Fermilab FERMILAB-TM-2175 July 2002

SAM Managed Cache and Processing for Clusters in a Worldwide Grid-Enabled System

Andrew Baranovski, Diana Bonham, Gabriele Garzoglio, Chris Jozwiak, Lauri Loebel Carpenter, Lee Lueking, Carmenita Moore, Ruth Pordes, Heidi Schellman, Igor Terekhov, Matthew Vranicar, Sinisa Veseli, Stephen White, Victoria White*

Fermilab, Batavia, Illinois, USA
**Northwestern University, Evanston, Illinois, USA*

April 12, 2002

Abstract

SAM has been developed within the Computing Division at Fermilab as a versatile, distributed, data management system. One of its many features is its ability to control processing and manage a distributed cache within a cluster of compute servers. Requirements, concepts, and features of this system are described and issues involved in interfacing it to several batch systems are discussed. The system is used within the Dzero experimental collaboration to distribute hundreds of Terabytes of data for processing and analysis around the world. Several hardware configurations deployed at Fermilab are described. Data is currently disseminated using this system to over two dozen sites worldwide, and this number will grow to nearly one hundred in the coming years. The planned design evolution to accommodate this growth is discussed, and the transition of the system to grid standard middleware is described.

1. Introduction

The SAM [1-4] system was developed at Fermilab to accommodate the high volume data management requirements for Run II Physics, and to enable streamlined access and mining of these large data sets. SAM stands for "Sequential Access to data via Metadata" where the "sequential" refers to the layout of physics events stored within files, which are in-turn stored sequentially on tapes within a Mass Storage System.

SAM is largely devoted to transparently delivering and managing caches of data. It is the sole data management system in use by the Dzero experiment at Fermilab; other experiments are either considering or planning its use in the near future. SAM is designed with a distributed architecture, using CORBA as its underlying framework. This has enabled the system to scale to meet the data distribution needs of the Dzero Collaboration which includes 60 institutions and over 500 Physicists located all over the globe. Metadata and configuration information for the entire system is currently maintained in a central Oracle database repository at Fermilab, and as the system matures, it is planned to distribute this function in a way that will reduce latencies and make the overall operation more reliable. Data from the Dzero detector and Monte Carlo simulations are likely to require more than a half Petabyte of permanent mass storage within the next two years. Mass Storage Systems (MSS) located at Fermilab and collaborator sites worldwide are integrated into the system to meet this need. The MSS at Fermilab is called Enstore; additional information on this can be found in reference [5].

2. *Overview of the SAM System*

The SAM data management system is used for all Dzero data cataloging, storage, and access. Sam provides distributed data storage and access to processing and analysis clusters within the Fermilab campus and to collaborating sites throughout the world. On compute systems or clusters where SAM is deployed, referred to as SAM "stations", management is provided in three key areas: 1) disk cache, 2) job dispatch, 3) physics group allocation of all data and compute resources. The system is used to catalog each data file produced by the experiment, along with the metadata that describes it. When files are needed at a particular station, they are replicated to cache disks managed by the station server at that site. Stations can take on many forms: single cpu desktops, multi-processor SMP machines, or hundreds of Linux boxes in a distributed network (farm) configuration. The SAM station server has many functions, but primarily it manages the cache, and brings processing and data resources together both temporally, and spatially. This means that a job is not allowed to run until the data needed for it has been cached, and if data pre-exists the job will be sent to the processor nearest the data.

The architecture of the system is illustrated in Figures 1 and 2. The server elements which comprise each station are illustrated in Figure 1. Cache disks over which SAM is given exclusive control are managed by the *Station and Cache Manager*. This element is also responsible for starting and communicating with *Project Masters*, which are in turn responsible for presenting data files to the user jobs that consume them. If the files already exist in the local cache, their locations are passed to the consumer and the files are locked in place while being used. If the files are not present in the local cache, they are brought in by *stagers* from other stations or MSS, replacing files which are no longer needed in the local cache. The *File Storage Server* (FSS) manages the storage of files. When a request to store a file to the MSS is made, the description of the file is added to the SAM metadata, and the job is placed in a queue for copy to the designated MSS.

