THE ARCHITECTURE OF THE ALIEN SYSTEM

P. Buncic, J. F. Grosse-Oetringhaus, A. J. Peters, P. Saiz, CERN, Geneva, Switzerland

Abstract

AliEn (ALICE Environment) is a Grid framework developed by the Alice Collaboration and used in production for more than 3 years. This paper presents the architecture of selected components of the AliEn system and describes its evolution in context of the EGEE* project.

INTRODUCTION

Unlike most mainstream Grid-implementations at that time, the AliEn system [1] has been from the very beginning based on Web Services and standard protocols. In a very short time, the ALICE collaboration [2] had a prototype that, while constantly evolving, allowed large distributed simulation and reconstruction vital to the design of the experiment's hardware and verification of the off-line computing model.

The system has been deployed for ALICE users at the end of 2001 for distributed production of Monte Carlo data, detector simulation and reconstruction at over 40 sites located on four continents. Up to the present, more than 400,000 ALICE production and analysis jobs have been run under AliEn control worldwide during Physics and Data Challenge exercises.

Following the report and recommendations of ARDA Requirements and Technical Assessment Group [3] of the LCG Project, the architecture of AliEn and its Web Services model were re-factored, taking into account input from other similar projects (EDG, VDT and others) into an architecture that became the basis for the next generation of middleware. This middleware, named gLite, is currently being developed in the framework of the EGEE project [4].

In the gLite prototype, parts of AliEn are used to provide an initial implementation of components and services: the File and Metadata catalogue, Task Queue, Package Manager, various user interfaces such as the command line prompt, an application API and the corresponding Grid Access Service (GAS).

The development effort is now focused on meeting the specific needs of the EGEE project, implementing the EGEE architecture and fulfilling the requirements of the wider user community. Particular attention is being paid to the conformity with a stringent set of security requirements and the EGEE software development process.

* EGEE is a project funded by the European Union under contract number INFSO-RI-508833

To fulfil the above requirements, a new development cycle was triggered during which the AliEn Catalogue has been split into independent File and Metadata Catalogue Services, a Package Manager Service was created out of the former Package Manager Component and a Grid Access Service was developed in compliance with the EGEE security model.

SERVICES AND COMPONENTS

The rest of this paper will briefly describe these re-factored AliEn services and show how they fit in the gLite architecture.

User Interface and API

The user interacts with the system through a User Interface layer. The interaction can be carried out from a command line interface with the most common Unix shell commands being implemented or via a Web portal which provides a convenient way to submit, inspect and manipulate a large number of jobs running at many sites. The application level access to AliEn is provided through an API. AliEn provides a Perl, C and C++ API.

Grid Access Service (GAS)

The Grid Access Service (GAS) provides access to different underlying Grid services (Figure 1). It acts as an adapter and exposes a flat interface to the collection of service components: e.g. the File and Metadata catalogue, Workload management, Data Management etc.

This collection is exposed to the user via the user interface or to the application by means of an API.

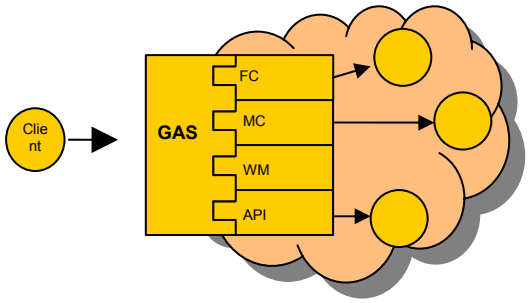

Figure 1 : Grid Access Service

Since the changes to the interfaces of the underlying services should not affect the interface of the GAS, the client applications remain reasonably well isolated from the changes in the middleware. Furthermore this allows for a seamless migration from one service implementation

to another. For example, in the present gLite prototype, the user can choose between AliEn Metadata Catalogue and an alternative implementation of a Metadata Catalogue, developed specifically for biomedical application.
The GAS itself is a Web Service following the WSRF [5] model and it runs under gContainer, a Web Service Container supporting WSRF. The gContainer includes service discovery and load management. For this purposes it communicates with other instances of gContainer. The core of the gContainer is a WSRF::Lite Container which is a secure HTTP server that is part of WSRF::Lite [6] toolkit and is suitable for running the Web Services written in Perl. The gContainer instance consists of a WSRF::Lite Container and a stateful management service called gController. Other services are created via the stateless factory service (gFactory).

