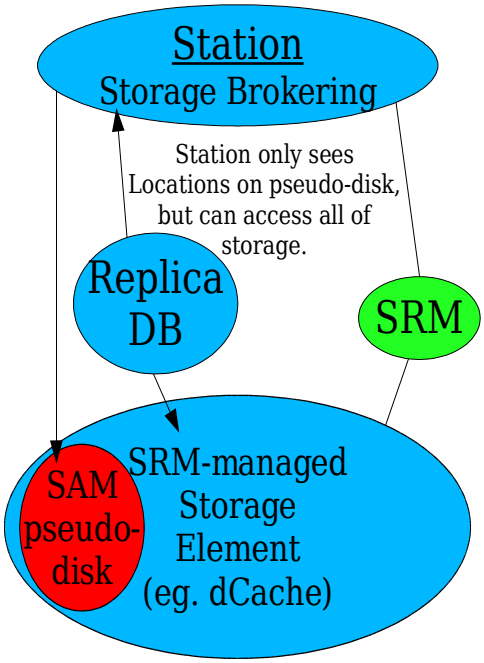


Figure 2: Pseudo-disk size less than actual storage size: station makes decisions based on only the pseudo-disk locations, though all files accessible.

without interfering with SAMGrid operations or other high priority development efforts.

We are beginning to plan the next stage in this integration project. We now have a proof-of-concept in hand to compare future work against as we more profoundly rework the existing SAMGrid station software. We can now begin to modularize the station components into distinct services with well-defined interfaces. Working with the Fermilab dCache-SRM team, we plan to re-implement the local SAM cache component with an SRM interface, something that will be simplified by the SRM implementation on top of a plain Unix file system. Then, SAM will adopt to the modern Grid GUID/SURL/TURL model and integrate the list of transfer protocols available into the data management decisions in the station. We plan to evolve to treat SAMGrid handling data staging and data storage symmetrically, whereas now each are separate services. We do not today see the adoption of a Web Services framework for SAMGrid as providing enough benefit since our existing SAM CORBA infrastructure works well enough. We are however organizing the new station design into a more service-oriented model, which will simplify Web Service adoption in the future should it prove worthwhile.

## SUMMARY

In an effort to simplify the SAMGrid storage interfaces, SAMGrid is adopting the Storage Resource Manager (SRM) concept as the universal interface to all storage devices. This has simplified the SAMGrid framework, especially the implementation of storage device interactions. It prepares the SAMGrid framework for future storage solutions equipped with SRM interfaces, without the need for long and risky software integration projects. In principle, any storage device with an SRM interface can be used now with the SAMGrid framework. The integration of SRMs is an important further step towards evolving the SAMGrid framework into a co-operating collection of distinct, modular grid-oriented services.

## ACKNOWLEDGEMENTS

We would like to thank Fermilab Computing Division for its on-going support of the SAMGrid project, and especially the CCF, CEPA, and Run II Departments. We would also like to thank everyone at D0 and CDF who has contributed to this project. This project is sponsored by DOE contract No. DE-AC02-76CH03000.

## REFERENCES

[1] http://projects.fnal.gov/SAMGrid

[2] G. Garzoglio *et al*, "Experience producing simulated events for the DZero experiment on the SAM-Grid", CHEP'04, Abstract 038, Interlaken, September 2004.

[3] A. Lyon *et al*, "SAMGrid Monitoring and Information Service and its Integration with MonALisa", CHEP'04, Abstract 451, Interlaken, September 2004.

[4] W. Merritt *et al*, "Housing Metadata for the Common Physicist Using a Relational Database", CHEP'04, Abstract 500, Interlaken, September 2004.

[5] S. Veseli *et al*, "The SAMGrid Database Server Component: Its Upgraded Infrastructure and Future Devleopment Path", CHEP'04, Abstract 462, Interlaken, September 2004.

[6] V. Bartsch *et al*, "Testing the CDF DIstributed Computing Framework", CHEP'04, Abstract 113, Interlaken, September 2004.

[7] S. Stonjek *et al*, "Deployment of SAM for the CDF experiment", CHEP'04, Abstract 468, Interlaken, September 2004.

[8] http://sdm.lbl.gov/srm-wg

[9] T. Perelmutov *et al*, "Storage Resource Manager", CHEP'04, Abstract 107, Interlaken, September 2004.

[10] http://www.dcache.org

[11] V. White *et al*, "The Data Access Layer for D0 Run II: Design and Features of SAM", CHEP'00, Abstract C241, Padova, February 2000.