SAMGRID WEB SERVICES

S. Veseli (FNAL)

SAMGrid

SAMGrid is a distributed (CORBA-based) HEP data handling system presently used by three running experiments at Fermilab: D0, CDF and MINOS. The system offers a wide variety of services, such as job management, data management, data transfer and storage, process accounting, etc. All services which require database accessare encapsulated within the SAMGrid *DB Server*. Examples of those are various cataloguing and dataset services. On the other hand, services involving data management, transfer and storage are provided by a set of SAMGrid *Station Servers*.

The primary means of accessing the SAMGrid system is via Python and C++ client APIs, but some of the cataloguing and dataset services are also provided by CGI scripts and Java Servlets. Python API incorporates all of the SAMGrid functionality, including various administrative and monitoring interfaces. It is distributed as a frozen binary with accompanying necessary shared object libraries.This technique has the advantage that users have full access to all of the SAMGrid interfaces, as well as to the standard Python modules, without worrying about possible compatibility issues related to a specific version of Python installed on a given system. On the other hand, SAMGrid C++ API has much less functionality and is targeted for use in experiments' C++ reconstruction and analysis software.

Why Web Services?

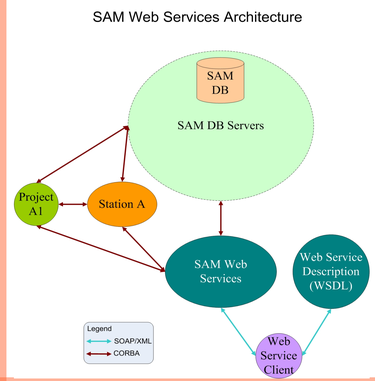
After several years of operations SAMGrid has evolved to be both robust and fault tolerant system. However, even though at this point the SAMGrid software is in a fairly mature state, there is still room for improvement, most notably in the areas of monitoring, software installation, configuration, and client access.

SAMGrid users are still facing non-trivial software installation and configuration issues. These issues usually do not represent a major obstacle for SAMGrid access from machines or clusters on which the software was installed and configured by the designated SAMGrid administrators. However, they make it difficult for regular users to setup their desktops or laptops for accessing SAMGrid via the distributed APIs.

SAMGrid Web Services have been designed to allow easy access to the system by using standard web service technologies and protocols (SOAP/XML, HTTP). In addition to hiding complexity of the system from users, these services eliminate the need for the proprietary CORBA-based clients, and also significantly simplify client installation and configuration.

Architecture

We have chosen Python as the implementation language for the SAMGrid Web Services. This allowed us to easily utilize the existing functionality in the SAMGrid Python API. We also used SOAPpy, a Python web service package.



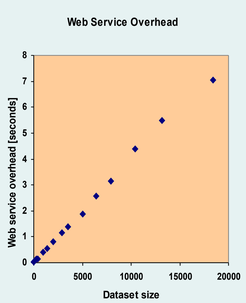
SOAPpy takes care of marshalling and un-marshalling SOAP messages, as well as of processing WSDL files. In addition to that, it comes with a reliable threaded service container, which greatly simplifies the service deployment.

For those WSDL interfaces that required communication to the database, implementation has been straightforward and mainly involved translation between CORBA and SOAP.

For the file delivery service we have established a protocol for retrieving data in small chunks.

Performance

For performance measurements we used the DB Server running on a 4-CPU Linux machine (2.4GHz Xeon processors) with 3.5GB of memory, while the web services were hosted on a dual Athlon MP 2000+ Linux node with 1GB of memory. In order to understand the overhead associated with the additional SOAP call, as well as with translating results returned by the DB Server into the corresponding SOAP/XML struct, we performed a series of identical back-to-back SOAP queries, and measured the time between sequential DB Server calls.

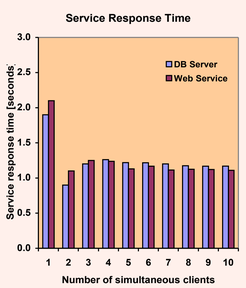


As expected, the average time necessary for additional SOAP processing grows linearly with the size of the SOAP message. The numbers shown in this figure correspond to about 12% of the total response time. Note that in most cases the observed overhead falls within variation of the direct DB Server query timing. For example, in the case of a query returning a list of about 2500 files, response times for 10 direct DB Server calls were in the range of 8.78 to 9.82 seconds, with the average of 9.24 seconds. The same query against the Web Service had response times between 9.04 and 9.88 seconds, with the average of 9.32 seconds for 10 calls.

Performance

Load tests involved a number of simultaneous clients invoking the same set of queries (unit of client work) 10 times. The average service response time per unit of client work is shown in the figure below. For a single client, the effects of SOAP overhead are clearly visible. For two clients, the average response time goes down because the DB Server had multiple connections to the database, and therefore queries could be executed in parallel. It is interesting that for larger number of clients the Web Service outperforms the DB Server. This is result of a more efficient usage of the DB Server proxies when the number of clients exceeds the number of available database connections.

Note that we performed the same test with 100 simultaneous clients without any performance degradation. The average service response time per unit of work was 1.12 seconds for the Web Service, and 1.16 seconds for the DB Server alone.



Future Work

Although the current functionality offered by the SAMGrid Web Services is sufficient for most of the regular system usage, several important pieces are still missing. Most notably, the ability for users to declare and store new files into the system is not there. In addition to the services that have to be added, some work is also needed to insure reliability of the file transfer service.

At this time the SAMGrid Web Services have been deployed in production for MINOS and are being tested for use at D0.