THE SAMGRID DATABASE SERVER COMPONENT: ITS UPGRADE INFRASTRUCTURE AND FUTURE DEVELOPMENT PATH

L. Loebel-Carpenter, S. White, A. Baranovski, G. Garzoglio, R. Herber, R. Illingworth, R. Kennedy, A. Kreymer, A. Kumar, L. Lueking, A. Lyon, W. Merritt, I. Terekhov, J. Trumbo, S. Veseli, FNAL, Batavia, IL 60510, USA

M. Burgon-Lyon, R. StDenis, Glasgow University, S. Belforte, INFN, Trieste

U. Kerzel, Karlsruhe University

V. Bartsch, M. Leslie, S. Stonjek Oxford University

F. Ratnikov, Rutgers University

A. Sill, Texas Tech University

Abstract

The SAMGrid Database Server encapsulates several important services, such as accessing file metadata and replica catalog, keeping track of the processing information, as well as providing the runtime support for SAMGrid station services. Recent deployment of the SAMGrid system for CDF has resulted in unification of the database schema used by CDF and D0, and the complexity of changes required for the unified metadata catalog has warranted a complete redesign of the DB Server.

We describe here the architecture and features of the new server. In particular, we discuss the new CORBA infrastructure that utilizes python wrapper classes around IDL structs and exceptions. Such infrastructure allows us to use the same code on both server and client sides, which in turn results in significantly improved code maintainability and easier development.

We also discuss future integration of the new server with an SBIR II project which is directed toward allowing the DB Server to access distributed databases, implemented in different DB systems and possibly using different schema.

INTRODUCTION

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. It 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 s being tested for use by MINOS and CMS.

DB Server Role in the SAMGrid System

Since its beginnings, the SAMGrid system has been using central Oracle RDBMS. Most of the communication with the database is done via a number of CORBA-based DB Servers, which encapsulate the SAMGrid cataloguing services (i.e., access to file metadata, event and replica catalog), dataset services (i.e., planning, creating, and manipulating user *dataset definitions*[*]). DB Servers also provide process accounting services, as well as the runtime support for SAMGrid *station services*[†] (e.g., file delivery and storage). The scope of these services alone indicates importance of the DB Server in the SAMGrid system. We further illustrate this in Figure 1, which shows the number of queries generated weekly by the D0 SAMGrid DB Servers over the period of three months (June-August '04). During this

[*] SAMGrid *Dataset definition* denotes a specification of file metadata, which is resolved to a list of files upon user request.

[†] *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).

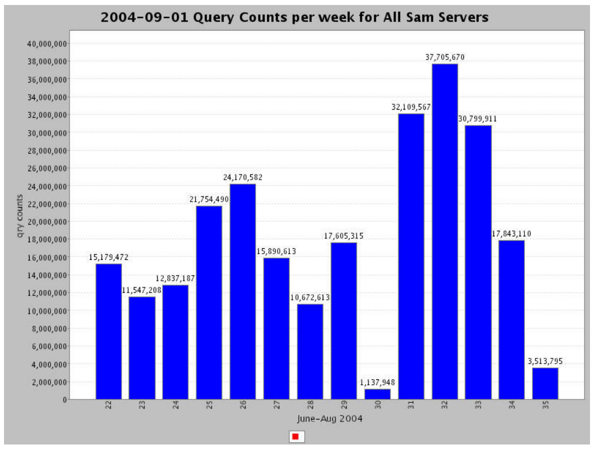

Figure 1: Number of DB queries generated weekly by the D0 SAMGrid DB Servers (June-August '04).

period, there were more than 250 million DB queries generated (on the average about 18 million queries per week).

PROBLEMS WITH EXISTING INFRASTRUCTURE

Even though the old DB server code has served the SAMGrid system well in the past, a number of problems related to code maintenance, development, and server performance, have accumulated over the years. As the data handling system developed, the DB server code base grew to about 27000 lines of python code with about 350 CORBA IDL methods defined and implemented. A large part of that code (about 60%) became obsolete and unused at some point. However, removing or modifying any given piece of the old code is fairly difficult, so that bug fixes often introduce a completely new set of problems that are not caught by our testing procedures. For this reason, any DB schema changes that were driven by experiments' requirements were not easy to implement regardless of their scope.