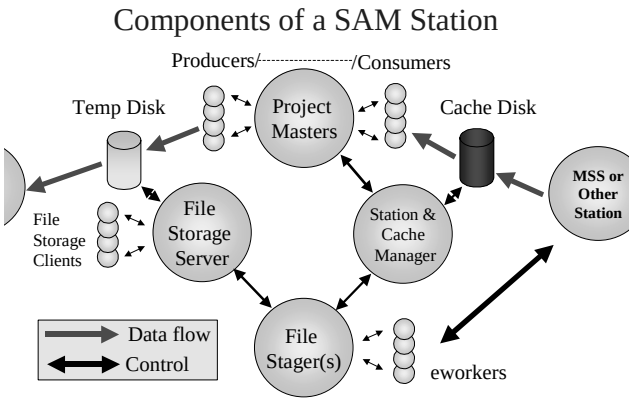

Figure 1. SAM Station components (see text for details).

SAM stations are network-distributed (Figure 2), and can receive data from, or route data through, other stations. Stations can also store data to local (relative to site) hierarchical storage systems or forward data through other stations for storage in remote hierarchical storage systems. In the current architecture there are several services shared among all stations; these include: 1) the CORBA name service, 2) the central Oracle database, 3) the *global resource manager*, and 4) the *log server*. The name service is the "switchboard" to register and receive addresses for the entire distributed system. The central database contains all metadata for each file registered with the system, as well as station configuration, cache, and operational information. The global resource manager reviews all requests for all stations and optimizes file deliveries. The log server receives logging information from all stations and records them in a central log file.

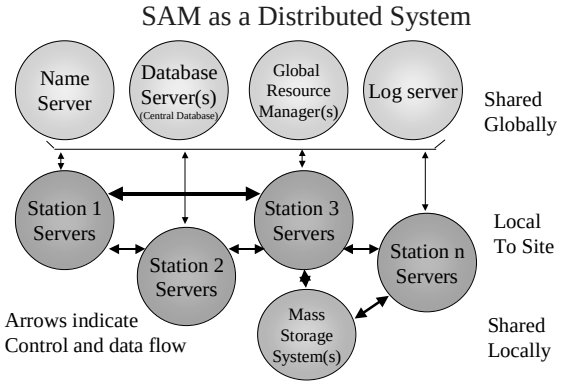

Figure 2. Overview of SAM distributed components.

3. Resource Management

In the SAM framework, a job, or *project*, is defined as the processing of a *dataset* with a particular user *application*. Therefore, when the project is submitted to the station a complete evaluation of the data and computing resources can be made. In order for the application to begin processing, at least part of the dataset must be present in the station's disk cache. The job control and data management must be coordinated in order to provide fair-share resource allocations to each group, and to maximize the overall work being accomplished by the system. These two goals employ, respectively, Fair Share scheduling and resource co-allocation. For a more detailed discussion of the implementation of the Fair Share and resource co-allocation refer to [6].

The Fair Share (FS) system uses the concept of a *benefit* to maintain a history of the resources used by each group. This history is then used to calculate the group's resource usage relative to all other groups defined on the station. The total benefit accumulated for each processing job is a linear combination of *benefit types*, each with an assigned weighting factor. Any quantity that increases monotonically with each job run can be treated as a benefit type. Currently, benefit types include: CPU, memory, number of cached files used, number of staged files used, number of tape mounts used. At the time of job submission the group's fraction of the cumulative benefits for all groups is determined. If the groups benefits do not exceed its FS allocation, the job will be scheduled ahead of other jobs, and vice versa.

