PROCEEDINGS OF SCIENCE

Optimization of Grid Resources Utilization: QoS-aware client to storage connection in AliEn

Grigoras Costin[1]
CERN
1211, Geneva, Switzerland
E-mail: `costin.grigoras@cern.ch`

Betev Latchezar
CERN
1211, Geneva, Switzerland
E-mail: `latchezar.betev@cern.ch`

Saiz Pablo
CERN
1211, Geneva, Switzerland
E-mail: `pablo.saiz@cern.ch`

Schreiner Steffen
CERN
1211, Geneva, Switzerland
E-mail: `steffen.schreiner@cern.ch`

In a World Wide distributed computing system like the ALICE Environment (AliEn) Grid Services, the closeness of the data to the actual computational infrastructure denotes a substantial difference in terms of resources utilization efficiency. Applications unaware of the locality of the data or the status of the storage environment can waste network bandwidth or fail to access data from remote or nonoperational storage elements. In this paper we present an approach to QoS-aware client to storage connection by introduction of a periodically updated Storage Element Rank Cache. Based on the MonALISA monitoring framework, a Resource Discovery Broker is continuously assessing the status of all available Storage Elements in the AliEn Grid. Combining availability with network topology information, rated lists of Storage Elements are offered to the client requesting access to remote data. The lists are centrally cached by AliEn, filtered in the course of user-based authorization and requested QoS flags. This approach to storage access shows significant improvements towards an optimized storage and network resource utilization and enhances the client resilience in case of failures.

13th International Workshop on Advanced Computing and Analysis Techniques in Physics Research - ACAT 2010
Jaipur, India
February 22–27 2010

[1] Speaker

1. ALICE Computing Environment

AliEn (ALICE Environment) [1] is a lightweight Open Source Grid Framework built around other Open Source components using the combination of a Web Service and a Distributed Agent Model. It was developed within the ALICE Offline Project at CERN and constitutes the production environment for simulation, reconstruction and analysis of physics data of the ALICE Experiment.

