AliEn: ALICE Environment on the GRID

S. Bagnasco, L. Betev, P. Buncic, F. Carminati, C. Cirstoiu, C. Grigoras, A. Hayrapetyan, A. Harutyunyan, A.J. Peters, P. Saiz

CERN, European Organization for Nuclear Research, 1211 Geneve 23, Switzerland

E-mail: pablo.saiz@cern.ch

Abstract. Starting from mid-2008, the ALICE detector at CERN LHC will collect data at a rate of 4PB per year. ALICE will use exclusively distributed Grid resources to store, process and analyse this data. The top-level management of the Grid resources is done through the AliEn (ALICE Environment) system, which is in continuous development since year 2000. AliEn presents several original solutions, which have shown their viability in a number of large exercises of increasing complexity called Data Challenges. This paper describes the AliEn architecture: Job Management, Data Management and UI. The current status of AliEn will be illustrated, as well as the performance of the system during the data challenges. The paper also describes the future AliEn development roadmap.

1. Introduction

AliEn [1] is a set of middleware tools and services that implement a Grid infrastructure. The development of AliEn started in year 2000 by the ALICE collaboration [2], and it was very quickly deployed for distributed Monte Carlo productions on several remote computing sites. The fast developing cycle continued, adding more functionality to the system. One of the most important features was the interfaces to other Grid implementations, which were appearing and were being adopted around the world. Thanks to these interfaces, AliEn can be used not only as a standalone Grid, but also in collaboration with existing grids. Since 2005, AliEn has been used both for data production and end-user analysis. During all these years, AliEn has very successfuly served its primary goal of hiding the complexity and heterogeniety of the underlying Grid services from the end user, even as the Grid technology was rapidly evolving. The basic AliEn components are as folows:

- File catalogue with metadata capablities.
- Data management tools for data transfers and storage.
- Authentication and authorization.
- Workload management system.
- Interfaces to other Grid implementations (ARC [3], OSG [5], GLITE [6]).
- ROOT interface [7].
- Monitoring.

AliEn was primarily developed by ALICE, however it has been adopted by other Virtual Organizations. Presently, it is used by PANDA [8] and MAGIC5 [9].

2. File catalogue

The file catalogue is one of the key components of the AliEn system. Unlike real file systems, it does not own the files but provides the mapping between the visible to the end user Logical File Names (LFN) and one or more Physical File Names (PFN). Multiple PFNs are remotely stored replicas of the same file. The PFN entries in the catalogue describe the physical location of the files by identifying the name of the AliEn storage element and the path to the local file.

LFNs can be manipulated by the user. Moreover, two different LFNs can point to the same PFN. To prevent duplicate file entries, each LFN is associated to a Globally Unique IDentifier (GUID) entry.

The interface to the file catalogue is similar to a UNIX file system. It is a hierarchical structure of files and directories. The catalogue also provides tools to associate user-defined metainformation to any entry.

The file catalogue is used by almost all components of AliEn. It contains information on all data and software packages used for data production and analysis. The job output is also registered in the catalogue, under the /proc/<jobid> (in analogy to the /proc file system in Linux). The metadata definitions and various triggers used for automatic data management or processing, are also registered in the catalogue.

The major file catalogue improvements in the past years have been:

- Database layout
- File collections
- Triggers

The rest of this chapter describes in more details these components.

2.1. Database layout

The file catalogue is built on top of relational databases. The catalogue design is such that it can be split in different databases at the directory level. The databases could be stored on different machines, thus assuring a high level of scalability. In the most recent version of the catalogue, this feature is also extended to the GUID space.

The structure of the catalogue is shown in Figure 1. From the user point of view, the catalogue behaves as a single entity. However the internal structure is divided between the LFN and the GUID catalogue, which are kept in independent databases.

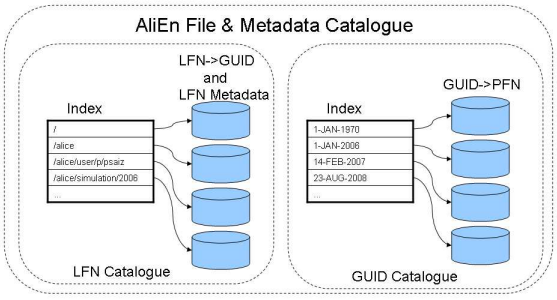

Figure 1. Structure of the AliEn File and Metadata Catalogue

Every LFN database has an index table. The index is used to locate the database and table that contains all the information of a directory or file. The next set of tables give the basic information about the LFN: owner, group, date, size, GUID, etc. In addition, there could be some user-defined tables that contain metadata information for any entry in the catalogue.

