LIGHTWEIGHT DEPLOYMENT OF THE SAM GRID DATA HANDLING SYSTEM TO NEW EXPERIMENTS. (CHEP 2006 - 244)

Arthur Kreymer, Andrew Baranovski, Elizabeth Buckley-Geer, Robert Hatcher, Lauri Loebel-Carpenter, Adam Lyon, Sinisa Veseli, Stephen White, Fermilab

Valeria Bartsch, University College London

Abstract

The SAM [1] data handling system has been deployed successfully by the Fermilab D0 [2] and CDF [3] experiments, managing Petabytes of data and millions of files in a Grid working environment. D0 and CDF have large computing support staffs, have always managed their data using file catalogue systems, and have participated strongly in the development of the SAM product. But we think that SAM's long term viability requires a much wider deployment to variety of future customers, with minimal support and training cost and without customization of the SAM software. The recent production deployment of SAM to the Minos experiment [4] has been a good first step in this direction. Minos is a smaller experiment, with 30 terabytes of data in about 600,000 files, and no history of using a file catalogue. We will discuss the Minos deployment and its short time scale, how SAM has provided useful new capabilities to Minos, and where we have room for improvement. The acceptance of SAM by Minos has depended critically on several new capabilities of SAM, including the C++ API, the frozen client software, and SAM Web Services. We discuss lessons learned, speculate on future deployments, and invite feedback.

MINOS DEPLOYMENT

MINOS Data

As of 30 January 2006, the MINOS data[5] consisted of about 32 Terabytes contained in 600,000 files. Files are stored in the Fermilab Enstore system. Users read the files directly via DCache, or indirectly by making local copies via dccp or ftp. At Fermilab, about 3 TB of AFS space is available for these local copies. Data files are organized in a traditional directory structure reflecting the origin and processing of the files. Raw data files go into monthly directories with names like neardet_data/2006-01/ . For each raw data file, Minos produces reconstructed data both in 'cand' candidate files with full event information, and in several streams of ROOT ntuple files. A typical reconstructed output path looks like
reco_near/R1_18_2/cand_data/2006- 01

At this time there are about 2.3 Terabytes of raw data, growing by roughly 1 TB per year.
Each reconstruction pass produces 4 to 8 TB of data, depending on the physics goals of the pass.

Traditionally, users have located files by doing an 'ls' command either locally on the PNFS exported file system, or remotely via ftp.

MINOS file names follow a fairly simple format. They are globally unique, which is fortunate, as this is a firm requirement in the SAM system.
Sample raw and reconstructed file names are
N00009668_0009.mdaq.root and
N00009668_0009.spill.cand.R1_18_2.0.root
The reconstructed file fields, for example, specify

- N - detector
- 00009668 - Run
- 0009 - Subrun
- spill - trigger type
- cand - output stream
- R1_18_2 - code release
- 0 - iteration
- root - format

Monte Carlo file names are much more complex, with at least eleven fields.

Motivation for MINOS SAM deployment

Lacking a formal data catalogue, Minos was effectively using the file system itself as a catalogue . This has obvious scaling problems when nearly a million files are involved. And this did not allow for possible future deployment to grid resources.

There were also specific local operational problems with the old scheme. A single user's attempt to list files in a single directory (typically 1000 files) could saturate the capacity of the central FTP server for 10 to 20 minutes, causing failures for all users of that system.

Given the clear success of SAM in the CDF and D0 experiments, and the availability of strong local support, using SAM was the natural choice.

Table 1: Comparison of Experiments

	D0	**CDF**	**MINOS**
Files (10^6)	3.8	1.9	.65
Terabytes	1420	1430	32
FTE dev	5	2	0
FTE ops	1	1	0.2

Goals of Minos SAM deployment

Our intention has been to make the most useful SAM functions available without in any way compromising the existing usage patterns. The most useful functions were thought to include :

- Getting lists of files based on metadata selections, both from programs and scripts, and via the web, at remote sites.
- Creating named dataset definitions based on metadata. This improves communication between people, and allows analysis results to be reproduced easily and reliably.
- Running 'projects' in the Minos software framework using these file lists and dataset definitions.