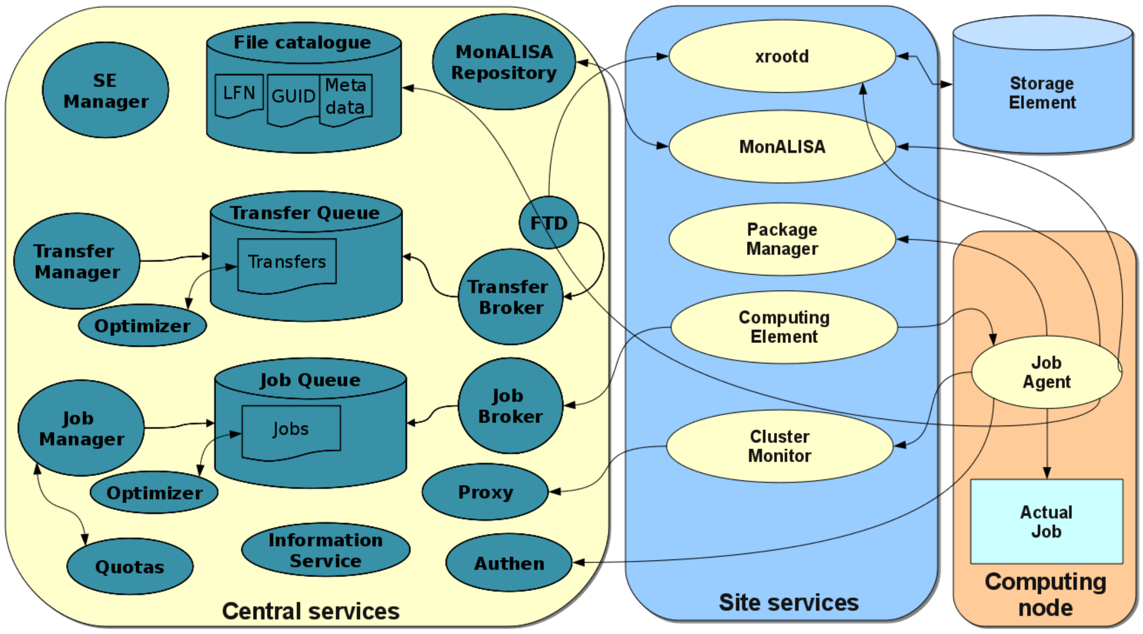


Figure 1- Structure of the AliEn services

AliEn has two types of services, some that run at each site and others that run centrally, as shown in Figure 1. The central services maintain the file catalog, jobs and transfer queues and offer authentication, authorization and accounting services. They also provide a uniform Grid access to the users. Each site runs a small set of services on a dedicated local server (called VoBox). These services are responsible for sending pilot jobs when jobs matching site resources are waiting in the queue, installing the required software packages and mediating jobs' interaction with the central services while they are running and monitoring the usage of site resources.

Most of the services can be run in load-balancing mode, both centrally and on the sites, with several instances of the same service running on separate servers.

As of the writing of this paper there are 86 sites and 110 VoBoxes on 5 continents providing access to more than 23,000 CPU cores, 9PB of disk storage in 47 storage elements and another 9 tape-backed storage elements for data archiving.

The ALICE Computing Model requires that jobs are executed close to the data location thus spreading the data on all available storage elements is important in order to optimize the use of the available computing power.

2. Monitoring infrastructure

The ALICE Computing Model requires that jobs are executed close to the data location thus spreading the data on all available storage elements is important in order to optimize the use of the available computing power.

MonALISA [2] is a Java-based monitoring and control framework. It features automatic services discovery and a large set of monitoring modules. New modules can be dynamically deployed as well as filters to aggregate the collected parameters in order to create overviews. They also implement secure communication channels so that agents running inside the services can communicate between themselves. The services expose the monitoring information to clients that can browse it and dynamically subscribe to sub-trees of the monitoring parameters.

While each service can keep the monitoring history in a database, a dedicated MonALISA component called *Repository* combines the client function of subscribing to parameters with database storing and an HTTP interface for data presentation.

The deployment of MonALISA closely follows the AliEn model, with one MonALISA service running on each VoBox along with the site services and a central repository collecting and displaying site-wide parameters.

One of the monitoring modules extensively used in AliEn is *ApMon* (Application Monitoring) [3], which converts UDP packets in the XRD open format into MonALISA objects. Many versions of sender libraries are available for the popular programming languages, all offering both basic host monitoring and application-specific monitoring.

AliEn services use the Perl implementation of ApMon to send monitoring information to the closest MonALISA service. One of the services that reports to the site local MonALISA instance is the *Job Agent* (AliEn name of the job pilot) that enables local machine monitoring and tracking of the resources used by the current job.

Other services like xrootd [4] use the C/C++ version of the library to report the machine parameters (network traffic, load, CPU usage, disk IO and so on) and other parameters like the available and used disk space on the storage.

The ALICE repository [5] subscribes mostly to site-wide overviews (like the total amount of CPU time consumed by the jobs running at each site, number of jobs in each state and so on) and to particularly interesting parameters. From the more than 2,700,000 parameters collected by MonALISA the repository only subscribes to about 80,000 of them that are updated with an average frequency of 400Hz.

Some of these parameters can be further aggregated at the virtual organization level and can be discarded afterwards or saved for more detailed history views.

3. Topology discovery

Another module that runs in each MonALISA site instance performs *tracepath* commands between pairs of sites and publishes all intermediary nodes and the Round Trip Time (RTT) to each of them. Other agents perform bandwidth tests between pairs of sites and publish the available bandwidth. The results of these tests are collected by the repository and later associated with the network paths between the respective sites.

Derived information is computed in the repository, associating the Autonomous System (AS) number to each of the nodes in a network path.

4. Storage discovery and testing

The repository can also run other monitoring modules that provide global values and one of them periodically queries AliEn for the list of defined storage elements and their size and usage according to the file catalog. Then periodic functional tests are performed from the central machine to check whether the basic file operations (add, get, remove) are successful. The entire software and network stacks are checked through these tests, thus the outcome should be identical for all clients trying to access the respective storages. Currently the functional tests are performed every two hours for a given storage element.

5. Ranking algorithm

Aggregating the monitoring and test results, a client-to-storage distance metric can be computed and used to sort the list of available storage elements to a particular client. Then the closest working storage elements can be selected either to save the data or, in case of reading, sorting the available locations based on this metric, trying to read from the closest location.

