Fermilab FERMILAB-Conf-00/140-E SAM June 2000

Distributed Data Access in the Sequential Access Model at the D0 Experiment at Fermilab

Igor Terekhov, Victoria White
(for the SAM [1] project)
Fermi National Accelerator Laboratory, Batavia, IL, USA
{terekhov,white}@fnal.gov

We present the Sequential Access Model (SAM)[1], which is the data handling system for D0, one of two primary High Energy Experiments 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. The design of SAM is not specific to the D0 experiment and carries few assumptions about the underlying mass storage level; its ideas are applicable to any sequential data access. By definition, in the sequential access mode a user application needs to process a stream of data, by 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. The adopted model allows for significant optimizations of system performance, decrease of user file latency and increase of overall throughput. In particular, caching is done with the knowledge of all the files needed "in the near future", defined as all the files of the already running or submitted jobs.

The bulk of the data is stored in files on tape in the mass storage system (MSS) called Enstore[2] and also developed at Fermilab. (The tape drives are served by an ADIC AML/2 Automated Tape Library). At any given time, SAM has a small fraction of the data cached on disk for processing. In the present paper, we discuss how data is delivered onto disk and how it is accessed by user applications. For a recent broad review of SAM, see [3]. We will concentrate on data retrieval ("consumption") from the MSS; when SAM is used for storing of data, the mechanisms are rather symmetrical.

All of the data managed by SAM is cataloged in great detail in a relational database (ORACLE). The database also serves as the persistency mechanism for the SAM servers described in this paper. Any client or server in the SAM system which needs to store or retrieve information from the database does so through the interfaces of a CORBA-based database server. The users (physicists) use the database to define, based on physics selection criteria, datasets of their interest. Once the query is defined and resolved into a set of files, actual data processing, called a project, may begin. Obviously, running projects involves data transfer and resource management.

The computing facilities with their CPU, disk, and other hardware resources are logically partitioned into collections of resources called *stations*. A station may be a single node, a fraction thereof (some of the machine's disks and/or CPUs may constitute a station) or a collection of smaller nodes. It is equipped with a server, called station master (SM), that coordinates data delivery and projects using the data. User requests to actually run a project proceed through the SM, which determines the amount of cache replacement, if any, needed to run the project. If viable, the user job is submitted into a station-associated batch queue, otherwise the project is rejected and the user may try another station.

Thus, the total "disk" need not be concentrated on a single machine; rather, it is arbitrarily apportioned to stations. Each station is dynamically configurable and controls its own disks but may access other stations' disks thus rendering the whole of the disk space a fully distributed cache. Within each station, the local resources are further divided (in the "fair share" fashion rather than partitioned) among different research groups. Within each group, the access patterns and the datasets of interests highly correlate, thus allowing each group to vary dynamically its cache replacement policy for optimal performance. Thus, the SM's cache manager is really a coordinator of groups' sub-cache managers.

Data delivery is performed asynchronously with respect to its consumption. In what follows, we describe how each project's consumption is coordinated by the SAM server called the project master (PM). There are two access modes of fundamental importance, both of which have been successfully used at Fermilab.

In the first mode, called the "Freight Train", a research group needs to scan a large (e.g., multi-TB) dataset, so that its disk cache effectively acts as a buffer. Each participant registers his or her application as an independent consumer each of which is intended to "see" every file in the dataset. Naturally, different consumers have different processing speeds, and the PM enforces the pace by limiting the maximum time a consumer works on a file. (A consumer which timed out for any reason will have to miss some of the files in the dataset; thus, the mode is best suited for already debugged, production type applications.)

A consumer application may have multiple threads of data processing. SAM refers to such conceptual threads as *consumer processes* sharing the consumer identity (CID). These processes may be implemented as threads or separate operating system processes. During the project, any process may be given any of the files from the dataset but different consumer processes "see" different files. New consumer processes may be added dynamically; in fact, a "slow" consumer may "catch up" with the Freight Train by registering a new process, contingent on CPU and other resource availability.

In the second fundamental mode, called the "Farm" [4], different consumer processes may run on physically different machines thus rendering the "consumer" another distributed concept. (The *Farm* is the Fermilab term for a uniform cluster, typically Linux). Files are routed to different farm nodes as buffer space therein becomes available, The aforementioned capability to dynamically add (and remove) consumer processes is particularly useful for this access mode. The cluster nodes are units of resources for the Farm Batch System[5] which may dynamically reallocate the nodes to different jobs (SAM projects). As with distributed computing in general, Farm mode data access is fault-tolerant: if an entire node becomes unavailable for any reason, the project master will eventually ensure that the unconsumed files that have been (or are being) delivered to the node will be re-routed to consumer processes on other nodes.

The Freight Train and the Farm constitute the two orthogonal axes for data delivery. Any hybrid mode is possible whereby multiple machines comprise a station that runs multiple project masters which coordinate file delivery and consumption to multiple consumers which in turn have multiple process running at different nodes. This capability to use resources under distributed ownership justifies SAM as a fully distributed data handling system.

The Mass Storage System is a single global resource in the system. (the cardinality is driven by economical reasons). The results of our practical analysis, not shown here, are such that the aggregate ATL bandwidth could not satisfy an uncoordinated stream of data requests from the hundreds of users. Thus, in addition to an efficient disk caching, SAM must provide coordination of requests to the MSS from the different stations. Note that the MSS itself performs optimizations of already submitted requests. SAM however, performs resource management and optimization of requests *before* they are even forwarded to the MSS. A single server, called Optimizer, or Global Resource Manager, groups (typically by volume) and regulates requests from different cache managers. Each Station Master must acquire Optimizer's authorization for file requests, and the authorized request group is treated as the unit of MSS transfer. (Of course, files from a group may be routed to different disks/nodes).

All of the distributed components are implemented with CORBA[6] interfaces in C++ and Python with some GUI written in Java. We use ORBacus and Fnorb [7] products.

We have described the SAM system that provides access to distributed data at Fermilab. As of Spring 2000, physicists at our laboratory have successfully used SAM for several months for storage and subsequent retrieval and processing of several TBytes of data.

References

[1] The SAM project L.Lueking[1], H.Schellman[2], I.Terekhov, J.Trumbo, S.Veseli, M.Vranicar, R.Wellner, S. White, V.White[1]. [1] Project Leader [2] Northwesterm University, Evanston, IL, USA `http://d0db.fnal.gov/sam`.

[2] The Enstore project home page `http://www-isd.fnal.gov/enstore`.

[3] V.White for the SAM project, "The Data Access Layer for D0 Run II: Design and Features of SAM", talk given at *The International Conference on Computing in High Energy and Nuclear Physics* (CHEP 2000), February, 2000, Padova, Italy,

[4] H.Schellman *et al.* , "Report on the D0 Linux Production Farms", CHEP2000, see [3].

[5] The FBS project `http://www-isd.fnal.gov/fbs`.

[6] The OMG home page `http://www.omg.com`.

[7] "ORBacus for C++ and Java" `http://www.ooc.com`. "Fnorb, a Python ORB" `http://www.fnorb.org`.