We need to provide these functions with a very small ongoing support cost, no more than a small fraction of an FTE. These functions are needed both at Fermilab and at remote sites, with minimal installation cost.

SAM Infrastructure and requirements

The are several servers used by SAM

- A single central Oracle database server.
- CORBA interfaces between the client and Oracle (called 'dbservers' by SAM.)
- SAM 'stations', servers used by clients to manage file delivery.
- Sam at at Glance - web monitoring [6]
- Database Browser - interactive web access to metadata [7]
- Code browser [8]

The database browser and code browser are shared by all the experiments.

The Oracle server is installed and managed by the Fermilab Computing Division Database Support Group, who support the CDF and D0 SAM Oracle servers. Oracle runs on a Sun Fire V20z dual processor 2.2 GHz Opteron system with 8 GBytes of memory, running RedHat Enterprise Linux 3. The present SAM tables use about 12 GBytes of disk. We are now making the transition to Oracle 10g, which is running fine for Minos on the development server.

The SAM dbservers, stations, and client software are deployed directly by Minos, on unremarkable 3 GHz dual Xeon systems. The dbserver processes use no more than about 1/2 GByte of virtual memory, and usually much less. The station's use of resources is negligible.

While it would be possible, for evaluation, to run Oracle, dbservers, stations and client code on a single system, this would not be advisable in the long term. Certified support of Oracle requires very specific operating systems and patch levels which may not be compatible with our other applications. SAM stations and dbservers are also best run on separate systems, due to interactions between the software components.

Timeline of Minos SAM deployment

(Calendar year 2005)

Feb - single file tests
Mar - generated raw data metadata
Apr - declared metadata to SAM
May
automatic keepup of raw data
installed production dbserver
moved to MINOS Oracle server
SAM in Minos framework
Jun - test reco declares
Jul
declare reco to SAM
started user beta tests
Aug
defined standard datasets
SAM in production
Oct
testing web services
SAM for production monitoring
Nov -
web services used in production
testing Oracle 10g

Strategy

SAM offers many powerful advanced features, including event level metadata, file transfer and storage utilities, multiple remote file caches on remote stations, and luminosity tracking. Minos has chosen to initially deploy none of these advanced features, testing only the small subset of basic features immediately needed by Minos, as noted in the Goals subsection above. This provides us with immediate benefits, with minimal support and migration costs.

Tactics

The timing of our deployment has been most fortunate, as several critical tools became available just when needed.

The SAM project had just completed a major rewrite of the dbserver code, from V5 to V7, just in time for us the use the much superior and stable V7 code, and avoid entirely the transition that CDF and D0 had to make.

The 'frozen' SAM client software became available just in time for us to use it, making client installation practically trivial. Previously, we had to install a half dozen products via UPS/UPD, then run special tailoring commands. On execution, SAM commands had to load and interpret a large number of Python scripts. Prior to deployment of the frozen sam client, CDF had been having severe overloading problems with NFS servers for its large analysis farms. Thanks to efforts of Alan Sill and other SAM developers, the sam command is now a self contained binary, accompanied by a few shared libraries and other files. Execution is also much faster.

A C++ SAM client interface became available just in time for Minos to use it. This was written to support the CDF SAM migration, and we integrated it into the Minos framework with relatively little trouble.

Some of the important remote Minos client systems are not running any of the operating systems supported by the SAM team. The SAM Web Services prototypes are already being used successfully by these clients.

Customizations

It may be useful to summarize the customizations made during the Minos SAM deployment.

Most elements, including the stations, dbservers, and even the database schema are identical to those used in CDF and D0.

There is an internal Minos static web page documenting the use of SAM in Minos. [9]

There was a slight adjustment made to the generic Database browser web interface, to provide default fields and values most likely to Minos users.

A custom web page [10] provides lists of files in formatted for later use with DCache (dcap), PNFS (encp) or FTP. The metadata are specified via menus. This is a custom CGI script, but it runs on the standard Fermilab central web servers.