The algorithm associates to each storage element a list of IP addresses representing known machines from its vicinity (site local VoBox IP address, all known xrootd nodes' IP addresses if the storage runs xrootd). Then the distance between the client IP address and the given storage is computed as the minimum value of the distance between the client IP address and each of the storage IP addresses. Figure 2 below is a partial topology that illustrates the steps that the algorithm takes to calculate the metric between a starting point (in this case *Catania*) and the sites providing storage.

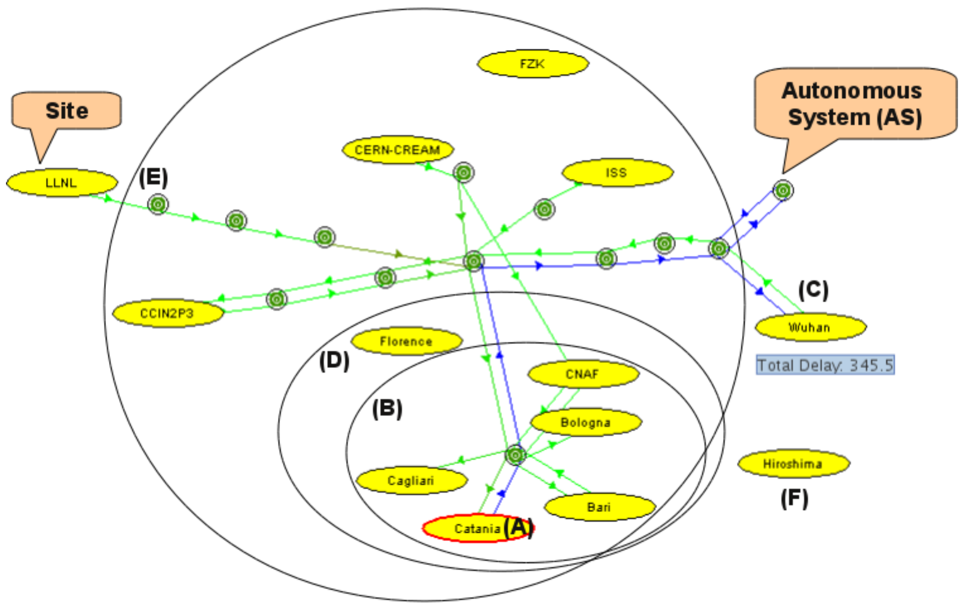


Figure 2- The metric representing the distance between one site (Catania) and all the others. Letters represent different cases in the algorithm below.

The distance between two arbitrary addresses increases with the following until one condition is met:

- the two belong to the same IP C-class (falls in case A in the picture above);
- they have the same domain name (also case A);
- they belong to the same Autonomous System [6] (case B);
- the RTT between the two respective C-classes is known, then it's a function of its value (case C);
- the average RTT between the two respective Autonomous Systems is know, then it's a function of its value (similar to case C but for a partial path);
- they are in the same country (case D);
- they are on the same continent (case E);
- lastly a large value meaning “far far away” (case F).

6. Feeding back monitoring in AliEn

The repository implements a simple web service that given an IP address returns the list of storage elements sorted according to the closeness to the requested address. From the set of AliEn storage elements only the ones for which the functional tests were successful are kept. The distance function described above is used to sort them together with the available disk space and a random coefficient to enhance the data distribution. The available disk space, taken from xrootd reporting or AliEn catalogue, influences the sorting order by giving a slight preference for large storages and removing the ones approaching full capacity. The random factor (Gaussian “normal”) helps to spread the data uniformly across all storage elements, otherwise there is a natural ‘clustering’ of the data close to the large sites.

Usually the clients are jobs running in a computing centre and writing the output files (several replicas) to storages providing a given quality of service (such as “disk”, “tape” or any other tag). In this case instead of the actual IP address of the job execution host we can use the IP address of the site VoBox. The reason for this choice is that the information can be more easily cached in the central database so that it is directly used in more complex queries preparing the information for the client. A Resource Discovery Broken periodically queries the MonALISA web service for the sorted list of storage elements in reference to each known AliEn site.