The structure of the GUID part of the catalogue is quite similar. The first table is an index pointing to the table and database that contains a specific GUID. The index is calculated based on the creation time of the GUID. This way, GUIDs created at the same time will be likely to be in the same database. For each GUID we can define two lists of storage elements (SE): the first list is of the SE that have the GUID, and can build up the PFN based on the GUID name; the second list contains the SE that cannot build the PFN directly. In this second case, we store as well the PFN in the central catalogue.

In order to do a LFN to PFN translation, the following steps have to be taken:

- Look up in the index for the database containing the LFN
- Do LFN to GUID translation
- Look up in the GUID in the GUID index
- Get all the PFNs from that GUID

There are several advantages of this two-step translation. First of all, it gives the possibility to work directly with GUIDs instead of LFNs, which is the natural namespace for the data management. Moreover, if a file with a given GUID is replicated, there is no need to update anything in the LFN databases. All the LFNs that point to that GUID will be aware of this new replica. Also, users can rename the LFN, and this does not affect the GUID catalogue.

The major drawback of the separation between the LFN and GUID catalogues is the risk of having orphaned GUIDs: if all the references to a GUID are deleted from the LFN catalogue, the GUID itself should also be deleted from the GUID catalogue. The solution for this is to keep a counter of all the LFNs that point to each GUID. However, this introduces an overhead of references bookkeeping - it is time consuming to update the references with each modification and slows down operations like directory copying. The solution implemented in AliEn is to delay the updates so that thy can be done in bulk operations.

2.2. File collections

AliEn supports file collections. A file collection is a user-defined list of entries. An entry can be either another collection, a LFN or a GUID. There is no limit in the number of entries in a collection, nor in the number of collections that can contain the same entry.

The collections look like normal files in the catalogue. The size of a collection will be the sum of the sizes of all of its components. There are commands to display, add or delete the files inside a collection. If a command is executed on a collection, it will be executed on every entry of that collection. For instance, if a user issues a command to replicate a collection to a particular SE, all the files of that collection will be replicated to that SE.

Grouping files together also has distinct advantages for the workload management. Users can specify with only one line that their jobs have to access a long list of files, thus keeping the job description short and manageable. The collection is used also for splitting the jobs into subjobs.

2.3. Triggers

The AliEn file catalogue gives the possibility to perform automatically user-defined actions when it is updated. At present, this is restricted only to modifications on LFN. Firstly, the user has to register in the catalogue a file with the action that has to be taken. This file could be for instance a shell script that sends an email. Secondly, the user has to associate the trigger with a directory in the file catalogue, and with a type of modification (insert, update or delete).

At the moment, ALICE is using the trigger mechanism to schedule file replication (copy to second storage) whenever a new file is registered in a specific directory, for example RAW data.

3. Data Management and Storage

3.1. Model

AliEn uses global and transparent file access. Every file can be accessed remotely in any storage element if necessary. To keep the file access optimal, the jobs are scheduled *close* to the storage location of the files. Files are referenced via a LFN or GUID and storage endpoints and PFNs are resolved by the AliEn file catalogue taking into account the client location. The policy is to return the *closest* replica of a requested file, unless a specific storage element is explicitly requested by the job. The closeness of storage elements is defined in a central configuration and by storage domain names.

3.2. Access Protocols

AliEn distinguishes between batch and interactive file access and scheduled file transfers. Interactive file access is done using the xrootd protocol provided by the scalla software suite [11]. xrootd protocol is well integrated into the ROOT software framework [7] used by ALICE. It offers specific optimizations for remote file access in experiment data analysis e.g.

- connection multiplexing
- vector read requests
- plugin architecture to integrate enhanced file system and security features
- fault tolerance with automatic retry mechanism, asynchronous server responses and server redirection

xrootd offers a copy client (xrdcp) and a posix interface for partial file access. ROOT has a special plugin for xrootd access (TXNetFile) to optimize file access using vector read etc.