Lessons learned

The weekly 1/2 hour "Sam Design" [11] meeting, held at Fermilab and via an H.323 video and telephone conference, has been extremely valuable, allowing an efficient and free exchange of information regarding data handling operations and planning.

Minos has not found it necessary to have people assigned to formal data handling shifts as in D0, or to have daily data handling operations meetings as in CDF.

It was a real challenge to generate and enter SAM metadata for 450,000 Minos data files. Doing anything that many times is bound to be difficult.

It is clear that it is not efficient to manage data in files a lot smaller than about 1 GByte. Various components of the Data Handling system, including DCache, Enstore, and SAM, have per- file overheads of order 1/10 second. The biggest offenders in Minos are the small ROOT ntuple files which result from producing one output file for each original raw data file The raw data is produced as a set of about 20 'subrun' files for each physics run of about one day. We will start merging these subruns into a single file as part of the reconstruction process, cutting the file count by about a factor of 20.

The original 20 to 50 MByte raw data files are also much too small for efficient handling. But they are primarily read once by the farms, and have relatively little other usage, so we will concentrate on the lower hanging fruit fruit for now.

Minos SAM performance

The single project file rate is limited to about 1/second, 10% CPU on dbserver. The global rate saturates at about 5/second, 100% CPU on dbserver, with 6 active projects. Because is takes several minutes to read a typical large data file, and Minos only has a few hundred nodes of batch capacity available to it at Fermilab, this should be no problem in normal use.

The Minos SAM deployment has been remarkably stable. The station has been running without interruption for nearly six months, and the dbserver for about 3 months. This is in spite of upgrades to the Oracle database and host operating system. The number of users and usage levels are relatively small, but they include critical applications like the primary physics quality monitoring of reconstructed data.

FUTURE DEPLOYMENTS

One obvious impediment to a broader deployment of SAM is the requirement of a rather expensive central Oracle database. The license cost of this may be changing, with the upcoming release of Oracle Express Edition. This version will only use 1 CPU, 1 GByte of memory and 4 GB of disk; this should be enough for many smaller users. We also need to reduce the personnel cost; we presently rely on the kind assistance of Database Administrators with extensive prior experience supporting SAM for CDF and D0.

Internal SAM development tests include use of the mini-sam environment, using a Postgresql back end. We may want to consider production support of this Postgresql back end.

The scripts used to install SAM station software provide a choice of standard configuration files for CDF, D0 or Minos. We need to make these scripts and configurations more portable.

The present SAM dbserver system requires an active connection to the backend database, with little caching of local state information. We are evaluating tools to remove this requirement, see paper 123 "SAMGrid Web Services" and 125 "SAMGrid Peer-to-Peer Information Service" by Sinisa Veseli at this CHEP conference.

The present SAM station contains hard coded support for the various local file cache managers (local files, DCache). We intend extend and generalize this by moving to a standard SRM interface.

Although Minos is a smaller customer, we have not forgotten the need to improve scalability.
In particular, we are prototyping a dbserver multiplexer layer, which allows the transparent use of additional dbserver systems, improving reliability and performance.

REFERENCES

.... REFERENCES NEED TO BE ADDED ....
[1] SAM - http://projects.fnal.gov/samgrid/
[2] D0 - http://www- d0.fnal.gov/
[3] CDF - http://www- cdf.fnal.gov/
[4] Minos - http://www- numi.fnal.gov/
[5] Minos data - http://www-numi.fnal.gov/minwork/computing/dh/dhmain.html
[6] Sam At a Glance - http://www-numi.fnal.gov/sam_local/SamAtAGlance/
[7] database browser - http://dbb.fnal.gov:8520/minos/databases
[8]code - http://cdfkits.fnal.gov/SamCode/
[9] intro - http://www-numi.fnal.gov/sam/
[10] Minos file lister - http://www-numi.fnal.gov/computing/findrun_sam.html
[11] sam-design - http://listserv.fnal.gov/archives/sam-design.html