THE gLite AMGA METADATA CATALOGUE

B. Koblitz, N. Santos, CERN, Geneva, Switzerland
V. Pose, JINR, Dubna, Russia

Abstract

We describe the design and implementation of the AMGA Metadata Catalogue that is part of the EGEE project's gLite middleware. It will allow users of the LHC computing grid to access the metadata describing the 10 Petabytes of data taken annually when the Large Hadron Collider starts operating in 2007. To allow scalable and reliable access from the currently over 200 sites worldwide, the catalogue provides mechanisms to replicate relational data independently of the underlying relational database back-end. The catalogue provides in addition fine-grained access control based on grid certificates, suitable for highly confidential biomedical data.

INTRODUCTION

CERN is currently building the Large Hadron Collider (LHC), the largest physics experiment in the world. The LHC is a collaboration of several thousands of people, distributed over around 200 sites worldwide, and organised in four major experiments: Alice, Atlas, CMS and LHCb. The LHC will begin operation in 2007, when physicists working on the different experiments will start collecting and analysing data from high-energy proton or ion collisions in the range of several hundred MB/s up to over a GB/s. The data taking period will extend over several months per year and the total data size (including data replicated, reconstructed information and calibration data) will amount to an order of magnitude of 10 PB per year. The large majority of the data contains the so-called events, i.e. the recording of a given proton-proton collision in the detector, and will be stored in files. Some information, like the bookkeeping information describing the data, will be stored in databases. These data are often referred to as metadata and its accessibility is a key factor for an efficient and successful data processing.

The LHC experiments are using the EGEE [1] Grid infrastructure (as today, over 200 computer centres worldwide). Within the EGEE project a new middleware stack called gLite [2] is being developed. Both the infrastructure and the middleware are targeting a wider scientific community (together with LHC, several application areas are active in the EGEE project, notably Biomedical and Earth Observations).

The AMGA catalogue we describe in this paper is the metadata catalogue of the gLite middleware. AMGA was designed for high performance access to metadata, suitable for the demands of high energy applications. It also provides very fine-grained security features and integration into the authentication and authorisation framework of the gLite middleware, which are necessary to support e.g. the sensitive data of biomedical applications. After an overview of the design and the implementation in the next section, we present use cases of AMGA, performance measurements and give an overview on the replication features built into AMGA.

DESIGN AND IMPLEMENTATION

AMGA was designed to fulfil the needs of such different groups of users like High Energy Physics, which primarily demand high performance access, as well as biomedical sciences, which require very strict authentication and fine-grained access controls. In order to understand and meet the users' needs, AMGA was developed in close collaboration with the different user groups. As a result a common metadata access interface was designed [3]. It is not only implemented by AMGA but also by other catalogues like the AMI metadata catalogue of the Atlas experiment [4]) or the gLite FiReMan file-catalogue.

Interface Design

Metadata is defined within AMGA in terms of *schemas*, *entries* and *attributes*. *Schemas are* hierarchically organised collections of *entries*. Schemas in AMGA resemble strongly the directories in a filesystem and users address schemas via (relative) paths, while entries resemble the files in a filesystem. A schema has a list of *attributes*, which have a unique name for each schema and a storage type. Entries assign to each of the attributes of their schema a (NULL-)value. Within the database back-end where AMGA stores the metadata, schemas correspond to tables, attributes to columns of these tables and entries are the rows.

The metadata interface was designed to be modular allowing a service to implement only parts of it. As an example the FiReMan file catalogue or a version of AMGA running alongside the LFC file catalogue [5] both do not implement the parts of the interface used to handle entries as both file catalogues already have this capability. Another design decision was to be flexible with regards to the statefulness of a service. For example the retrieval of large result sets can be implemented using a stateless or stateful service.

Implementation

AMGA is implemented as a multi-process C++ server with a relational database back-end. It has two different front-ends, a webservice front-end implemented using gSOAP and a text-protocol based front-end, which is able to stream data back to the client. We will give here only a brief overview of the implementation of AMGA focusing on the features relevant for AMGA's performance or replication capabilities. A more detailed description can be found in [6].

AMGA is a service in front of a relational database system used as a storage back-end. Currently Oracle, PostgreSQL, MySQL and SQLite are supported. Requests received by either of the two front-ends are forwarded to the back-end, which streams data into a buffer shared with the front-end. This data is either streamed back to the user by the text-based front-end whereas the front-end using SOAP provides the user with an iterator mechanism to browse the result. The text-based protocol significantly outperforms the SOAP protocol [6], but only the SOAP front-end provides a fully compliant implementation of the interface as defined in [3]. Both access protocols support bulk operations where several entries are inserted or read in bulk.