Resource co-allocation refers to the simultaneous management of data and processing resources. When a user job is submitted to a station master, the input dataset is translated into an explicit set of files that will be needed for processing. Files that are already in the cache are locked for the duration of the project, and files that need to be brought from other stations' caches, or from mass storage, are queued for staging. The station cache replenishment for the job is done asynchronously from the processing of the job in a manner that minimizes the overall processing time. The initial dispatch of the job is coordinated with the existence or delivery of some minimal amount of data. On the other hand, if files are delivered to a station and there are no processing queues available, the data might be removed before it is accessed, or a deadlock condition might be established which ties up resources without any useful work being accomplished. This situation is avoided by a timeout parameter that stops the project and precludes the queue being tied up unnecessarily. In the case of a station configured with a distributed cache, SAM also attempts to initiate applications on nodes where requested data already exists in the local cache.

3.3 Distributed Cache Management

There are two aspects of the distributed nature of the disk cache management in SAM; local and global. The cache is *locally* distributed in the sense that a *station* (introduced above) manages resources of a cluster, including a collection of independent file systems – disks. The cache is *globally* distributed in the sense that these stations in turn form a global network, or grid, of caches.

As described in the next subsection, the jobs controlled by the station on a cluster are distributed. These jobs process locally cached data, i.e., user applications do not perform any remote I/O. Thus, the station relates the jobs and their *consumption sites* (essentially nodes where the actual processes run) to the disks *accessible* from these sites in order to ensure that files are brought to their consumers. For faster, smoother processing of datasets in a stream-like fashion, the station does not wait until a consumer becomes "hungary" Rather, it works pro-actively and pre-stages files at the appropriate nodes in anticipation of consumers requesting them. The *project*, i.e., the input dataset for a job, is known upfront to the station.

Naturally, fluctuation of the relative performance of the various processes in a project will result in some consumers becoming "hungrier" than others. The station uses sophisticated algorithms to rebalance dynamically the load so as to ensure the maximum project throughput and to avoid ready consumers occupying a process slot waiting for data that had been over-delivered elsewhere. This feature of eager intra-cluster replication may result in excessive network load. To address this issue, we configure each station such that the described behavior can be modified within a broad range. On the one extreme, highly CPU-intensive farm reconstruction jobs require the highest utilization of the CPU and always having data ready for the active jobs wherever the batch system can find an idle machine, whereas network usage is

hardly an issue. On the other extreme, relatively fast, I/O intensive analysis jobs do not always require CPU utilization but are expected not to swamp the network and thus "lazier" replication takes place. All these requirements are accommodated by the appropriate choice of the configuration parameters. There are other, more advanced, but not fully implemented features of cluster cache management, such as designated I/O (gateway) nodes for efficient retrieval of data onto a cluster for subsequent intra-station replication, or *cache accessibility matrix* for partially shared file systems.

As to the global aspect of the stations, it is critical that the stations are fully capable to exchange data. First, the data may be stored from any station in the D0 network to any Mass Storage System (MSS). For performance and other reasons, these MSS' are not directly accessible by every station; in fact, they are typically accessible to a few stations within their corresponding site. Thus, the data must be *routed* from the station initiating a file store to its final destination (MSS). In the course of such a global store, data is dynamically replicated at the intermediate location(s) possibly facilitating subsequent retrieval. We have therefore implemented a *grid replication service*. Retrieval from a remote MSS or other location may also proceed in stages (limitations exist in the present implementation but are being removed such that full symmetry between store and retrieval takes place). Stations that need files for their consumers apply administratively configured *replica selection* criteria to choose the best location for a file. At present, replica selection is based on ordered lists of *preferred* and *avoided* locations. In the future, we will need to base this selection on the dynamically changing resource management considerations.

3.4 Distributed Process Management

As we have described above, stations run data consuming (and producing) *projects* for their users. A project may be viewed as a locally distributed, coordinated consumption of the input data, pre-staged for it dynamically by the station. There are, in general, multiple processes for the same *consumer*, as well as multiple consumers in the project. A consumer is a logical grouping of the processes wanting to process, in any order, a dataset. It is the job of the *project master* process to ensure that each file from the dataset is "seen" exactly once by one of the processes of each consumer. Thus, physics results from an analysis are reproducible as long as one assumes no particular order of the events in the stream. The processes are typically started by the batch system; SAM is non-intrusive as far as deciding where these are created. The project master adapts to these dynamically created and destroyed processes and communicates the respective information about its consumption sites to the station master. For example, if the batch system decides to run two processes of the job on one node of the cluster and one process on another, the project master and the station master will, transparently to the user, ensure that the first node gets roughly twice as much data as the second node.