Another significant issue is related to performance. The existing server code is single threaded, so any long database query causes the DB Server to stop responding to client requests, which often leads to CORBA communication errors on the client side. Although we managed to partially mitigate this issue by using multiple DB Servers that are handling requests for different sets of clients,[‡] this problem still occurs rather frequently.

NEW DB SERVER DESIGN AND FEATURES

[‡] For example, at the moment the D0 production system uses 9 different DB Servers

Adoption of SAMGrid as the data handling system for CDF [2,3] has resulted in significant DB schema changes required to fully replace functionality embedded in the CDF Data File Catalog. Among other things, the description of file metadata has been redesigned in order to satisfy requirements of both D0 and CDF experiments. These schema changes, together with numerous problems with the existing server code, have led to the decision to update the DB Server infrastructure.

In addition to updating the treatment of file metadata, our primary goals in redesigning the DB Server were improved code maintainability, easier development, and improved server performance. At the same time we have kept some of the key pieces of the old infrastructure: python as the implementation language, CORBA as the communication protocol[§], and the automatic server base generation by the DB Server Generator [4]. The main features of the updated infrastructure are shown in Figure 2 and described in the following subsections.

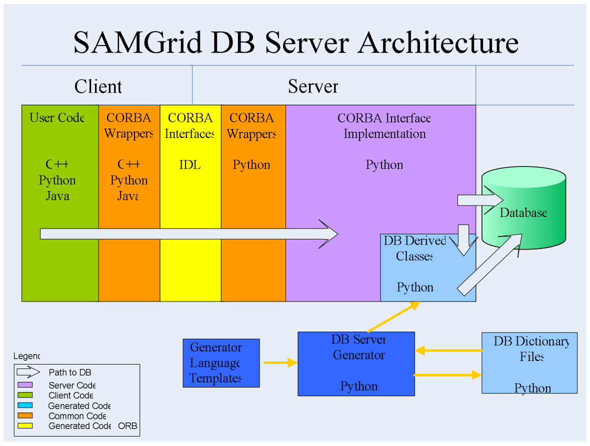

Figure 2: SAMGrid DB Server Architecture.

File Metadata

One of the main goals of the DB Server upgrade was updating the treatment of the file metadata. The old system of declaring files was outdated, and unable to cope with changing experiment needs. This was completely redesigned and aligned with the latest DB schema changes [3]. In the new system, file metadata is described as dictionaries (keyword-value pairs), but each different type of file has a certain set of required parameters that have to be present in the metadata. This not only allows us to enforce experiment requirements (these are configurable on a per-experiment basis), but it also gives us flexibility to handle any type of metadata and to easily adapt to any possible changes in the future.

[§] Note that in the new DB Server we have made a transition from Fnorb [5] to omniORB [6].

CORBA Interfaces

All of the existing IDL interfaces have been redesigned and reorganized so that they closely match the services which DB Server provides. At the same time, we have also introduced several new administrative interfaces for updating the SAMGrid DB. This will, for example, allow configuration of the SAMGrid station software before or during the installation phase, so that the station installation procedure can be fully automated. The current list of the most important interfaces is given bellow:

- Station Interface (run-time station support)
- Project Interface (run-time project support, process bookkeeping)
- Dataset Interface (manipulation of user datasets)
- Datafile Interface (metadata and replica catalog)
- Admin Interface (DH administration services)
- Station Admin Interface (station configuration and administration)
- Datalogger Interface (Online services)
- User Interface (VO management services)
- Autdodestinations Interface (file storage support)

CORBA Infrastructure

We have built a layer of code (CORBA Wrapper Classes) on top of the ORB-generated structs and exceptions with the purpose of shielding the developers from having to manipulate those directly. The wrapper classes can be initialized in a number of ways and used just like any other python (C++) class, and can be cast into the corresponding CORBA representation. In this way, creating, receiving, and sending CORBA structs (or throwing/catching CORBA exceptions) is done automatically in both server and client code.

