EXPERIENCES WITH THE GLITE GRID MIDDLEWARE

B. Koblitz*, CERN, Geneva, Switzerland
T. Chen, W. Ueng, Academica Sinica Computer Centre, Nanking, Taiwan;
J. Herrala*, M. Lamanna, D. Liko, A. Maier, J. Mockicki†, A. Peters, CERN, Geneva, Switzerland;
F. Orellana‡, CERN/University of Geneva, Switzerland;
V. Pose, Joint Institute for Nuclear Research, Dubna, Russia;
A. Demichev, Moscow State University, Moscow, Russia;
D. Feichtinger§, Swiss Institute of Particle Physics, Switzerland

Abstract

gLite is the grid middleware being developed by the EGEE project. In May 2004 a prototype was delivered to the ARDA project which has the task of providing prototype grid systems for the LHC experiments. We describe our first experiences with the gLite prototype and present the results of studies of its individual subsystems.

INTRODUCTION TO THE ARDA PROJECT AND GLITE

The ARDA project (A Roadmap to Distributed Analysis for LHC) was started in April 2004 to build grid-based analysis prototypes together with the four LHC experiments using the software developed by the EGEE (Enabling Grids for E-Sciences in Europe) project which was launched at the same time. To fulfill its task, the ARDA group works as an interface within the EGEE project to the LHC experiments, collecting and communicating the needs of the high energy physics (HEP) community to the EGEE middleware developers, and adapting existing experiment software to the EGEE framework.

gLite is the prototype software provided to the ARDA group by the EGEE project on May 18th 2004. It is strongly influenced by the Alien software developed by the ALICE collaboration and by the experience of many software projects among them EDG, Condor and Globus. The task of the EGEE project is to evolve this prototype into a fully featured and easily installable grid middleware. The time frame to achieve this is two years.

ACCESSING GLITE

The first contact to any grid middleware is made by a user by logging into a grid system. gLite uses X.509 certificates as grid credentials using the Globus 2.4 GSI (Grid Security Infrastructure) framework software to handle authentication, authorisation and encryption[1] via SSL and the GSSAPI.

In order to sign on to such a grid, the user first needs to acquire a certificate and sign up to a virtual organisation (VO) to become part of a group sharing resources on the grid. Unfortunately, these two steps turned out to be not easily overcome and it took some users several weeks to sign up: since the certificates are supposed to fulfill high security standards, personal identification was necessary at the issuing authority (CERN). Then the certificates had to be loaded into a web browser to request membership at the VOMS (virtual organisation management system). The overall procedure turned out to be complex and time consuming. Clearly there is room for improvement.

THE GLITE SHELL

gLite offers an interactive shell to users to access the grid services. The shell is implemented as a client which the user starts up and within which the user can issue commands similar to those in a standard Unix shell.

Central to the system is the file catalogue which is organised in a hierarchical way similar to a file system. Commands similar to *ls* or *rm* can be used to list or remove file. Adding files into the file catalogue can be done in two different ways: by adding a reference to an existing file in an accessible storage element or by specifying a file URL in which case the file is first copied to a (selectable) storage element before such a reference is made.

Since all shell commands withing gLite are implemented especially for gLite, several more advanced shell features are not implemented, namely the possibility to pipe output from a command into another one, thus denying the user to build more complex functionality using the existing commands as building blocks. Also, although over 70 commands are provided, the functionality of the shell is far from the functionality of several hundreds of commands offered by a normal Unix shell.

A solution to these problems can only come from adding command line tools which can be called from within a standard Unix shell. To build these stand-alone clients, the

* Funded by the German Ministry for Education and Science, Berlin, Germany

† Funded by Particle Physics and Astronomy Research Council, UK

‡ Financial support from the Paul Scherrer Institute and the European COST Action Nr.283

§ Financial support by the European COST Action Nr.283

[1] The connections from the user client to the services are currently not encrypted.

ARDA project has implemented a C API and created a library of POSIX file access and directory browsing functions which are used to implement the necessary commands, as described later.