Scheduled data transfers are handled by the AliEn FTD (File Transfer Daemon). Depending on the capabilities of the storage elements involved in the transfer, the protocol can be configured individually. If supported by the storage system, wide area transfers are done using common Grid middleware services like the gLite FTS (File Transfer Service) [6] which itself uses GLOBUS GridFTP [12] as a WAN (*Wide Area Network*) optimized transfer protocol and an SRM interface [13].

3.3. Storage Types

The data management system is currently based on four storage technologies:

- CASTOR2
- dCache
- DPM
- Scalla(xrootd)

In ALICE, CASTOR2 [14] and dCache [15] are used in Tier-0 and Tier-1 centers since they provide a tape backend. DPM [16] and Scalla are used mainly in Tier-2 centers as disk pool managers. There is an ongoing development to enable xrootd protocol on CASTOR2, dCache and DPM, which will allow applications to use only xrootd as a global protocol for data access to any storage. These three storage elements provide also a SRM interface and GridFTP protocol for scheduled WAN file transfers.

3.3.1. xrootd storage interfaces

CASTOR2 interface
Two different xrootd interfaces to CASTOR2 exists. For the first one, an additional layer of xrootd disk servers is added in front of a CASTOR2 pool. These disk servers act as a dynamic cache, which stages on demand files from CASTOR2 into the xrootd cache space. A garbage collector keeps the disks occupancy under a predefined threshold. The main advantage is the complete decoupling of xrootd and CASTOR2 services. Disadvantages are additionally needed servers and delay induced by an additional copy operation from xrootd disks to CASTOR2 disks or vice-versa.

The second option overlays an xrootd cluster on top of a CASTOR2 disk pool. In this scenario xrootd servers access directly files on the CASTOR2 disk server. No extra disk servers are needed. A special peer/manager setup allows to bypass the CASTOR2 service chain (e.g. the LSF scheduler) for the reading of files which have been once discovered through the CASTOR2 chain. While concurrent write access is slow (few seconds) - read access is very fast once a file has been staged (like in a native xrootd setup, 50 ms).

dCache interface
dCache provides a native java implementation of an older xrootd protocol version without advanced options like synchronous I/O. Clients using newer version of xrootd have to disable the read cache to get performance for file reading.

DPM interface
The DPM interface is implemented via a plugin in the redirector node and an Open File System (OFS) plugin on the disk server (see [11] for details of plugin architecture). Currently, every file read and write has to pass through DPM services. Hence the file open time for read and write is in the order of one second.

4. Workload Management

In the past years, the AliEn workload management has undergone significant improvements. The most important change has been the introduction of Job Agents (JA).

From the point of view of the user, the interface is still the same. Users describe their jobs via the Job Description Language (JDL). The submitted JDLs are collected in a central Task Queue running on the AliEn central servers. Several job optimizers look at this Task Queue, and rearrange the jobs according to their priorities and requirements. For a more detailed description see [10].

The jobs are distributed among the computing resources using a pull mode. Every site providing computing resources runs a Computing Agent (CE), which is the interface between AliEn and the local batch system or another Grid system. The CE sends a description of its capabilities to a Job Broker running on the AliEn central servers, which performs the matching with the list of jobs waiting in the Task Queue. If there is a match, the Job Broker tells the CE to submit JAs. However, at this point the job is not yet assigned to the CE, and if another CE matches the same job, the Job Broker also instructs the CE to start JAs concurrently.

When the submitted JA starts on a worker node, it executes a set of sanity checks on the environment and if successful, sends its description to the Job Broker. This description is more detailed than the description sent by the CE and it contains, among other things, the available disk space, memory, platform architecture and OS, application packages currently installed or that could be installed and its time to live. If this description satisfies the requirements of a job waiting in the Task Queue, the Job Broker assigns the job to the JA. Once the user job terminates inside the JA, the latter registers its output in the catalogue and requests another job to execute. If there are no more jobs to execute, the JA exits.

Job Agents improve the system in multiple ways. First of all, they eliminate the possibility of job failure due to problems with the local batch system or on the worker nodes themselves. Moreover, they also reduce the time between the submission and execution of a user job, since several sites can submit JA for the same job, and the fastest JA takes the job. Finally, since a JA can execute several jobs, it reduces the load on the local batch system.

At the same time the JA model increases the load on the Job Broker. In the model without JAs, only the site CE communicates with the Job Broker and multiple jobs can be assigned to the same CE simultanesouly. This is not possible with the JAs, since they are independent asynchronous processes and each of them has to communicate with the Job Broker.