This infrastructure clearly promotes greater code maintainability, easier development and code re-use. For example, our new python API uses the same infrastructure code that is used by the DB Server, and both client and server code is to a certain extent isolated from any future IDL changes.

DB Server Generator

While its CORBA interfaces and infrastructure have undergone significant modifications and improvements, one piece of the DB Server architecture has not changed: automatic generation of the core (DB-derived) classes using the DB Server Generator. As their name suggests, each of those classes corresponds to one database table, and can be used to easily query or modify that table. This, however, does not completely isolate the rest of the DB Server code from talking directly to the database for those cases where table joins are needed.

Multithreading

Addition of multithreading will have significant impact on the new DB Server and overall DH system performance. This will completely eliminate problems of client requests timing out because the server is waiting on completion of a complex database query requested earlier.

OUTSTANDING ISSUES

At the moment there are several outstanding issues with the new DB Server that are still being worked on. One of them is the impact of the new CORBA infrastructure with respect to the server performance. In most cases the additional overhead related to CORBA wrappers is not significant. However, creating large lists (on the order of 1000 or more) of objects (e.g., list of files in a dataset, or list of files cached on a given station) has a noticeable effect on overall DB Server performance.

We have not completely finished transferring all the functionality of the current DB server into the new version. The current system includes support for 'request systems' which allow users to enter job specifications into the database for later extraction by the data handling system to create local or grid job submissions. The request system parts of the schema are being revisited and hence the reimplementation of this support in the DB server has not yet been undertaken.

DEPLOYMENT PATH

The scope of changes in the SAMGrid software that is related to the DB Server upgrade is enormous: the DB Server itself was redesigned, our python API was rewritten to make use of the new common infrastructure, new infrastructure was developed for the station software, etc. Clearly, deployment of such changes into a production system without major disruptions to users is a difficult task. Our approach to this problem is deployment in three phases. In the first phase, which was completed in June '04, we upgraded the experiments' (D0 and CDF) production databases to the latest DB schema. Since the schema changes were not backwards compatible with the existing DB Server, the old server software had to be patched in order to function properly. The second phase (ongoing at the moment), involves deploying new DB Server in parallel to the existing infrastructure, as well as installing new client and station software to several sites for testing. In the third phase (planned for October '04) we'll start gradually upgrading the main production stations with the new software. The important point is that this upgrade will be incremental, so that its impact on both users and the DH system itself should be minimal.

INTEGRATION WITH SBIR II

As mentioned in the introduction, at the moment the SAMGrid database is centralized. Although performance of the Oracle DB has been very good in the past, the centralized DB is nevertheless a single point of failure in the system. For this reason we are currently investigating various possibilities for database replication. One of those possibilities is integration of our DB Server with SBIR II project [7], which strives to provide access to distributed databases with a single query. In collaboration with SBIR II developers we are working on interfaces which will allow us to plug different query mechanisms into our code.

CONCLUSIONS

In this paper we discussed the recent upgrade of the SAMGrid DB Server. We described the design and features of the new server architecture, which we believe will significantly improve performance of the entire DH system. We also outlined our plan for deployment into production, which is expected later this year.

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] D. Bonham et.al, "Adapting SAM for CDF", Proceedings of the 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03), La Jolla, California, 24-28 Mar 2003.

[3] A. Lyon, et.al., "New SAM Schema at D0: Description and Requirements", ,http://www-d0.fnal.gov/~lyon/samSchema/v03/SamNewSchemaDoc.pdf

[4] http://d0db.fnal.gov/db_server_gen/

[5] http://sourceforge.net/projects/fnorb

[6] http://omniorb.sourceforge.net

[7] M. Vranicar, "A Database Grid Solution", SBIR Phase II Proposal.