THE SAMGRID MONITORING SERVICE AND ITS INTEGRATION WITH MONALISA

A. Lyon, P. Vokac, M. Zimmler, A. Baranovski, G. Garzoglio, L. Loebel-Carpenter, R. Herber, R. Illingworth, R. Kennedy, A. Kreymer, A. Kumar, L. Lueking, W. Merritt, I. Terekhov, J. Trumbo, S. White, S. Veseli, FNAL, Batavia, IL 60510, USA, M. Burgon-Lyon, R. StDenis, Glasgow University, S. Belforte, INFN, Trieste, U. Kerzel, Karlsruhe University, M. Leslie, V. Bartsch, S. Stonjek, Oxford University, F. Ratnikov, Rutgers University, A.Sill, Texas Tech University

Abstract

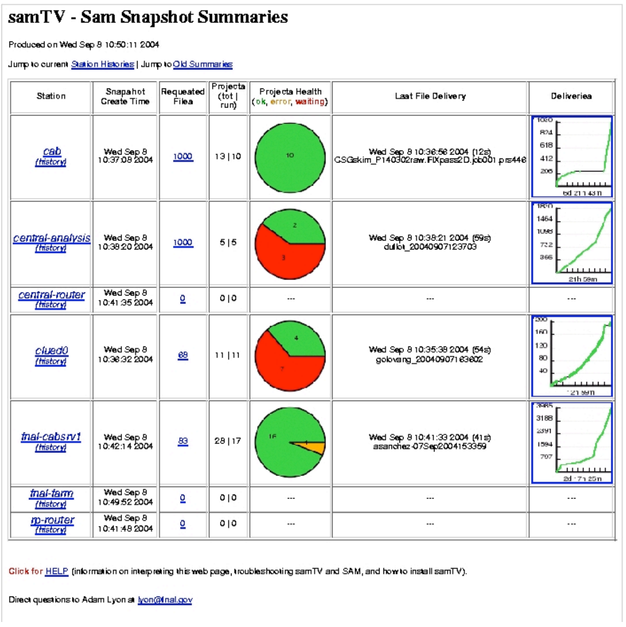
The SAMGrid team is in the process of implementing a monitoring and information service, which fulfils several important roles in the operation of the SAMGrid system, and will replace the first generation of monitoring tools in the current deployments. The first generation tools are in general based on text log-files and represent solutions which are not scalable or maintainable. The roles of the monitoring and information service are: 1) providing diagnostics for troubleshooting the operation of SAMGrid services; 2) providing support for monitoring at the level of user jobs; 3) providing runtime support for local configuration and other information which currently must be stored centrally (thus moving the system toward greater autonomy for the SAMGrid station services, which include cache management and job management services); 4) providing intelligent collection of statistics in order to enable performance monitoring and tuning. The architecture of this service is quite flexible, permitting input from any instrumented SAMGrid application or service. It will allow multiple backend storage for archiving of (possibly) filtered monitoring events, as well as real time information displays and active notification service for alarm conditions. This service will be able to export, in a configurable manner, information to higher level Grid monitoring services, such as MonALISA. We describe our experience to date with using a prototype version together with MonALISA.

INTRODUCTION

Overview of the Existing Monitoring and Information Services

SAMGrid[*] [1] is a general data handling system designed to work for experiments with peta-byte sized datasets and widely distributed production and analysis facilities. The system has relied on the centralized Oracle RDBMS since its beginnings, not only for file metadata, event and replica catalogues, but also for process bookkeeping and keeping track of *station*[†] configuration and state. Thus, to a large extent the system information services, which are vital for its operation, have been encapsulated within the SAMGrid CORBA-based DB Servers.

On the other hand, for the system monitoring we have used a wide variety of tools, such as the CORBA Naming Service polling scripts and SAM-at-a-glance web pages, which periodically inquire the data handling servers about their status. The most recently written tool, called SAMTV [2], provides graphical information and statistics about the system performance, status of the data handling servers and of the user projects. This tool (see Figure 1) has proven to be invaluable for monitoring and troubleshooting the system and diagnosing problems.