The additional functionality offered by the JAs far outweights the drawbacks and in the current operation model of ALICE we have not observed an overload of the Job Broker with more than 6000 jobs running in parallel.

5. Monitoring

Monitoring in AliEn is based on the MonALISA framework [17]. The monitoring architecture consists of a hierarchical structure with selective aggregaction of the monitoring information sent upward to the next level (see Figure 2). This aggregation is a determinant factor in reducing the overall volume of information, while preserving important details.

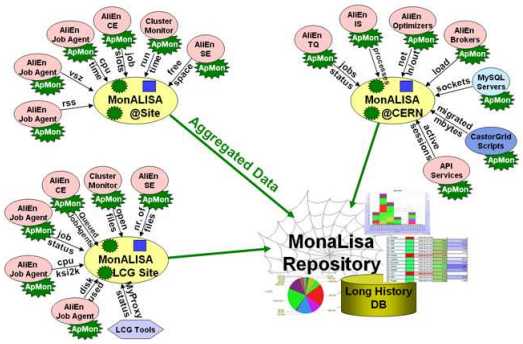

Figure 2. Information flow from the monitored entities

All AliEn services and clients are instrumented with ApMon [18], which transmits general job and host monitoring information and also allows services to send specific parameters. Extensive information is collected about CPU and memory usage, consumed and wall CPU time, open files and network traffic. Each site of the AliEn Grid has a dedicated noce (VO-Boc) where the Alien site services run. On this node, MonALISA also monitors the health of the AliEn services through periodic functional tests. Each MonALISA service has a short-time temporary buffer for highly detailed history data. It performs the filtering and aggregation of data and sends it to a central repository.

A global view of the entire AliEn Grid, as well as long term persistence of the monitoring data is assured by the MonALISA repository [19]. This service subscribes to general interest data and stores them into a PostgreSQL database. It offers both near real-time and history views, with different levels of detail - ranging from general overviews to details about individual user jobs (see Figure 3). Currently the MonALISA repository for ALICE is keeping the history for 40.000 different data series for the last 18 months. The data stream is in average 2.500 values per minute whith a current database size of 140GB ($\sim$1.4 billion data points).

In addition to presenting the information to the users and site administrators, the repository also has the task of taking automated decisions based on the monitoring information received.

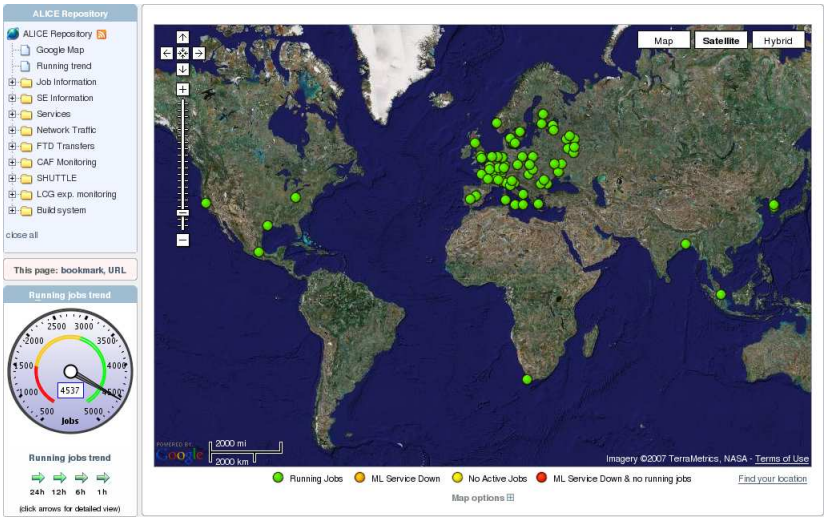

Figure 3. ALICE MonALISA Repository

The actions vary from automatic restart of the site services with email notification in case of persistent problems to automated submittion of production jobs and dynamic DNS updates to ensure proper load-balancing of the core AliEn services.

6. Interoperability

One of the major new development efforts in AliEn are the interfaces to other Grid flavours. At present, these interfaces allow AliEn to interoperate with the gLite, ARC and OSG Grids. The interfaces can be deployed at different hierarchy levels. For instance, it is possible to make a whole Grid behave as a single CE through a gateway deployed by the Grid service provider. The interface with ARC is geared towards this model. Another approach is to associate each computing element from a Grid with an AliEn computing element. This is used in the gLite interface.

