THE SAMGrid MONITORING SERVICE AND ITS INTEGRATION WITH MonALISA

A. Lyon, S. Veseli, P. Vokac, M. Zimmler (FNAL), M. Leslie (Oxford University)

SAMGrid

SAMGrid is a large scale distributed system to deliver petabyte scale datasets for processing at the CDF and DØ experiments.

It does this by providing the following services in a single unified framework:

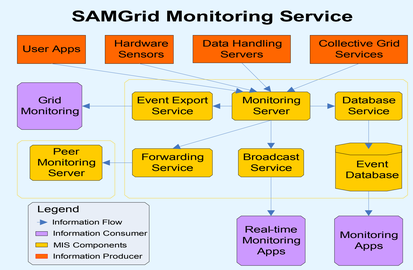
- Managing File Storage
  - Files are housed on tape and cached on disks around the world
- Managing File Delivery
  - Get files from tape or cache
  - Provide location transparency
  - Manage your local cache
  - Use a variety of file transfer mechanisms
- Managing File Metadata
  - The SAM database allows metadata based file retrieval
  - User does not need to know a filename
- Providing Analysis Bookkeeping
  - What files you ran over, with which application.
- Managing Jobs
  - Choose an execution site, deliver job and data to it and store output

Monitoring SAMGrid

SAMGrid consists of many services running many machines. Monitoring is essential for the day to day operation of such a system, and plays an important role in testing new SAMGrid components.
We have designed a system that can accept monitoring data from many sources, store it for applications that monitor historical data, forward it to real time event monitoring applications, and export it to standard grid monitoring services.

Monitoring information can be pushed from clients to the servers, or a server can pull information from a client.

Customizable 'Event Processors' then pass on, store, or translate the events as required.



Monitoring Service Features

- Fast Multithreaded Design
  - A slow event processor will not slow down other server functions
- Modular and Scalable Architecture
  - Load balancing and forwarding allows load to be spread across several machines
- Small unobtrusive client side API
  - Fast Python and C++ monitoring APIs will not impact performance of service being monitored.
- Flexible Dictionary based Event Format
  - Events contain arbitrary data in dictionary format (key/value pairs)
- Exports monitoring data to standard grid monitoring tools
  - Monitoring information is passed on to MonALISA
- Database Event Logging
  - Database stores all events for a configurable period of time
- CORBA based communication
  - CORBA is the remote object access method used by the rest of the SAM system, allowing monitoring to be added with almost no overhead

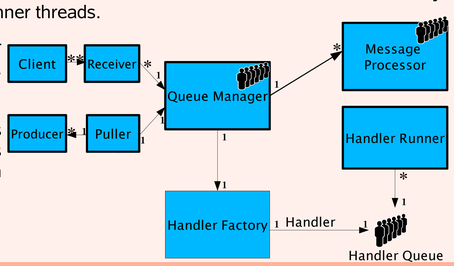
Server Architecture

The server design aims to be efficient and easily extensible. When monitoring events are received, they are simply inserted into a queue. This means the client can return almost immediately. The queue manager thread then decodes each event and passes it on to be processed. Processing is done using one of two mechanisms - Message Handlers and Message Processors.

Message Processors run in their own thread, and have their own queue so that they can work independently. Handlers are more lightweight, and are created as messages are received. Handlers do not run in their own thread, but are run by general purpose handler runner threads.

Processors are thus better suited for common tasks, such as database logging.

Handlers are used for less common tasks, such as executing server shutdown commands.



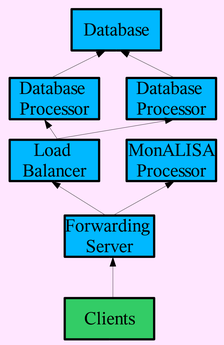
Deployment

The MIS server runs event processors selected in its configuration file.

This, combined with the availability of load balancing and forwarding processors allows deployment across more than one machine. In the diagram to the right, each of the blue message processor boxes may be deployed on a separate node.

The diagram to the right shows how a large scale system might use several machines to cope with the high load.

At present, we expect a single machine to be sufficient for monitoring SAMGrid. The ability to run a message forwarding server is however still useful in situations where farm nodes do not have Internet access. These farm clients could use the farm's head node as a server, which could forward information to the main monitoring service



Performance

Performance testing has revealed that the server works well even on a relatively slow machine. In the graphs shown below, the server is under a constant load of 45 events per second, when a 70 event per second spike is received
The y-axis shows how many monitoring events remain unprocessed in the server's various queues at any given time. In the graph on the left, the server is running on one 800Mhz machine, and recovers from the spike in just over 80 seconds. In the graph on the right, the server is running on two 2.8Ghz machines and recovers from the spike in just 5 seconds. This demonstrates the scalability of the server's architecture.