SAM will gracefully recover from a cluster node becoming unavailable in the middle of a project. At any time after such a failure other nodes will presumably continue to need more data, and the station master will re-route the data to the seemingly alive nodes, such that the healthy processes do not feel any data shortage. Some data may still be attempted for deliveries to such a node. After a configured time-out, the project master will detect inactivity at the node and will signal a "definite out" update of the consumption sites to the station master such that all the data will be re-routed (or re-delivered) to the remaining nodes of the distributed job.

It should be noted that, while farms and some analysis clusters run fully distributed data processing jobs, not all the HEP applications are readily parallelizable. Physicists often face challenges combining results of distributed analysis. These issues are beyond the scope of this paper, but when required can be accommodated, with extra care, within a system that supports data and resource handling for distributed jobs.

4. Interaction with Existing Batch Systems

In order to maximize the overall throughput, as well as to accomplish the fair-share resource allocation for different physics groups, SAM integrates user job control with data management. When submitting their

jobs to SAM, users do not need to know specific details of the batch system (or systems) deployed on their cluster. Instead, they use a generic *sam submit* command which defines the input set of files to be processed (*dataset*), the name of the application that will process these files, and any additional application or batch-system parameters (such as job description files, etc.). This triggers the staging of the input files for the project, as well as the actual job submission to the underlying batch system. Although the batch system takes care of the job scheduling, SAM may suspend its execution until all of the job's SAM-related requirements (e.g., data availability) are met.

Interaction with the different batch systems is handled through batch system adapters. The batch adapters know how to submit a job to a specific batch system, how to suspend a job or to impose additional requirements which must be met in order for the job to be executed, how to notify the batch system that those requirements have been met and that the job may be resumed, and how to inquire about the status of a running job. At present, SAM provides adapters for LSF [7], PBS [8], FBS [9], and Condor [10], but adding others is fairly straightforward.

We would like to emphasize that the integration of data delivery with job management is one of the novel features first introduced by SAM. In spite of its importance for the data-intensive applications, this issue has never been fully addressed before.

5. Operational SAM Cluster Examples

To demonstrate the range of possibilities, we describe several large clusters that have been deployed at Fermilab. Each one uses a different configuration to match resources and needs, showing the flexibility of the SAM model. The Dzero Reconstruction Farm station controls some 130 dual processor nodes in their processing of Dzero detector data, using FBS for the batch system. Central-Analysis is the SAM Station on Dzero's main analysis machine. This station manages over 13 TB of disk cache on a 176 processor SMP and employs the LSF batch system to schedule jobs. Central-Analysis-Backend is comprised of the same large Unix machine along with backend Linux nodes to provide economical processing power, and schedules jobs using PBS. The ClueD0 SAM Station is a Linux PC cluster made up of nodes contributed by various Dzero collaborators as desktops, and also uses PBS as the batch system.

5.1. Dzero Reconstruction Farm

The Dzero reconstruction farm is an example of a cluster designed for high CPU, low data I/O, grossly parallel processing tasks. It is comprised of an 8 processor SGI Origin 2000 "Master" node, and 130 dual processor Linux "worker" nodes. The SGI runs the station servers and manages a half TB of output disk cache. Each worker node has a small (<= 10GB) cache of local disk space. The farm handles all the reconstruction of raw data created from the Dzero detector, at a rate of nearly 500 GB per day. This processing is very CPU intensive, requiring about 8 hours of CPU time per 1GB file. With small local caches, long file processing times, and a high price on CPU time, the SAM Station is configured to aggressively replicate data to minimize CPUs waiting for file delivery. All the nodes in the system are connected through a high speed switch, and network transfers between nodes are considered to be "inexpensive".