There is also a possibility to interface any number of Virtual Organizations running AliEn to each other. This interface makes a whole AliEn VO appear as a single computing element for another VO. The interface uses fully the Job Agent model, where one VO is submitting JAs to the resources of another VO. The interaction between the VOs in such 'Grid of Grids' is illustrated in Figure 4.

7. Future work

The future development path of AliEn is summarized below:

- Improvements of service scalability and failover capability. At the moment of writing this article, AliEn handles more than six thousands concurrent jobs without its services showing signs of overload. However during the normal operation of the ALICE experiment, the services availability should be above 99 percent and the number of simultaneous jobs will increase to approximately 20.000 within the next three years.
- Resource bank allowing a finer control on job priorities through job pricing.
- Virtualization of services and applications.

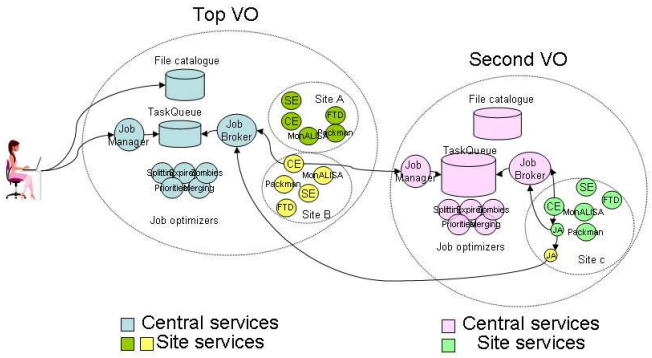

Figure 4. Multi AliEn-VO interface

8. Conclusions

AliEn is high level middleware adopted by the ALICE experiment at CERN. It has been used for massive Monte Carlo event production since the end of 2001 and for user data analysis since 2005. Its capabilities are well suited to the requirements of the ALICE computing model for data management and processing. Its components have been validated under heavy load in a series of data challenges of increasing complexity. In addition to the modules needed to build a fully functional Grid, AliEn provides interfaces to various Grid flavours, thus enabling a true Grid interoperability, while hiding the underlying complexity from the end user.

In the coming years, the development of AliEn will continue along the architectural path chosen at its initial conception. An ever increasing array of capabilities will be added, allowing ALICE to fully harness the distributed computing resources needed to manage the multi-PB data collected by the experiment.

Acknowledgments

This work was partially funded by EGEE. EGEE is a project funded by the European Union under contract INFSO-RI-031688

References

[1] Saiz, P. et al., AliEn - ALICE environment on the Grid, Nucl. Instrum. Meth., A502 (2003) 437-440.
[2] The ALICE collaboration, http://aliceinfo.cern.ch
[3] ARC, the NorduGrid middleware, http://www.nordugrid.org/middleware/
[4] Bagnasco, S. et al., AliEn - EDG Interoperability in ALICE, ECONF C0303241 (2003) TUCP005
[5] OSG, http://www.opensciencegrid.org/
[6] gLite - Lightweight Middleware for Grid Computing, `http://glite.web.cern.ch/glite`
[7] ROOT - An Object-Oriented Data Analysis Framework, `http://root.cern.ch`
[8] PANDA experiment, htpp://gsi.de/panda
[9] Magic5, http://www.magic5.unile.it/
[10] Saiz, Pablo and Buncic, Predrag and Peters, Andreas J., AliEn Resource Brokers , ECONF C0303241 (2003) TUAP002
[11] The Scalla Software Suite, `http://xrootd.slac.stanford.edu`
[12] GridFTP, `http://www.globus.org/grid_software/data/gridftp.php`
[13] The Storage Resource Manager Interface Specification, `http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html`
[14] CASTOR - CERN Advanced STORage manager, `http://castor.web.cern.ch/castor/`
[15] dCache, `http://www.dcache.org/`
[16] Disk Pool Manager, `http://www.gridpp.ac.uk/wiki/Disk_Pool_Manager`
[17] MonALISA: An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications, I.C.Legrand, H.B.Newman, R.Voicu, C.Cirstoiu, C.Grigoras, M.Toarta, C. Dobre, CHEP 2004, Interlaken, Switzerland

[18] ApMon library, http://monalisa.caltech.edu/monalisa__Download__ApMon.html
[19] MonALISA repository for ALICE, http://alimonitor.cern.ch/