samTV - Sam Snapshot Summaries

Produced on Wed Sep 8 10:50:11 2004

Jump to current Station Histories | Jump to Old Summaries

Station	Snapshot Create Time	Requested Files	Projects (tot | run)	Projects Health (ok, error, waiting)	Last File Delivery	Deliveries
cab (history)	Wed Sep 8 10:37:08 2004	1000	13 | 10	10	Wed Sep 8 10:36:56 2004 (12s) CSGskim_P140302raw.FIXpass2D.job001.prs446	6d 21h 43m
central-analysis (history)	Wed Sep 8 10:38:20 2004	1000	5 | 5	2, 3	Wed Sep 8 10:38:21 2004 (59s) duflot_20040907123703	21h 59m
central-router (history)	Wed Sep 8 10:41:35 2004	0	0 | 0	---	---	---
clued0 (history)	Wed Sep 8 10:36:32 2004	68	11 | 11	4, 7	Wed Sep 8 10:35:38 2004 (54s) golovang_20040907163602	21h 59m
fnal-cabsrv1 (history)	Wed Sep 8 10:42:14 2004	83	28 | 17	16, 1	Wed Sep 8 10:41:33 2004 (41s) asanchez-07Sep2004153359	2d 7h 25m
fnal-farm (history)	Wed Sep 8 10:49:52 2004	0	0 | 0	---	---	---
rip-router (history)	Wed Sep 8 10:41:48 2004	0	0 | 0	---	---	---

Click for HELP (information on interpreting this web page, troubleshooting samTV and SAM, and how to install samTV).

Direct questions to Adam Lyon at lyon@fnal.gov

Figure 1: Snapshot of the main SAMTV [2] web page for the D0 SAMGrid stations running at Fermilab. The

[*] The SAMGrid system offers a wide variety of services, including those for data transfer, storage and management, as well as for process bookkeeping on distributed systems. The system is used by D0 and CDF, and is being tested for use by MINOS and CMS.

[†] *Station* denotes a particular set of hardware resources that are managed by SAMGrid servers. End users request a set of files by submitting a SAMGrid *project* to one of the SAMGrid stations. Their applications are served input files by the *project manager* (one of the station servers).

summary page contains links to information for individual stations and user projects running on those.

Recent enhancement of the SAMGrid system with the Job and Information Management (JIM) components [3] has also introduced new functionality for the system monitoring and information services. Those components use XML Xindice database [4] for configuration information, as well as the Globus MDS software [5] for job monitoring [6] (see Figure 2).

SAM GRID INFORMATION & MONITORING SYSTEM
Launching the Monitoring System:
Participating Experiments:
D0
CDF

Figure 2: Snapshot of the JIM monitoring service [6]. The main web page contains links to both D0 and CDF JIM monitoring sites.

Problems with the Current Infrastructure

The existing infrastructure has lots of functionality built in and serves the SAMGrid system well. However, it is not without problems: centralized database and monitoring based on the log-file parsing.

Even though the performance of the centralized DB/DB Server has been very good so far, and will further improve with introduction of the new DB Server software [7], those centralized components remain as a single point of failure in the system. Thus, the whole data handling system is down whenever the machine hosting the central database, or the database itself, requires maintenance.

The SAMTV scripts, as well as many of our other tools, are based on parsing of the system text log-files. In addition to being non-scalable, this is also a maintenance nightmare. Parsing of the non-formatted log-files is difficult, it often breaks with new software releases, and it cannot be used for real-time monitoring of the data-handling servers and user projects.

In order to address the issues discussed above, we have started working towards improving the existing infrastructure by introducing the new event-based monitoring service, and distributing the centralized information services.

EVENT-BASED MONITORING SERVICE

General Architecture

The design of the new SAMGrid monitoring service is based on the Grid Monitoring Architecture [8], and aims to satisfy the following set of requirements:

- Scalability
- Reliability and performance
- Non-intrusiveness
- Interoperability with other monitoring systems
- Flexibility and extendibility
- Security

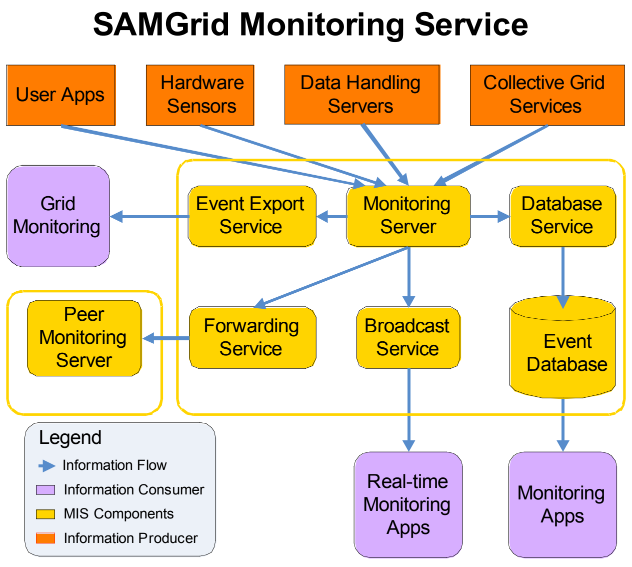


Figure 3: SAMGrid Monitoring Service Architecture.