JOB SUBMISSION

Jobs can be submitted from within the gLite shell via a small job description script which specifies which files are to be first copied to the node, which executable is then to be run and which files are afterwards to be copied back into the grid. For this to work, the job description file and the executable need to be accessible in the virtual grid filesystem.

The job submission scripts also allow more advanced features like forwarding arguments given to the submit command to the application later run, the definition of packages which contain software installed on the nodes in some specialised disk area prior to starting the job's application and even the splitting of jobs based on directories or files automatically. This job splitting allows the creation of a master job which can be later referenced for handling the whole group of created sub-jobs. For tracing this group of files, killing the job or retrieving output, all operations can be done to the master job and will apply implicitly also to the rest of the sub-jobs.

Currently jobs are run on three worker nodes located at CERN (two) and in Madison (one), which form two computing elements which are seen by gLite as two different entry points. The operating systems are CERN Linux 7.3 and Scientific Linux 3 (one node at CERN). During job submission only the compute element can be chosen, no option is available to select the operating system which must be provided by the worker node.

The ARDA team has set up a monitoring system for the job processing system of gLite, which submits a simple job every hour into the job queue of the compute element at CERN. The success of these jobs can be seen in Fig.1 where the time it took a successfully run job to start up after submission is plotted versus the time of the submission of the job. Failing or crashing jobs are also indicated. In total a success rate of jobs of 80% is achieved, but this number must be related to the fact that only a few worker nodes are being used and the job only returns a simple string, that is, it does not depend on further external resources. The low number of worker nodes effectively prevents problems with badly configured worker nodes which may attract jobs which afterwards fail, a common problem for larger scale systems.

PACKAGE MANAGEMENT

The possibility to create software packages has already been mentioned in the description of the job submission process. This capability of gLite allows the user to require that certain tar archives, which are accessible via the virtual filesystem of gLite, are unpacked on the worker node into a dedicated disk space prior to the job's binary being started. These packages can contain additional libraries, additional executables or common data needed for job execution. The job description files allow the users to specify scripts which are run before and after installation of the packages which allows the user to set up these packages, for example setting up the PATH or LD_LIBRARY_PATH variables. These scripts also alleviate the problems caused by the fact that the packages are unpacked into a space in the filesystem of the worker node a priory unknown during package creation.

While the job submission scripts allow a high flexibility to allow basically all software packages to be set up out of a tar archive, this flexibility has to be weighted against the inflexibility for requiring certain capabilities of the the worker node in the first place by stating them in the job submission script. The burden to create a suitable environment for jobs to run on the node lies entirely by the user: he needs to provide every software package he needs, even C compilers or certain system libraries, which need to be compiled by the user and set up correctly — a task that is potentially very difficult. It would be a great relief for the user to be able to set certain requirements at least on system libraries which the worker nodes need to provide in order to be able to run certain jobs. However, the greatest relief for the user would certainly be if capabilities of worker nodes could be made available temporarily on demand like installation of additional system libraries in a separate space of the worker nodes file system.

FILE ACCESS

Files can be accessed from the prototype installation of gLite in two storage elements (SE): the CASTOR[2] and dCache[1] storage systems which both are disk-based caching frontends to tape-based large data stores. Both SEs have been set up twice in Madison and at CERN.

The availability of the SEs are paramount for the ARDA project because they allow access to experiment Monte Carlo generated data files already present on tape storage. This data is needed to run jobs which are already run by the experiments for processing event data. This capability is also important to attract experiment users who would want to perform their individual analysis tasks.

