SAM for D0 - a Fully Distributed Data Access System

I. Terekhov, V. White, L. Lueking,
L. Carpenter, H. Schellman[†], J. Trumbo, S. Veseli, M. Vranicar

Fermi National Accelerator Laboratory, Batavia, IL 60510
[†] *Northwesterm University, Evanston, IL, 60208*

Abstract. The SAM (Sequential Access through Meta-data) system is being built as a distributed cache management and data access layer for the D0 experiment at Fermilab. The innovation of the project is the fully distributed architecture of the system which is designed to be deployable worldwide. It uses a central database for the meta-data and a hierarchy of CORBA servers for the actual data movement. SAM provides distributed disk caching, data routing and replication and therefore has components attractive to the Grid.

OVERVIEW

The SAM (earlier, alternative expansions include Sequential Access Method) system is the data access layer for the D0 experiment at Fermilab. During the next several years, the D0 experiment will store a total of about 1 PByte of data, including raw detector data and data processed at various levels. However, it is not the mere size of the data that presents the challenge; as any modern High Energy Physics collaboration, D0 is a truly distributed institution: it includes over 550 scientists from some 65 institutions scattered worldwide.

The SAM project (1) has taken an innovative approach to the data management by designing and implementing the system, from the outset, as a distributed mutli-tiered client-server system deployable worldwide. In the present paper, we concentrate on the distributed nature of the architecture, and leave out the few D0-specific aspects of the projects. This presentation is in the spirit of the Particle Physics Data Grid, where SAM is the FNAL's participant (2).

Fundamentally, the goal of SAM as the data access system is to facilitate physicist access to their data, while efficiently using the scarce hardware resources such as Mass Storage Systems (MSS), networks and computers with their disks, CPU's etc. The bulk of the data is planned to be stored in the MSS's, primarily Fermilab's Enstore (5). The primary functionality of SAM is to provide a globally distributed disk cache; this distributed caching, which involves Grid features such as file routing and replication, is the focus of our paper. For a more general overview of the project, see (3).

THE APPLICATIONS SERVED BY SAM

In the sequential mode, a user application (a physics analysis or reconstruction program, or a Monte-Carlo event generator) processes data in a stream, accessing each data unit exactly once, the order of data units in the stream being irrelevant. The units of data are laid out sequentially in files. Consider file data retrieval, or consumption, in SAM (production, or storage is symmetric.) The application requests a next file from the stream, using a call to a server called project master (4). The application is notified via a callback when a file is ready, i.e., has been physically placed (or staged) on a local disk [1].

The application then uses the standard UNIX system calls to open, read and close the file, at which point it requests another file. The delivery of data to consumers is done by project master and other servers asynchronously with respect to the actual consumption. In a smoothly functioning system, the next file will have been delivered by the time the consumer finishes its current file. There are other interesting aspects of the process, for which we refer the reader to (4); for example, the consumer may in fact be spread across multiple processes perhaps running even on different machines (thus making a SAM consumer a distributed concept itself).

[1] In the Grid, other mechanisms exist to access files. Globus (6) provides specialized library to handle URL-based file names, Condor (7) allows transparent redirection of local file operations to remote servers.

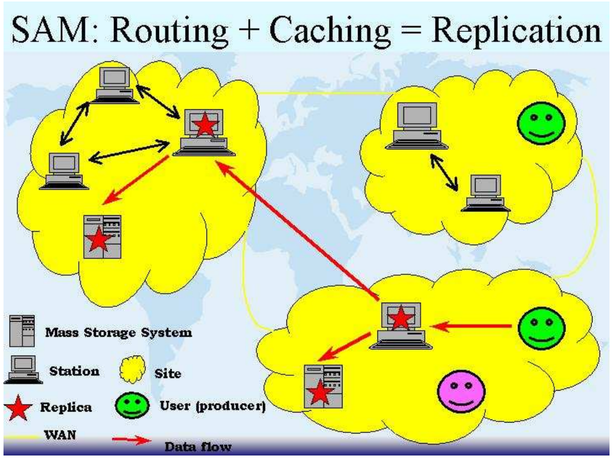

FIGURE 1. File routing while storing data with SAM.

DATA ROUTING IN SAM

For our presentation, it is important that the user reads (writes) a local file, pulls it from (pushes to) the SAM system, by name, and is not concerned with the way that the system relocates the file. The functionality of the distributed caching is to move the data through the disk cache, making multiple copies if necessary, between the user applications and the MSS's. In the most interesting case, a file may come from (or go to) a remote MSS, thus requiring multi-stage transfers of files for a given user. We say that such transfers require *routing*, which is an important aspect of SAM's distributed caching. There are at least two reasons for such routing:

- The MSS may not be directly accessible from the user's location, because of a slow or unreliable network connection or low-performance disk;
- It is in general desirable to leave a copy of a file at "nearby" location(s) for subsequent access

For the purposes of routing, we consider the distributed caching network, exemplified in Figure 1. Nodes on this network are MSS's and *stations*. A station in SAM is a collection of local hardware resources, such as disk and CPU, used for data processing. It is a distributed concept in itself (in general, it spans a relatively homogeneous local cluster), but it is beyond the scope of this paper so we consider the station a point-like node on the global network. Nodes are connected with Wide-Area Network (WAN) links.