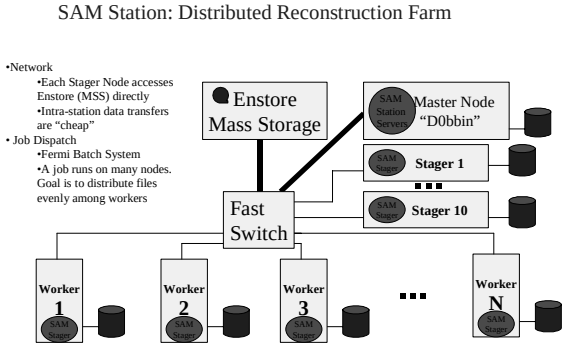

Figure 3. Dzero Reconstruction Farm Configuration. Data is directed through Stager nodes and replicated to the workers for processing.

With the small cache sizes at each node, it is possible to "overfill" individual caches, which can lead to dead-lock or delays in delivery. Because of the necessity to maintain maximum CPU utilization of the farm, the SAM Station must avoid this. To maximize the station's ability to stream data from the Mass

Storage System, and to minimize idle CPU time, the farm Station utilizes an additional ~10 nodes as file servers through which external deliveries are constrained. This way, many files can be "pre-fetched" at once onto file-server disk space, without filling worker node cache. The Station can then very efficiently deliver files only to nodes where CPU is available. The farm station aggressively replicates data to worker nodes to ensure that CPUs are not kept waiting; it also aggressively removes replicas that are delivered too early in order to make room for immediately necessary files.

The Dzero reconstruction farm uses FBS as its batch system, which has proven to work very reliably with the SAM Station. Farm administrators use job description files to specify the nodes on which the jobs will run, and have very tight control over scheduling to maximize CPU efficiency.

5.2. Central-Analysis and Central-Analysis-Backend

Central-Analysis is the main local analysis SAM Station for Dzero at Fermilab (Figure 4). It runs on a large 176 processor SGI Origin 2000 SMP called "D0mino" and has control over 13 TB of disk space. Six Gigabit Ethernet network interfaces are used to transfer data to and from other network sources, including the MSS. This SAM Station is confined to one machine, but its large cache size and large bandwidth to the Mass Storage at Fermilab make it ideal for providing users quick access to large datasets. It has become the most important SAM Station at Fermilab and reliably manages upwards of 40,000 files for hundreds of users. Central-Analysis utilizes the LSF batch system to manage all the jobs on D0mino.

In order to augment the compute resources additional Linux PC "compute servers" are being added to the D0mino installation. Another station named Central-Analysis-Backend is made up of the same large SGI machine, as well as 16 "backend" dual processor Linux worker nodes to provide more computing power for CPU-intensive analysis jobs. This configuration allows for adding low-cost processors while maintaining many of the advantages of the Origin system, including its ability to handle large numbers of logins, fantastic I/O capacity, and enormous disk storage. The Central-Analysis-Backend station is configured to have all external deliveries routed through D0mino, and all users submit their jobs from D0mino.

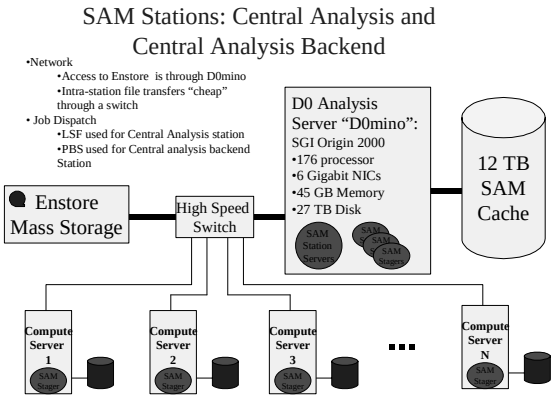

Figure 4. Dzero Central Analysis and Central Analysis Backend Stations. All data is brought in through D0mino.

