The Data Access Layer for D0 Run II: Design and Features of SAM

Lee Lueking[1], *Heidi Schellman*[2], *Igor Terekhov*[1], *Julie Trumbo*[1], *Siniša Veseli*[1], *Matthew Vranicar*[1], *Richard Wellner*[1], *Stephen White*[1], *Victoria White*[1]

[1] Fermilab, Batavia Illinois, USA
[2] Northwestern University, Evanston Illinois, ,USA

Abstract

The Sequential Access Model, or SAM, is the data access layer being built for Run II data handling at Dzero. An overview of the Dzero data handling system and a description of the data access software being built are given. At the center of the data management system is an RDBMS which is used to track file-based data and trace processing steps and usage patterns. The procedures by which data is added to the system and distributed to users are described. SAM is a functioning system and much operational experience has been gained running it. Additional features are planned to be added before the Fermilab collider run begins in spring of 2001.

Keywords: data storage,data access,RDBMS,CORBA,distributed systems

1 Introduction

The Sequential Access Model, or SAM, is the data access layer that is being built for Run II data handling at D0. It requires the mass storage services of a separate Run II component, the Enstore mass storage system [2], also developed at Fermilab. SAM's goals are to provide a straightforward interface for users of both batch and interactive applications, to provide a robust and easily accessible database for locating and describing datasets, to archive descriptions of data analysis projects, to promote efficient use of computing and network resources, and to help optimize performance from the data storage and delivery system. Although it has been built with D0's requirements for Run II in mind, its design is not D0 specific.

Details of the hardware configuration are shown in Figure 1. Tape drives are served tapes in an ADIC AML/2 robot. These drives are attached to mover nodes which are attached to the network through Fast Ethernet connections to a central network switch. Gigabit Ethernet connections are made to the switch for the on-line DAQ, Central Analysis Server, Reconstruction Farm and Database Server machines. Several test facilities are also shown in the illustration. The software which is employed to operate the system includes a user interface and access layer, SAM, the Mass storage layer, Enstore, and various batch and lower levels which operate the individual components of the system.

2 Database

A commercial RDBMS is used to maintain a carefully designed schema which includes event, file, processing and required physics information. The database chosen was ORACLE because a robust and mature product was essential to the longterm stability of the system. Also, the size of the database is anticipated to be several hundred GB, and many features in Oracle are useful in managing a repository of this magnitude. In addition, Oracle offers many tools needed to design and implement the database as well as to tune and monitor its operation.

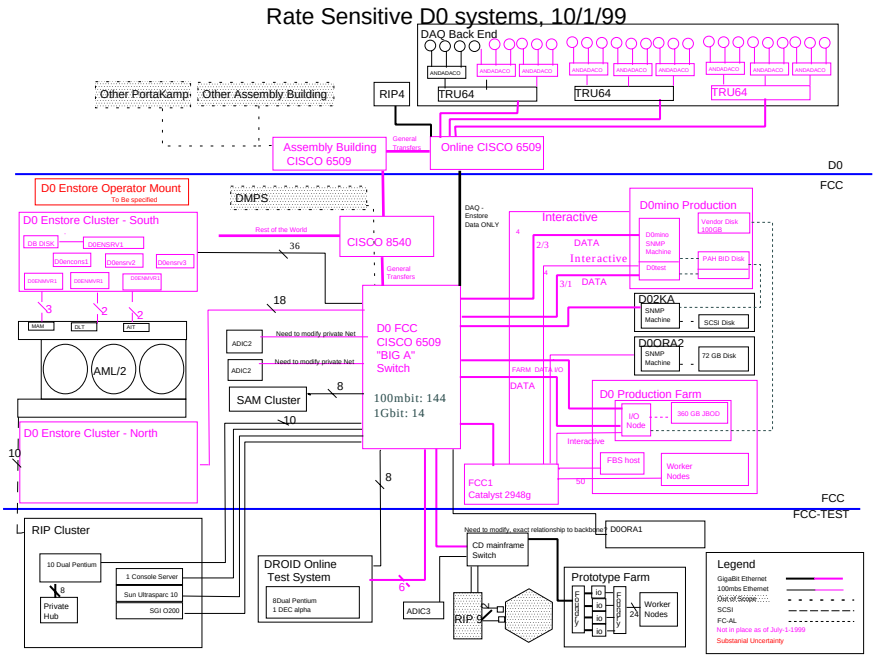

Figure 1: The data storage and delivery system is entirely network distributed with many hardware components and network connections. Shown are the AML/2 robot and mover nodes (left), the Cisco network switch (center), and the central analysis server, database servers, and reconstruction farm (right). At the top is the on-line system, and at the bottom are several test and development systems.