As shown in the figure, when a user requests to store a file in an MSS, the file in general travels through one or more intermediate stations. At each station, disk cache allocation takes place. This allocation is subject to the local station's rules (we will elaborate below). While the file resides at an intermediate disk awaiting further transfer it actually is a part of the cache and is of course available for retrieval at this (and other) stations.

By the time of file reaching its final destination, there may be multiple copies of it left on caches on the path. As a result, by routing the file through the network of stations and using caching at each node, SAM performs the dynamic *data replication* service, an important aspect of the Grid. By configuring the routing information stored in each station the global system administrators have control over how the replication takes place.

DISK CACHING AND DATA REPLICATION

We stress the fact that the data replication in SAM is dynamic and naturally occurring. Each file replica is a part of the respective station's cache and as such is subject to the local caching policies which we will briefly describe now. A station continuously replaces files that are not in use by newly requested files. (Since multiple groups and types of users access the station's resources, multiple, perhaps different, algorithms are possible for different virtual parts of the cache.) A cached file is in use when it is either being routed on the distributed cache network, or is being served to the local projects (SAM's term for the activity of data consumption)(4). Thus, in general, a replica will be deleted if it has not been in use long enough and if the disk space is claimed for other purposes. Similar to the routing, by configuring the cache replacement algorithms and allocating cache for different user groups or types[2], the local system administrators control the lifetime of a replica and therefore affect the global replication process.

Occasionally, such dynamic caching and replication may be undesirable for some critical applications requiring that all their files be local for immediate access. SAM provides means for an administrator to lock a dataset on disk statically. Static replication is an effective reservation tool, but in general it sacrifices overall efficiency as it may result in unused data occupying the cache.

Implicit in our discussion so far has been a global *replica catalog*, also known as a catalog of file locations. This catalog is updated every time a file enters or leaves disk cache (or an MSS copy is created). The catalog is consulted in the course of file retrieval, for finding the best replica. SAM implements this catalog as a part of the meta-data database (the "M" in "SAM").

---

[2] SAM categorizes data access by *access modes*. These types, or modes of access play a primary role in global resource management, not covered here (1).

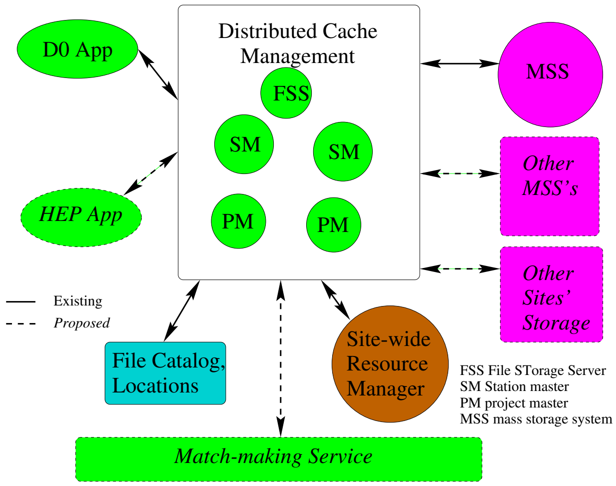

FIGURE 2. SAM's distributed caching in the grid picture.

SAM IN THE GRID

SAM is a highly modular and flexible system. In Figure 2, we show how its distributed caching framework can be used in the Grid. First, it can serve arbitrary application performing sequential access to the data (on the left). Second, it may use multiple abstract MSS's for transferring files (on the right). Third, SAM can be interfaced to an external replica catalog, such as Condor's match-making service (7) (at the bottom).

SUMMARY

In this paper, we have described distributed caching in the Sequential Access through Meta-data (SAM). We have outlined caching with the emphasis on file routing and replication. SAM is a real system, with many components having been in production since year 1999. At the time of writing this paper, SAM is however in active development; we encourage the reader to check SAM's web pages for its evolving design and project status.

REFERENCES

1. The SAM project home page `http:d0db.fnal.gov/sam/`.
2. The Particle Physics Data Grid, Fermilab page `http://grid.fnal.gov/ppdg/`.
3. L. Lueking *et al*, The Data Access Layer for D0 Run II: Design and Features of SAM, presented by V. White at *The International Conference on Computing in High Energy and Nuclear Physics* (CHEP 2000), February, 2000, Padova, Italy.
4. I. Terekhov and V. White, Distributed Data Access in the Sequential Access Model at the D0 Experiment at Fermilab, in *The Ninth IEEE International Symposium on High Performance Distributed Computing*, Pittsburgh, Pennsylvania, August, 2000 in Proceedings.
5. The Enstore project home page `http://www-isd.fnal.gov/enstore/`.
6. The Globus project home page `http://www.globus.org/`.
7. The Condor project home page `http://www.cs.wisc.edu/condor/`.