This configuration provides convenience for users of the system. Someone wanting to do small jobs on a large number of files can log into D0mino and submit their project to the Central-Analysis Station. The job is forwarded to queues on the LSF batch system, and the Central-Analysis Station can quickly give the job the files that are already on the large amount of cache, or quickly retrieve them from Mass Storage. Users desiring to run a more CPU intensive jobs can log into D0mino and submit their project to the Central-Analysis-Backend station, specifying the number of backend nodes they would like to use. The backend station will begin delivering files to the staging area on D0mino. The Station also submits the actual jobs to the PBS batch system which manages the scheduling of processes on the backend nodes. Once processes are started on the worker nodes, the Station delivers files from the staging area on D0mino to the correct worker nodes to feed CPUs as they become "hungry".

Initially, the SAM Batch Adapters were designed to pass the user's environment to the batch system along with the job, so that the job would run on the execution node with the same environment the user had setup at the time of submission. The PBS batch system, however, has an upper limit on the size of environment it can handle. We found that the D0 analysis tools code requires a very large environment that exceeds the PBS limit. Therefore, for this specific batch system used with both the Central-Analysis-Backend Station and the Clued0 Station, the Batch Adapter had to be changed to not pass the user's environment. The actual job then has to re-establish this on its own at the execution node.

5.3. ClueD0

Another variation of SAM station with a network-distributed set of processing nodes is ClueD0 (Figure 5). This station consists of one Linux file server and over 100 satellite worker nodes. The file server has a Gigabit Ethernet interface to the network, 640 GByte of physically attached disk, and it is given access to the MSS and other stations at Fermilab. The worker nodes are Linux boxes used as desktops, located in reasonable proximity to the server and connected to the network with 100 Mbit Ethernet, but not given direct access to the MSS or to other stations. These satellite nodes have various CPU and disk resources, but their systems are all managed in a consistent manner. The total cache disk deployed throughout the station exceeds 5TB. The station is configured so that the cache disks are physically distributed among all of the satellite nodes, but managed by a single station master. Each worker node has its own SAM stager which communicates with the station master. Analysis projects requesting files on any one of the worker nodes will be satisfied by one of the following actions: 1) if the file is already resident on the node it will be used, 2) if the file exists in the cache on another node in the cluster it will be copied to the local node, or 3) if the file is not in the cache of the ClueD0 station it will be brought in, through the file server node, from another station or from the MSS.

Jobs are submitted through SAM to the PBS batch system, similar to the Central-Analysis-Backend configuration. Because the hardware is contributed by different institutions, the batch system is configured to schedule jobs based on the portion of ClueD0 resources provided by the user's home institution.

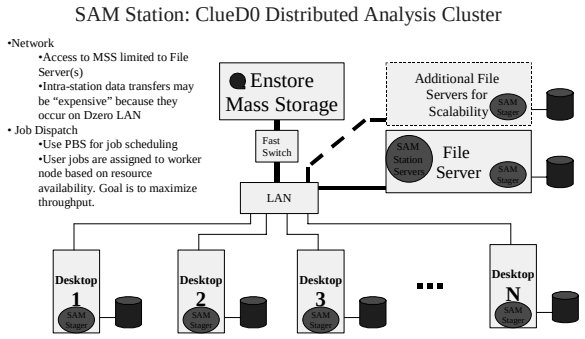

Figure 5. ClueD0 analysis cluster. All data is brought in through File Server node(s) and replicated to desktop worker nodes as needed.

With a cluster in this type of situation, where a fraction of the nodes are equipped with SAM cache disks, SAM jobs can run only on a subset of the entire cluster managed by the one PBS batch scheduler. It is necessary for the SAM Station to submit jobs to the batch system with information about nodes on which it can be successfully executed. PBS configuration (as well as other batch systems configurations) allows giving arbitrary properties, or attributes, to nodes in the cluster. Jobs submitted to the batch with the requirement of a given property will only be scheduled on nodes with the given property. The SAM Batch Adapter submits jobs to PBS requiring the property "SAM", and the cluster administrators are able to configure PBS to acknowledge which nodes are SAM-capable. This allows for an easily expandable and dynamic set of nodes that make up the SAM Station.