The schema has been designed to catalog event, file, various run and physics information. The simplified schema is illustrated in Figure 2. It allows tracking data through a wide variety of complex processing steps, and tracing lineage from detector or Monte Carlo files through analysis stages. Information crucial to many types of analysis, such as trigger, luminosity, and accelerator conditions are available through the system. Also, information about the location of files, on tape or disk cache, is used to optimize the performance of the overall system.

3 Implementation and Features

SAM is a fully distributed system using a client-server architecture. Communication between the many servers and their clients is done via CORBA. One of the principal servers manages all communication with the ORACLE database, another manages data delivery to multiple client data-file consumers running on various platforms, another is responsible for managing storage of data, and others actually interact with the underlying storage system, Enstore, to effect data movement between a file system and tape. Resource management servers cluster and re-order work for the storage system to minimize tape mounts, interact with the Batch system, and manage disk caches and disk cache policies. Additional servers collect information from the system at large, analyze and present information, and provide other important functions related to resource management. The servers are written in C++ and Python. Clients are written using Python, C++, and Java. Current implementations, running on Irix, Linux and OSF1, use ORBacus, a lightweight but robust C++ and Java CORBA implementation, available free with full source code for non-commercial use [5].

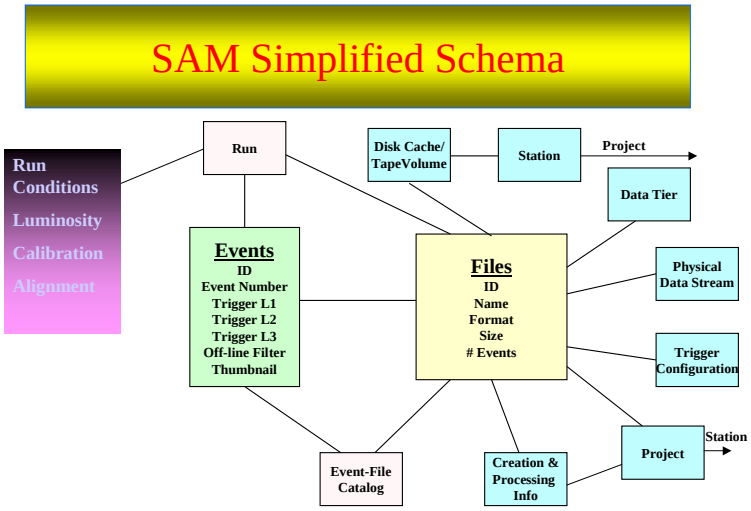

Figure 2: The SAM database schema provides information about the locations of files, processing stages, and group and user data usage profiles. This information is used to provide access to the data, and also optimize the use of resources.

Python clients and servers use the fnorb CORBA implementation [6].

All SAM user functions, such as defining the dataset to be analyzed, starting delivery, registering to receive files, storing data files, etc. are available from the command line. In addition read/write access to data files has been seamlessly integrated into the D0 analysis framework [3], and the D0 persistency mechanism d0om [4] through a standard framework package for SAM. Transparently to other framework packages, the SAM package handles open/close file events and communicates with the SAM system to receive/store files.

The SAM system has now been integrated with Enstore and with the D0 analysis system, and is being used for testing and for D0 Monte Carlo data management. Its feature list includes 1) data storage on tape and disk cache(s), 2) data description storage in an ORACLE database including a detailed event catalog for raw detector data, and other selected data files, and storage of parameter files 3) data delivery to multiple and coordinated users, 4) storage and re-use of pre-defined data queries 5) command line interface to all user functions 6) GUI interface to applicable functions 7) web-based database browsing tools, 8) disk cache and buffer management, and 9) administration and monitoring tools. Several additional features are planned over the next 6 months, including a facility for extracting and providing single events, additional recovery and restart features, and greater support for remote-site installations of the system.