Access control is supported either on a per schema or per entry basis. The default is per schema access control, where all entries in a schema share the same ACL (Access Control List). Per entry access control is also supported, at the cost of some performance degradation. The implementation also supports groups of users. Other security features are authentication based on certificates, grid-certificates or password, VOMS (Virtual Organisations Management System) [7] integration, and secure connections using SSL.

Schemas are implemented hierarchically in AMGA. This model has the advantage of being natural to users as it resembles a file-system, and of providing good scalability as metadata can be organised in subtrees that can be queried independently. In addition this implementation choice allows to support replication of only parts of the schema hierarchy or to setup federated schemas where remote (subtrees) of schemas are mounted into the local schema hierarchy.

AMGA IN USE

Several applications are currently using AMGA to store their metadata. Among them are the MDM (Medical Data Manager) [8] application implemented by the BioMedical community, the GANGA [9] physics analysis tool from the Atlas and LHCb experiments and a Grid library application called gLibrary [10].

The MDM application uses AMGA to store relational information on medical images stored on the grid, plus information on patients and doctors in several tables. User applications can retrieve images based on their metadata for further processing. Access restrictions are of the highest importance to the MDM application because the stored data is highly confidential. MDM therefore makes use of the fine-grained access restrictions of AMGA.

The GANGA application uses AMGA to store the status information of jobs running on the Grid that can be controlled by GANGA. AMGA's simple relational database features are mainly used to ensure consistency when several GANGA clients of the same user are accessing the stored information remotely.

Finally, the gLibrary project makes similar use of AMGA as the MDM application but provides many different schemas to store not only images but information on texts, movies or music. Another difference is that there is only a central librarian updating the library while for MDM updates are triggered by the many image acquisition systems themselves.

PERFORMANCE MEASUREMENTS

A metadata catalogue needs to be able to serve thousands of concurrent clients accessing the service and requesting possibly very large responses. A Grid service with similar requirements are the File Catalogues which also serve data from a database back-end. We can therefore compare the access speed to the catalogue with the speed of the LFC [5] and FiReMan File Catalogues access published in [11]. In order to compare similar operations on the catalogues, we set up a simple table mapping file-names to GUIDs (Global Unique IDs) and insert name-GUID pairs into this table as well as query for the GUID of a file. Both operations are very common for a File Catalogue.

We perform the tests on a dual Intel Xeon server with 2.4 GHz and 2GB of memory running the AMGA server and an Oracle database back-end used by all three services. Up to 100 concurrent clients are run in parallel on a single desktop computer. The setup is identical to that in [11]. All operations use SSL to encrypt the transferred data and X509 certificates for authentication. Figure 1 shows to the left the total number of entries entered into the respective catalogue as a function of the number of clients connecting concurrently via a LAN with a latency of $\approx 0.5\,\mathrm{ms}$. For inserts the performance of the LFC in session mode, AMGA in session mode and FiReMan in bulk mode is similar. However, in when AMGA uses streamed bulk operations in addition, it is about 10 times faster than the other catalogues and close to the limit of the database. The number of concurrent clients is limited to 20 for LFC due to problems with the heavyweight authentication procedure and 50 in FiReMan due to memory consumption of the large bulk messages. The limit to 100 concurrent requests in AMGA is due to the specific setup of the database backend. Figure 1 shows on the right the performance of the query operations. Again the streamed bulk access of AMGA is about an order of magnitude faster than the other catalogues'.

Figure 2 shows the performance for inserting and querying entries via a WAN with a latency of $\approx 300\,\mathrm{ms}$ such that the number of round-trips necessary for the operation dominates the result. The use of bulk operations is essential to

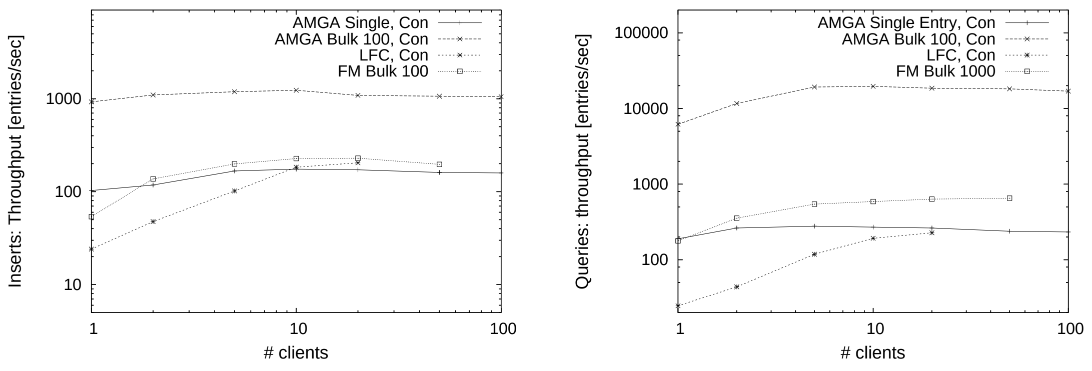

Figure 1: Total number of entries inserted (left) and returned in a query (right) for AMGA, LFC and FiReMan. Measurements done on a LAN with SSL security and authentication with X509 Certificates.