6. SAM as a Globally Distributed System of Clusters

There are currently over two dozen operational production SAM stations deployed at Fermilab and remote institutions. There are six major processing centers that have been using these stations for nearly two years to forward Monte Carlo data to the central tape storage system at FNAL. We anticipate the deployment to continue over the next two years when each collaborating institution will operate one or more SAM stations. In addition, there will be special Regional Analysis Centers (RAC) with high speed network connections to Fermilab, and to each other which will have additional resources to cache and serve large quantities of data. These RACs will in turn provide the data to lower tier sites aligned geographically, politically, based on physics interests, or other criteria, to share data for processing and analysis at sites equipped with high capacity network connections to each center. This is controlled by configuration information stored for each station that is effectively a "static route", specifying from which stations it is allowed to get data files. These routes will be set up such that they first try other stations within their local domain, and then the RAC. It is imagined that the data selection managed at each RAC will be based on which data is most actively used at the corresponding analysis sites. If there are certain data sets that are required at the RAC, they can be loaded and "pinned" in the cache.

Fermilab (5 stations)	**Batavia, IL**
Imperial College (2)	**London,UK**
IN2P3	**Lyon, France**
Lancaster	**Lancaster, UK**
Munich	**Munich, Germany**
NIKHEF	**Amsterdam, NL**
Prague	**Prague, Czech Rep.**
Wuppertal	**Wuppertal, Germany**
Boston University	**Boston, MA**
University of Arizona	**Tuscon, AZ**
U. Texas, Arlington (2)	**Arlington, TX**
U. Oklahoma, Langston	**Langston, OK**
Indiana University	**Bloomington, IN**
Louisiana Tech	**Ruston, LA**
University of Kansas	**Lawrence, KN**
Michigan State University	**East Lansing, MI**

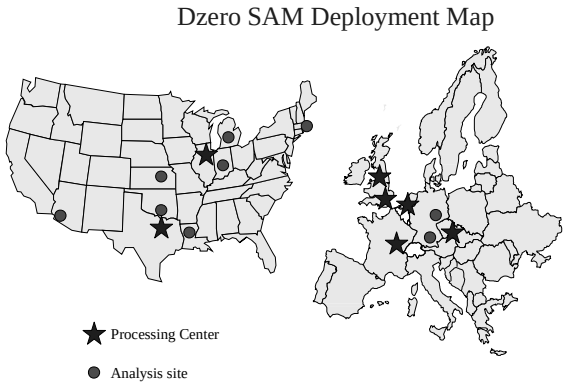

Figure 6. Dzero Sites where SAM Stations are deployed.

7. Future Improvements and Transitioning to Grid-Standard Middleware

Although the system is working quite effectively now, we have plans to improve and streamline the operation in the next two years. The station cache manager and file storage server (FSS) components will be consolidated so that the FSS utilizes the general station cache. This will make management of temporary cache used by the FSS more robust, and will allow the movement of files through the station cache en route to and from an MSS to be more easily controlled. The system will move toward a much less centralized model in which stations have autonomy and are capable of operation without contacting the central database at Fermilab for long periods. Each station (or possibly site) will have its own information service which will track operational information for the station, such as cache history and project activity. Each site's information service will also track station activity in a way that will allow a global information service to access the activities for all stations and monitor the overall health and activity of the station network.