The CASTOR storage element was the first to be set up and the process revealed many problems of the deployment of grid middleware which needs to access existing resources, in this case the existing CASTOR system at CERN. Since users of the grid lose their identity in the sense of a UNIX user, and are only identified by certificates, they have to access a resource like CASTOR, which was designed to allow UNIX users access based on their user ID, via a single user which has no access restrictions. Setting up CASTOR correctly and the storage elements software which takes over access permissions checking, took several months and was mainly hindered by the communication between the concerned parties from the gLite

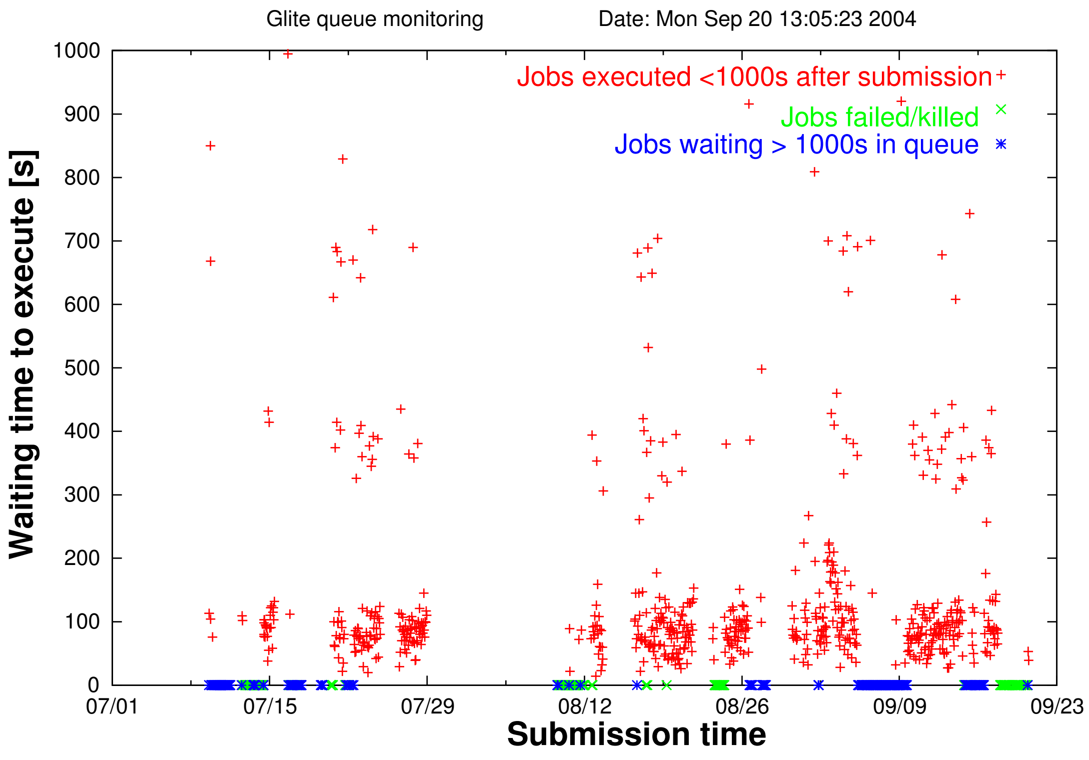

Figure 1: Plot showing the failure rate and the waiting time until execution starts for successful jobs. The rate of success is about 80% for very simple jobs (see text).

development and deployment team as well as users and the people responsible for the storage system.

METADATA

The gLite prototype software stores its metadata together with file related data in the file catalogue which is using a MySQL RDBMS backend to store the data. The user can updload database table schemas into the file catalogue and declare them to be the metadata schema of all files within a directory. Internally, the possible metadata keys form the columns of a table which has all the files of a directory as its rows. The user can later on associate values on a per file basis with these keys.

Fig.2 first shows the performance of the catalogue for inserting files and associating metadata. The performance degradation for large directories is due to the fact that the files' data is stored in a single large table, which degrades the performance of the MySQL backend.

ARDA has studied the behaviour and performance of several experiment metadata solutions (see [3]) with an emphasis on the behaviour of the performance of concurrent access of many clients with queries generating large responses. Implementing a solution which scales for such queries using a web-service based client interface is a demanding task. gLite uses a streaming-based solution where the result is transferred to the client as text. Fig.3 shows the behaviour of gLite for such an access pattern. As can be seen, the response time to fulfill all of the concurrent