Authentication and Authorization

Figure 2 illustrates the login use case using the GAS. The authentication follows the pattern often used to implement Grid portals so that GAS can be seen as an ad-hoc user portal.

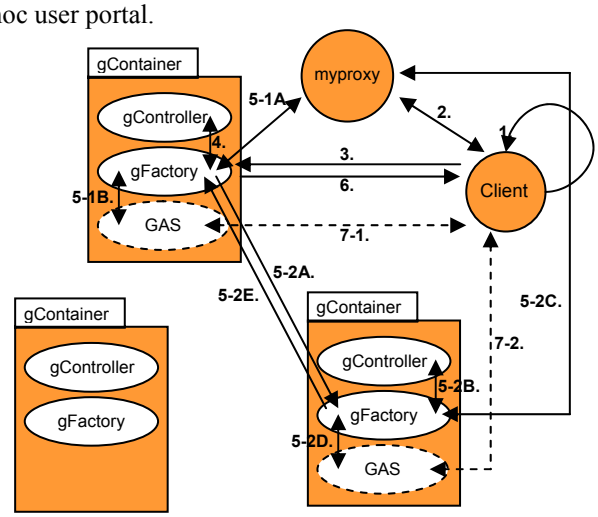

Figure 2 : GAS login use case

Before connecting to GAS, a user creates a proxy certificate (1) and stores it in the myproxy server (2). In the next step, the user connects to the gFactory service running under an arbitrary gContainer and submits a session request (3). The gFactory queries the gController for the best possible locations to create an instance of GAS (4), retrieves the proxy from the myproxy server (5-1A), creates the GAS (5-1B) and finally returns the address to the user (6). Afterwards the user talks directly to the newly created service (7-1).

If the service needs to be created within a different gContainer the steps (5-1A) and (5-1B) are replaced as follows: the gFactory connects to the gFactory service at the remote gContainer (5-2A) and submits a session request. The gFactory at the remote location follows the sequence 5-2B to 5-2D and returns the address (5-2E) to the initial gFactory which returns it to the user.

If in addition to myproxy service we also use Virtual Organization Membership Service [7] (as will be the case in gLite), the system will be able to take into account user roles when creating the GAS interface.

Workload Management

In contrast to the push model traditionally implemented in other Grid systems, the AliEn Workload Management is based on pull architecture [8] which is also applied to Data Management. The job description (JDL) in the form of the Condor ClassAd, is kept in a Task Queue while waiting for the Computing Elements to advertise their status and capabilities and to request jobs. While the jobs are waiting in the Task Queue, the Job Optimizers will inspect the JDLs, optimize and order the requests. The system can trigger a file replication in order to make a job eligible to run on a specific site in order to balance the overall load or enforce specific policies.
In a possible deployment scenario (Figure 3), the AliEn Workload Management can be integrated with the EDG/LCG Resource Broker in complementary fashion and allow a user to choose if he wants the job to be inserted directly into the Task Queue and be handled by the AliEn Workload Management based on the pull model, or to hand over the job to the Resource Broker. In the latter case, the Logging and Bookkeeping component of WMS will notify the Task Queue about existence of a new job and its status so that a new entry can be inserted into the Task Queue which is essential for a consistent view of the system from user perspective. Alternatively, both systems can be deployed concurrently with no mutual interaction but still sharing the resource (CE) and user (GAS) entry points.

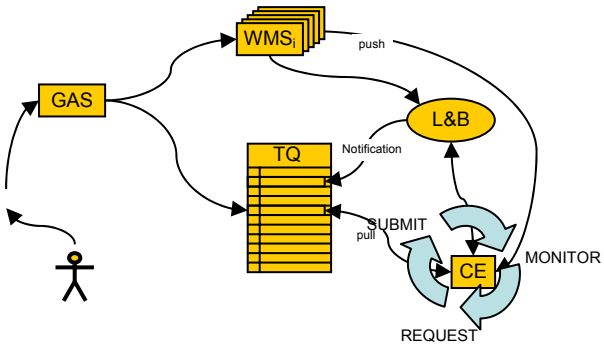

Figure 3 : Possible common deployment scenario of AliEn Task Queue and EDG/LCG Resource Broker