This decentralization will remove the current single-point-of-failure and greatly improve performance of the system as it is scaled to the larger world. It is also part of the natural progression of the system toward a "standard" Grid system. We are moving toward standard grid middleware wherever possible. We will soon be using components form the Globus [11] toolkit including Grid Security Infrastructure (GSI) and GridFTP. We plan to provide standard interfaces to our data that will include those used by Storage Resource Manager (SRM)[12], a possible emerging standard in the grid world. This will enable even non-SAM enabled users to access Dzero data by providing a url and file name. We are working with Condor and Condor-G[10] as possible tools for Grid job scheduling and resource management. Users will be authenticated and authorized using standard grid protocols. This compliance with standards is very important at Fermilab and at many of our collaborating institutions, especially those where computing resources are shared by multiple experiments.

Effectively using these resources will be the responsibility of the data handling system that evolves from a hybrid of the current SAM and more global Grid systems. Job definition and submission information will be sent to a grid resource broker that will determine the best host(s) for each job. Based on the resource broker's "decision" each job will be sent to a compute system with available resources and given to the local job scheduler. For systems under the control of SAM software, the local scheduler will be coupled to the sam station management, and the data and computing resources will be brought together to complete the task. For systems not under the control of SAM, standard Grid interfaces will be provided to dispatch the job to the local job scheduler, and bring data to the system though a Storage Resource Manager.

8. Conclusion

The model designed for the SAM Data handling system has been extremely successful in enabling the Dzero Experiment to establish many useful computing configurations. In the SAM architecture large numbers of compute and data storage resources can be managed in a coordinated fashion by a set of SAM station servers. Users can submit jobs to a particular station, and data and processing are brought together in a way that most effectively utilizes the available resources. We have deployed the system in several configurations ranging from a large SMP central server, to completely distributed Linux pc clusters. A wide range of data processing challenges have been met for the experiment ranging from highly CPU-intensive event reconstruction, to highly I/O intensive data analysis. The performance, scalability, and robustness of the system are constantly being understood and improved. In addition to specific cluster installations, SAM stations of various configurations are installed worldwide as a grid-enabled system. We are working to understand current grid technologies and software. We are designing additional Grid resource management features and will transition the system to utilize grid middleware as it becomes viable.

Acknowledgements

We would like to thank the Fermilab Computing Division for its ongoing support of SAM, especially the ODS, D0CA, and ISD Departments. We would like to thank everyone at D0 who has contributed to this project, and the many important discussions we have had there. This work is sponsored by DOE contract No. DE-AC02-76CH03000. Additional funding for Grid related SAM work is provided by the DOE SciDAC program through Dzero's association with the Particle Physics Data Grid (PPDG) collaboration.

References

[1] The SAM team, L. Lueking (co-leader), V. White (co-leader) , A. Baranovski, C. Jozwiak, L. Loebel Carpenter, C. Moore, H. Schellman, I. Terekhov, J. Trumbo, S. Veseli, M. Vranicar, S. White, http://d0db.fnal.gov/sam

[2] Terekhov et. al., "SAM for Dzero – A Fully Distributed Data Access Layer for Dzero Run II", The VII International Workshop on Advanced Analysis Techniques in Physics Research (ACAT 2000), October 2000.

[3] Loebel Carpenter et. al., "SAM Overview and Operational Experience at the Dzero Experiment", "Resource Management in SAM and the D0 Particle Physics Data Grid", "SAM and the Particle Physics Data Grid", CHEP 2001, September 2001, Beijing China.

[4] Igor Terekhov, "Distributed Processing and Analysis of Physics Data in the Dzero SAM System at Fermilab", Fermilab-TM-2156.

[5] The Enstore home page, http://www-isd.fnal.gov/enstore

[6] I. Terekhov et. al., "Distributed Data Access and Resource Management in the Dzero SAM System", High Performance Distributed Computing Conference, August 6-10, 2001.

[7] The LSF Batch system, Platform Computing, http://www.platform.com/

[8] The Portable Batch System, http://http://www.openpbs.org

[9] FBS is the Fermilab Batch System, http://www-isd.fnal.gov/fbsng

[10] The Condor project home page, http://www.cs.wisc.edu/condor/.

[11] The Globus Project, http://www.globus.org

[12] Storage Resource Manager, http://sdm.lbl.gov/srm