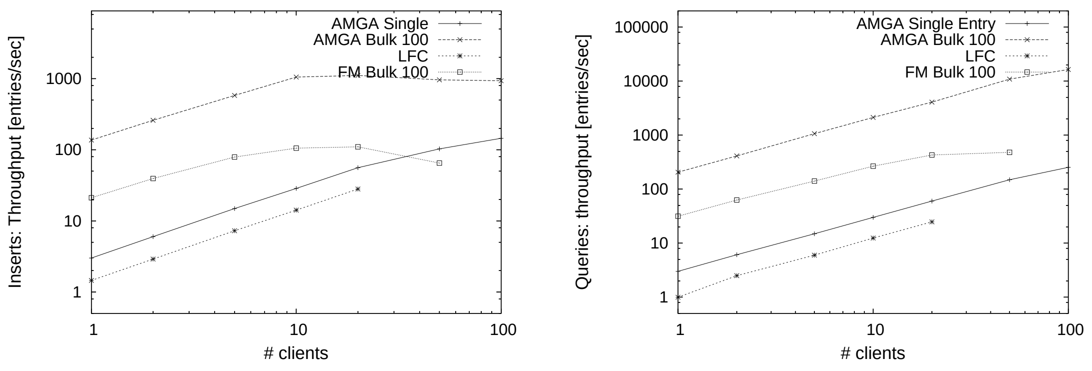

Figure 2: Total number of entries inserted (left) and returned in a query (right) for AMGA, LFC and FiReMan. Measurements done on a WAN ($\approx 300\,\mathrm{ms}$ latency) with SSL security and authentication with X509 Certificates.

reduce the number of round-trips as can be seen in the results for the bulk operations of FiReMan and AMGA, but also the streaming feature of AMGA provides an advantage for single entry operations as shows the comparison with LFC. The streamed bulk operations of AMGA are again about 10 times faster than the fastest other catalogue.

REPLICATION OF METADATA

To provide a reliable metadata service that scales with the size of a grid, multiple instances of such a metadata service on several sites are necessary. This can be achieved by *replicating* the metadata. Solutions by most of the various database vendors exist, which provide replication on the database level, but they do not provide replication across the heterogeneous database infrastructure of the grid, nor do these solutions provide support for resource allocation and distributed management for administrative users on the grid. AMGA has therefore been designed with replication in mind.

Replication is implemented in the AMGA server by recording all metadata commands that update a database in a special logging table, which is later sent to other AMGA servers listening to these changes. This provides a simple solution to replicate databases across different vendors. Another advantage of such a solution is that the authentication and authorisation mechanisms used on the Grid can be used to restrict the access to the administrative interface of the replication mechanism, whereas the administrative interfaces of the database replication do not support such a mode of operation.

AMGA implements asynchronous replication (that is slave services will not see updates of the master immediately but with a certain delay) with a single master per schema. Figure 3 shows the different replication modes AMGA supports, which all take advantage of the hierarchical structure of the schemas within AMGA. Supported are simple full replication of the entire table space (top left) as well as replication of only a sub-treee of the schema space (top right). Replication of only parts of the schema makes it also possible to share the load of updates of the database as long as different schemas are concerned.

Using the mechanisms for mounting parts of a master's schema space into the schema space of a slave, it is possible to combine the schemas of several metadata master services into a unified metadata system. Thus, federated storage of metadata is possible (shown in figure 3, bottom left). But AMGA will also be working as a proxy so that commands are forwarded to the service which has the necessary information to serve the query (bottom right).