The AliEn Computing Element (CE) is effectively an agent that provides an interface to the local batch system. The task of a CE is to gather information about the status of local CPU resources, installed packages and policies and periodically advertise this information to the Job Manager. Upon successful matchmaking, it will get the

job JDL, translate it to the syntax appropriate for the local batch system and execute it. As a matter of strategic choice, only the Condor interface is used in gLite. In turn, Condor will provide reliable interfaces to multiple batch system back-ends. Once sent to the batch system, each job is wrapped up in another Web Service, the Job Agent, allowing users to interact with the running job, send a signal or inspect the output. Prior to job execution, the Job Agent can automatically install the software packages required by the job using the Package Manager Service. The Job Agent has a large part of the CE functionality built in and can autonomously advertise itself using the approach similar to one pioneered by Dirac system [9].

File and Metadata Catalogue

The File Catalogue (Figure 4) is designed to allow each directory node in the hierarchy to be supported by different database engines, possibly running on different hosts and even having different internal table structures optimized for a particular directory branch. This assures scalability of the system and allows growth of the catalogue as the files accumulate over the years of data taking. For example, the current File Catalogue for the ALICE experiment already contains 9 million entries.

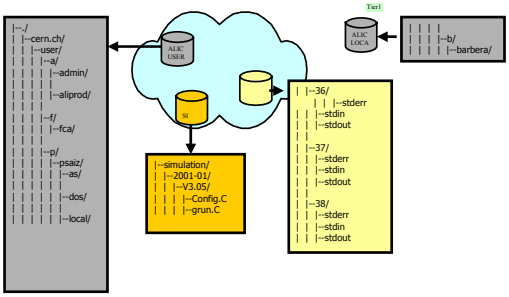

Figure 4 : AliEn File Catalogue, based on distributed set of databases

The File Catalogue extends the familiar file system paradigm to include information about running processes in the system (in analogy to the /proc directory on Linux systems). Each job inserted into AliEn Task Queue gets a unique id and a corresponding /proc/id directory where it can register temporary files, standard input and output as well as all job products.

The directories and files in the File Catalogue have Unix like privileges. This means that every user or a group can have exclusive read and write privileges for his portion of the logical file namespace (home directory). In gLite, this will be extended into a full Access Control List (ACL) model.

The File Catalogue keeps an association between the Logical File Name (LFN), an immutable file identifier (GUID) and Storage File Names (SURL). The system supports file replication and caching and will use this information when it comes to scheduling jobs for execution. To improve scalability and responsiveness of the system, the gLite implementation will require splitting of the File and Replica Catalogue functionalities. Instead of handling SURLs, the File Catalogue in gLite (Figure 5) will simply hold information about the list of Sites or Storage Elements where information about SURLs is kept in the Local Replica Catalogue.

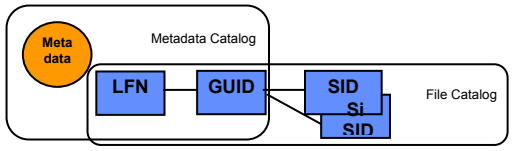

Figure 5 : File, Metadata and Local Replica Catalogue in gLite

The hierarchy of files and directories in the AliEn File Catalogue is reflected in the structure of the underlying database tables. It is possible to attach to a given directory an arbitrary number of additional database tables, each one having a different structure and possibly different access rights and containing metadata information that further describes the content of the files in a given directory thus building the AliEn Metadata Catalogue.

Data Management

The Data Management services in AliEn have, in full analogy to the Workload Management, a set of central components (File Transfer Queue, Brokers and Optimizers) and a site component (File Transfer Daemon, FTD). The FTD runs typically on the same host as the Storage Element and provides scheduled file transfer functionality. In the context of gLite, this part will be replaced by components coming from other projects but will still follow the same architectural model.

The Storage Element (SE) in AliEn is responsible for saving and retrieving files to and from the local storage. It manages disk space for files and maintains the cache for temporary files.

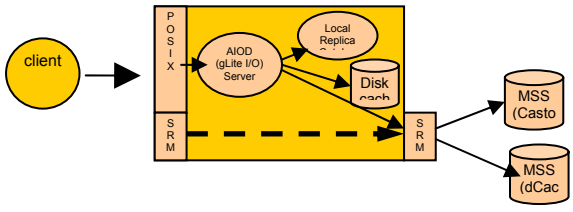

Figure 6 : Intelligent gLite Storage Element aggregating POSIX and SRM interfaces

In the gLite implementation (Figure 6), a set of services that work together (gLite-I/O server, Local Replica Catalogue and SRM based interface to Mass Storage Systems) can be logically seen as an intelligent Storage Element that will be a building block for deployment. It offers a POSIX like I/O capability by means of the gLite-I/O server, based on AIOD service of AliEn [10] and

enhanced with GSI security and a control channel similar to the SRM interface.