4 Status and Experience

The system is documented and being used to store Monte Carlo data and by D0 physicists to perform analyses tasks. The home page for the SAM system can be found at [1], and for Enstore at [2], these include documentation and other information about the operation of the systems. To date we have stored over 1.5 TB on Mammoth I tapes in the robot, and have observed cumulative transfers in and out of the robot as high as 1.2 TB in one day. Data is added to the system by providing a *description* file which includes information about the parentage, size, and physics parameters for

files. These parameters are used to add catalog information to the database, and then the data files are transfered to the robot. A mechanism is also provided for adding data tapes directly to the robot and storing a meta-data file to the Enstore system which provides the system with information needed to transparently access the data later. Also tapes can be written and ejected from the robot and sent to remote sites for analysis. These features are illustrated in Figure 3. To exercise the system, Monte Carlo data from Lyon, Amsterdam, Prague, as well as Fermilab is being imported into the system, mostly over the network up to now, but soon data will be imported on tape also.

A test harness is rapidly evolving. This simulates use of the system by multiple clients storing and retrieving data and running across all of the systems involved - from the online logger systems, to the production processing farms, to the central analysis server and additional smaller Linux servers. Although the throughput of the storage system is currently limited by the number and type of tape drives available, Enstore provides facilities for sinking data and delivering random data at full 10MB/sec/tape rate, and these can be used for some of the scalability tests of the system.

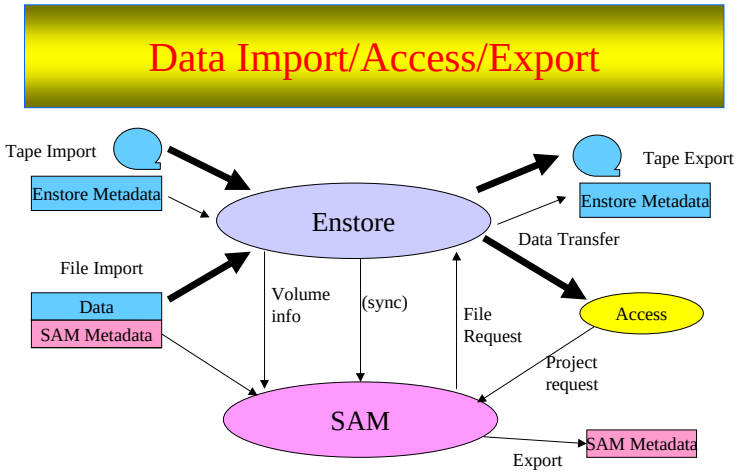

Figure 3: Data is added to the system, either from disk-files, or tape-files by supplying a description file to the SAM system. Data can be accessed through the SAM system, and can be exported to remote sites on tape if desired.

5 Conclusion

A large part of the data management and access system for the Dzero Run II has been built and is being tested. The system is able to store, catalog, and access data which is being imported form all over the world. The system includes many useful features now, and several additional ones are planned to be added in the next few months. So far the system is working well, and we continue to test and push its performance under heavier loads and larger data sizes. We hope to shortly demonstrate the full scalability of the system up to its design goal, namely movement of data in/out of the underlying storage system at 200MB/sec, including at least 150 MB/sec data transfers into the D0 central analysis system and 30 MB/sec in/out to the production farm system.

References

1 “The Sam Home Page”, http://d0db.fnal.gov/sam

2 “The Enstore Home Page”, http://www-isd.fnal.gov/enstore/

3 J. Kowalkowski,”The D0 Framework”, CHEP2000 presentation.

4 “The d0 object management system: D0om”, http://www-d0.fnal.gov/newd0/d0atwork/computing/infrastructure/d0om/d0om.html

5 “ORBacus for c++ and java”, http://www.ooc.com

6 “fnorb, a python ORB”, http://www.fnorb.org