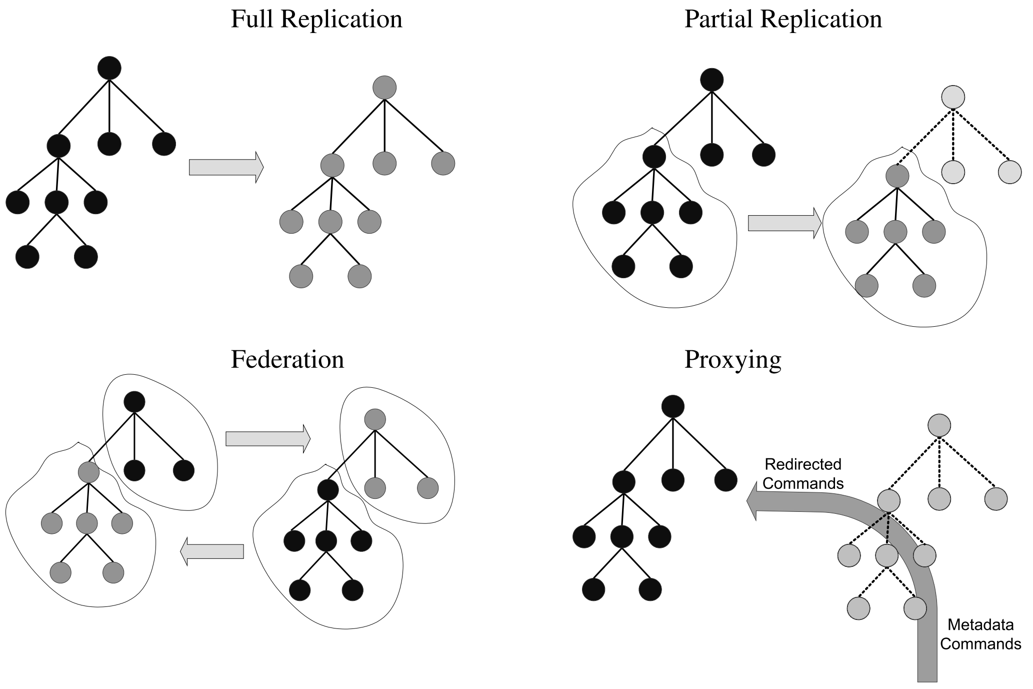

Figure 3: Different replication modes of AMGA, which take advantage of the hierarchy of schemas.

Federated storage and proxying is useful for applications that deal with highly confidential data as it allows the data to be stored in a federated manner, so that if the database back-end of an AMGA server is compromised, not all the metadata is compromised. Especially the biomedical community has expressed deep interest in these features.

SUMMARY AND OUTLOOK

We have presented the AMGA metadata catalogue as a high performance service to access relational data on the Grid for high energy physics applications and applications of other science communities. Our current work focuses on the replication of metadata as this is essential to provide scalable, reliable and high performance access to relational data on the Grid. So far only a simple initial implementation exists in AMGA. In the future we will work to improve in particular the recovery mechanisms and improve the performance via load balancing by proxying of requests.

ACKNOWLEDGEMENTS

The work presented in this paper was performed in the LCG ARDA project and we would like to thank our colleagues, in particular M. Lamanna, for many stimulating discussions. Many people have contributed to the design of the metadata interface and contributed ideas to the implementation of a metadata catalogue, among them the gLite Data Management team, the GridPP's metadata working group and the LHCb's GANGA and bookkeeping projects. This work was funded by Ministerium für Bildung und Forschung, Berlin, Germany, by the Portuguese Foundation for Science and Technology (FCT) and by the European Commission (contract INFSO-RI-508833).

REFERENCES

[1] Enabling Grids for E-sciencE: `http://public.eu-egee.org/`

[2] The gLite middleware: `http://glite.web.cern.ch/glite/`

[3] P. Kunszt *et al.:* `https://edms.cern.ch/file/573725/1.2/EGEE-TECH-573725-v1.2.pdf`

[4] T. Doherty: Development of gLite Web Service Based Security Components for the ATLAS Metadata Interface, *EGEE User Forum*, CERN, Switzerland, 2006.

[5] J. P. Baud, S. Lemaitre: The LHC File Catalogue (LFC), *HEPIX 2005*, Karlsruhe, Germany.

[6] N. Santos and B. Koblitz: Metadata services on the grid. In Proc. *Advanced Computing and Analysis Techniques*, ACAT'05, Zeuthen, Germany.

[7] R. Alfieri *et al.*: VOMS, an Authorization System for Virtual Organizations in Proc. *1st European Across Grids Conference*, Santiago de Compostela, Spain, 2003.

[8] J. Montagnat *et al.*: Bridging clinical information systems and grid middleware: a Medical Data Manager. In proc. *HealthGrid 2006*, Valencia, Spain.

[9] Ganga - Gaudi / Athena and Grid Alliance: `http://ganga.web.cern.ch/ganga/`

[10] T. Calanducci: gLibrary: a Multimedia Contents Management System on the Grid, *EGEE User Forum*, CERN, Switzerland, 2006.

[11] C. Munro and B. Koblitz: Performance Comparison of the LCG2 and gLite File Catalogues. In Proc. *Advanced Computing and Analysis Techniques*, ACAT'05, Zeuthen, Germany.