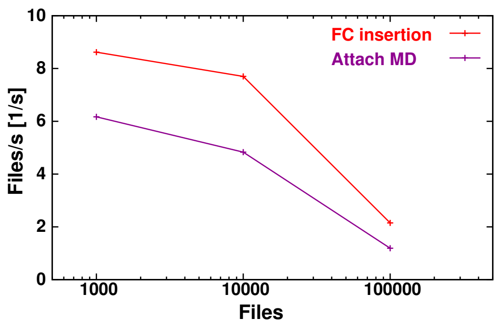

Figure 2: Upper curve: time to insert files into the Metadata catalogue (only adding a reference to a file already stored in a Storage Element!). Lower curve: time to attache some metadata to these files (5 properties, 36 bytes in total).

requests rises linearly with the growing number of clients indicating the good scalability of the server model when dealing with many clients as well as a low memory footprint of the processes serving the requests. This behaviour contrasts strongly with other implementations, which first prepare the full response, pack them into e.g. a SOAP message and dispatch them eventually. This leads to a breakdown of the server performance for either large responses or large numbers of concurrent client connections.

The users can define schemas in gLite flexibly, as it allows to define arbitrary keys with arbitrary data types, as long as they are supported by the underlying SQL backend.

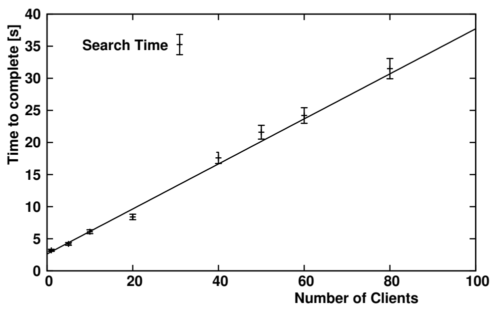

Figure 3: Time to complete several concurrent search request by a growing number of clients selecting 2500 files out of 10000 files stored in a directory. The total time scales linearly with the number of clients showing the good scalability of the server with the number of clients and the total size of the data to be served.

In addition, the solution is very performant because searching in the SQL tables can be done within the database itself and since the tables are stored in the same database as the file related data. In this case searches for files fulfilling certain criteria with their metadata are easily implemented and performant. The drawback is that schemas cannot be evolved and the application must be aware of the underlying schema since there is no way the schema can be discovered. However, it seems such features could be added to the existing implementation.

FUTURE DEVELOPMENTS

One of the largest problems for the ARDA developers who want to adapt existing experiment software to work within the gLite grid was the lack of a C/C++ API. ARDA has therefore developed a generic C/C++ interface to the services of gLite. The interface consists of a generic web-service and a client library with a C/C++ API. The client sends the PERL structures, which are used to communicate with the gLite services, by first UUEncodeing their data, then encrypt the resulting ASCII-only text using OpenSSL and finally send the command using the SOAP protocol. This approach thus provides in addition generic encrypted web-services to all gLite functionality. This increases the performance of the encrypted service: the clients first perform an authentication procedure during which shared secrets are exchanged between client and server, which later makes communication possible without the need for further authentication steps.

SUMMARY

The EGEE project delivered a fully working prototype of grid middleware which is now being evaluated by the ARDA team and used to set up grid middleware prototypes for the four HEP experiments at LHC. In the future this prototype is expected to evolve based on the experiences gained by users of these analysis prototypes. The collaboration with the EGEE project team was very successful and many problems have been solved. The prototype looks very promising with the functionality it offered.

ACKNOWLEDGEMENTS

The ARDA project members wish to express their gratitude to the many EGEE developers which have strived very hard to develop and deploy a working prototype of the gLite software which is now used as a basis for grid services for the LHC experiments. We would also like to thank all the EGEE members for the excellent collaboration.

REFERENCES

[1] http://www.dcache.org

[2] http://castor.web.cern.ch/castor/

[3] J. Andreeva *et al.* [ARDA project], http://lcg.web.cern.ch/-lcg/PEB/arda/public_docs/CaseStudies/refdb_draft_v0.2.pdf; B. Koblitz *et al.* [ARDA project], http://lcg.web.cern.ch/-lcg/PEB/arda/public_docs/CaseStudies/ami_new.pdf