The general architecture of the system is shown on Figure 3. The service is based on *events*, which are produced by the data handling servers and user applications, and sent via CORBA calls[‡] to the monitoring server. Monitoring server, which can operate in both *push* and *pull* event modes,[§] processes events and stores them into its local database backend for later mining with tools like SAMTV. It can also forward certain types of events to the real-time monitoring applications and/or to its peer servers, as well as export events into other monitoring systems.

[‡] The SAMGrid system uses CORBA as its communication protocol.

[§] In the push event mode, producers push event information into the monitoring server, and in the pull event mode the monitoring server requests producers to generate events. In the SAMGrid system the push mode is used for irregularly occurring events, such as a file delivery to a user application. On the other hand, the push mode is used for those events where regularity is desired. This is the case, for example, with events describing the station status.

Each monitoring server with its accompanying components will be able to provide service to one or more SAMGrid stations, which ensures that the system scales well. Events are based on dictionaries (keyword-value pairs), and can be stored into the local database as dictionaries. This makes the system extremely flexible: the existing events can be modified, and new events added to event producers without the need for modifications of any of the monitoring service components. Monitoring applications can obtain from the database the event information in the dictionary form, and decide how to process it.

Monitoring Server Design and Features

The monitoring server is designed with performance, robustness, and reliability in mind. It is written in Python, multithreaded, and, as mentioned earlier, can both receive and pull events at the same time.

The event flow inside the server is shown on Figure 4. When a message arrives into the server, it is inserted into the incoming event queue. In case of clients pushing information into the server, the client call returns immediately after message is received. In this way the monitoring system is as non-intrusive as possible, and should not affect the performance of the data handling servers and user jobs that are being monitored. All incoming messages are unpacked by the queue manager and passed on for processing by a number of designated message processors, such as the database processor, broadcast service processor, event exporting processor, etc. The list of message processors which server will be using is configurable, so that, for example, one can choose whether or not to export any of the monitoring data into external (grid) monitoring systems. Each of the configured message processors is running in its own thread and has their own event queues. In this way, in case one of the processors starts having problems and its performance starts declining, the other processors will not be affected.[**]

In addition to the designated message processors which deal with all events and are provided with the monitoring server, certain event types can also be processed by the specialized event handlers. The event handlers are small python modules imported by the monitoring server when needed. These modules can be easily written or customized for the local SAMGrid installation. For example, one might want to send mail to the local station administrator in case one of the station disks starts exhibiting problems.

[**] For example, the speed at which events are inserted into the database might be affected by performance of the database itself.

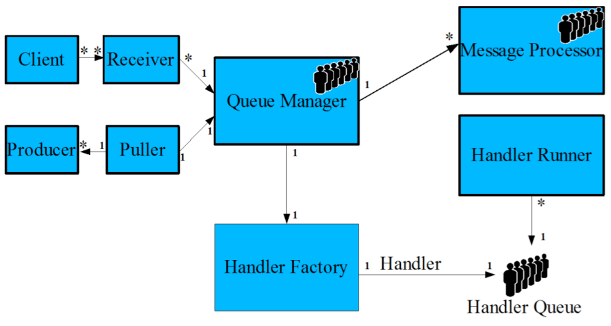


Figure 4: Message flow inside the SAMGrid Monitoring Server.

Note that the monitoring server can be configured to act as a simple event-forwarding service. This feature may be used to spread the monitoring load over several machines. The server performance can also be tuned by increasing the number of threads that perform certain tasks (e.g., one can increase the number of message processors which insert events into the database).

Client Support Infrastructure

Our client support libraries, which exist for Python and C++, aim to simplify the process of producing and sending events to the monitoring service. For example, in the case of a Python application one can add and send a new event with only a couple of lines of code. In addition to simplifying instrumentation of the SAMGrid (and user) code with new monitoring events, those libraries also contribute to the non-intrusiveness of the whole monitoring system by detecting any possible problems in communicating events to the monitoring service.