Package Manager

This service is a helper service that automates the process of installing, upgrading, configuring, and removing software packages from a shared area (software cache) on a Grid site. The Package Manager Service does not manage the installation of middleware software; it manages the packages that are common for all users of a VO and possibly packages provided by individual users.

The packages are installed on demand, when requested by the Job Agent running on a worker node or upon an explicit request by the VO Software Manager. The Package Manager checks if the requested package is already installed in the cache and if that is not the case, it proceeds with the installation. The right version of the package is downloaded from the File Catalogue and installed in the directory specified as a service configuration parameter. It will also install all the dependencies required by the package. The Package Manager returns a string with a command (a shell script) that the client has to execute to configure the package and all its dependencies. This script performs actions like setting environment variables.

The package installation can be triggered by a process running on the worker node or by a person with appropriate privileges (identified by the certificate subject). In both cases the requestor obtains the lease for the package for a specified time. The Package Manager manages the local disk cache and will clear the disk space if it needs the disk space to install newer packages but it will not remove the packages for which someone holds the lease. The maximum lease time for the packages is a configurable parameter. While any user or process can list already installed packages, only the VO administrator can remove a package from the local cache regardless of its current lease status and in that case the currently running jobs requiring that package might fail. Removing a package does not remove the packages that depend on it. If any of those packages are used, the removed package dependency will be automatically installed again.

Auditing and Accounting

A distributed Logger Service provides a mechanism for all services to report their status and error conditions. This allows the Grid manager to monitor all exceptions in the system and to take corrective action. Within gLite, auditing and accounting records will be also passed to an R-GMA [11] information system and logging service based on the log4j model.

Monitoring

Since the AliEn Resource Brokers do not depend directly on sophisticated monitoring information for scheduling, we did not develop any special monitoring tools. Instead, we deployed the MonALISA framework [12] and used it extensively to understand the behaviour of the system. In gLite, the information repository will be based on the R-GMA model.

CONCLUSIONS

The AliEn Grid framework was developed during the past three and a half years by a small team (between two and four developers), working closely with an experiment and deploying the software in rapid, extreme programming cycles. The use of scripting language like Perl and its OO features helped maintain this cycle and allowed for an easy integration of external software components in the form of reusable Perl modules. The result of this development was a service oriented architecture and an implementation that was continuously tested and deployed to more than 40 ALICE computing sites. Focusing on industry standards and trends, but always leaving room for evolution and alternative solutions was a good strategy as we could easily accommodate variations of the technology (e.g. OGSI to WSRF transition). The choice of Web Services as a core concept helped the quick prototyping, but in a complex and large project like gLite we are encountering interoperability problems with handling of services implemented in different programming languages. In addition, the observed performance penalties seem to be large enough that in certain cases it might be preferable to implement client-server communication using alternative protocols. This is a commonly recognized problem and the expectation is that the initiatives like Fast Web Services [13] will provide a solution.

REFERENCES

[1] P. Buncic et al.: The AliEn System, status and perspectives, CHEP'03, http://arXiv.org/abs/cs/0306067

[2] http://www.cern.ch/alice

[3] http://lcg.web.cern.ch/LCG/peb/arda

[4] http://www.eu-egee.org

[5] http://www.globus.org/wsrf/

[6] http://www.sve.man.ac.uk/Research/AtoZ/ILCT

[7] http://hep-project-grid-scg.web.cern.ch/hep-project-grid-scg/voms.html

[8] P. Saiz et al.: "AliEn Resource Brokers", CHEP 03, http://arxiv.org/abs/cs/0306068

[9] N. Brook at al.: DIRAC - Distributed Infrastructure with Remote Agent Control, CHEP03, http://arxiv.org/abs/cs.dc/0306060

[10] A. J. Peters at al.: AliEnFS - a Linux File System for the AliEn Grid Services , CHEP 03, http://arxiv.org/abs/cs/0306071

[11] http://www.r-gma.org

[12] http://monalisa.cacr.caltech.edu

[13] http://java.sun.com