Note that the standard SAMGrid C++ API already has entire event infrastructure built in, and similar work is planned for our Python API in the near future. This will allow direct monitoring of all user applications built on top of those APIs.

Interoperability with Other Monitoring Systems

The SAMGrid Monitoring Service architecture allows interoperability with external (grid) monitoring systems via event processors that have the ability to export event data into their corresponding external monitoring system. Our first implementation of such event processor is intended to work with MonALISA [9]. This processor filters incoming SAMGrid events and extracts information that can be stored into MonALISA, and subsequently uses the MonALISA Python clients to actually store the data. At present there are certain limitations in MonALISA, which prohibit us from fully exploiting its functionality for the SAMGrid monitoring (e.g., MonALISA GUI can at the moment only display numerical values). However, we are currently pursuing collaboration with MonALISA developers with the goal of resolving the outstanding issues in the near future.

Initial Performance Numbers

Our initial investigations into the performance of the new system is based on the monitoring server and its MySQL [10] database backend[††] running on a dual 800MHZ Pentium III machine with 1GB of RAM. In this configuration the server was able to handle events at a constant rate of 45 events per second, plus additional 2-second spike of 70 events per second (see Figure 5). These numbers give us confidence that the new monitoring service will be able to easily handle the load produced by the SAMGrid system. This conclusion is further strengthened by the fact that the monitoring server is very configurable and that its performance can be tuned. However, more extensive performance evaluation and testing is still needed and will be done before the system is put into production later this year.

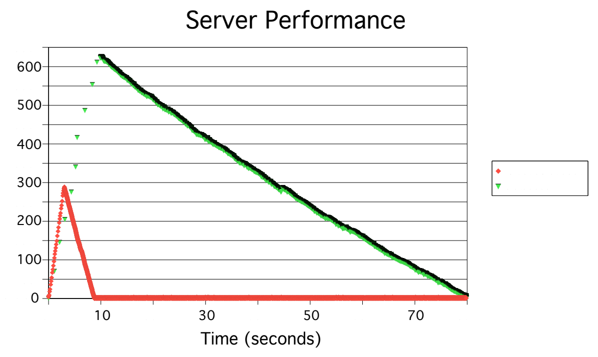


Figure 5: Monitoring Server performance under constant load of 45 events per second, plus one 2-second load spike of 70 events per second. Server needed about 80 seconds to clear the queue of messages waiting to be inserted into the database.

PLANS FOR DISTRIBUTED INFORMATION SERVICE

Our work on distributing the SAMGrid information services is still in its early planning stages. The overall system architecture will re-use many components of the new Monitoring Service (e.g., database backends), and will integrate many of the existing services provided by the SAMGrid DB Server and JIM software.

The main goal of this work is to move the system toward greater autonomy for the SAMGrid station services by providing runtime support for those services locally, as opposed to depending on the central database. This will in turn result in a better and more robust performance of the entire data handling system.

[††] At the moment we support only the MySQL database backend. Support for the other databases will be added in the future.

CONCLUSIONS

In this paper we have described the architecture, design and implementation of the new SAMGrid Monitoring Service, which will be deployed into production later this year. The system is flexible and robust, can interoperate with other monitoring systems, and our initial investigations indicate its very good performance.

In addition to discussing the Monitoring Service, we have also outlined plans for distributing the SAMGrid information services.

ACKNOWLEDGEMENTS

We would like to thank Fermilab Computing Division for its ongoing support of the SAMGrid project, and especially the CCF, CEPA, and Run II Departments. We would also like to thank everyone at D0 and CDF who has contributed to this project. This project is sponsored by DOE contract No. DE-AC02-76CH03000.

REFERENCES

[1] http://projects.fnal.gov/samgrid
[2] http://www-clued0.fnal.gov/~sam/samTV/current
[3] http://www-d0.fnal.gov/computing/grid
[4] http://xml.apache.org/xindice
[5] http://www.globus.org
[6] http://samgrid.fnal.gov:8080
[7] L. Loebel-Carpenter et al., "The SAMGrid Database Server Component: Its Upgraded Infrastructure and the Future Development Plan", this conference proceedings
[8] R. Aydt et al., "A Grid Monitoring Architecture", Global Grid Forum/GMA Working Group Document GWD-GP-16-1 (http://www.didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-1.pdf)
[9] http://monalisa.caltech.edu
[10] http://www.mysql.com