In the case of real users it can either be assumed they are close to a given site, which is usually the case as they work at one of the institutes providing computing resources, or the client machine's IP address can be used for a dynamic discovery. Currently the first option is used and another service maps the client's IP address to one of the sites but the second option is also available and could be used in the future.

The clients implement a failover mechanism to compensate for unavailability of the storage services. In those cases the client asks again the central services for a list of storage elements providing the list of names that have already been tried, either because the operation was successful or has failed. This information feedback functionality is also used for passing specific user requirements like forced storage elements or explicitly excluded ones. The central services will return the best storage elements according to the feedback and the client will keep trying as long as there is a valid reply of storage elements or it has saved in all the required storage elements. If the job doesn't manage to create the required number of replicas and/or the explicit user requirements are not met it will be put in a special state (DONE_WARN), giving the possibility to replicate the data at a later time, satisfying the initial requirements.

7. Discovery process in action

Storage discovery is very important for the jobs, and here is an example of how everything is put together. The job description has the following output requirement:
root_archive.zip:galice.root,...,TrackRefs.root,AliESDs.root,Run.root@disk=3*
meaning that the Zip archive of some ROOT files produced by the job should be uploaded to the closest 3 storage elements of type "disk".

This is a MonteCarlo production job that splits in 1000 subjobs, each of them being able to run on any site. The output is uploaded in different locations, depending on where the job ran and the status of the storage elements when it finished. Below are trace log extracts from a few subjobs of this production.

Sep 30 09:37:12 2010 – job running at FZK
Uploading file based on Storage Discovery, requesting QoS=disk, count=3.
We got an envelope for a discovered SE: **ALICE::FZK::SE**
We got an envelope for a discovered SE: ALICE::GSI::SE
We got an envelope for a discovered SE: ALICE::Prague::SE

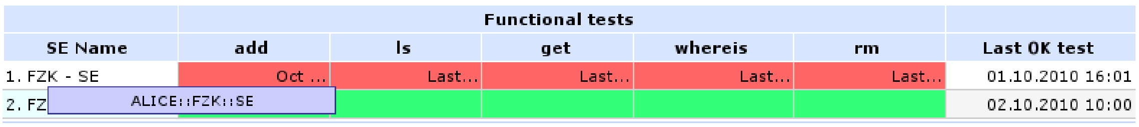
Oct 1 17:05:39 2010 – job running at GSI
We got an envelope for a discovered SE: ALICE::GSI::SE
We got an envelope for a discovered SE: **ALICE::FZK::SE**
We got an envelope for a discovered SE: ALICE::ISS::FILE
Adding the file file://lxb639.gsi.de/.../root_archive.zip to ALICE::GSI::SE
Adding the file file://lxb639.gsi.de/.../root_archive.zip to **ALICE::FZK::SE**
ERROR storing file://lxb639.gsi.de/.../root_archive.zip in ALICE::FZK::SE
Adding the file file://lxb639.gsi.de/.../root_archive.zip to ALICE::ISS::FILE
We got an envelope for a discovered SE: ALICE::Prague::SE
Adding the file file://lxb639.gsi.de/.../root_archive.zip to ALICE::Prague::SE
Successfully stored the file /.../root_archive.zip on 3 SEs.

Oct 1 23:23:16 2010 – job running at FZK, while the local storage element is down
We got an envelope for a discovered SE: ALICE::GSI::SE
We got an envelope for a discovered SE: ALICE::Prague::SE
We got an envelope for a discovered SE: ALICE::CNAF::SE

Sep 30 23:32:06 2010 - job running at Troitsk
We got an envelope for a discovered SE: ALICE::Troitsk::SE
We got an envelope for a discovered SE: ALICE::RRC-KI::SE
We got an envelope for a discovered SE: ALICE::JINR::SE

The first case shows the activity of one job running at the ALICE Tier1 site in Germany, FZK. It has discovered the site local storage element, another storage element in Germany (at ALICE Tier2 site GSI) and a third closest one in Prague. Everything worked fine and the job requirements were met.

In the second case the file could not be uploaded to the FZK storage element, so the discovery mechanism returned a fourth option (Prague) where the upload was possible and in the end the requirements were met and the job finished successfully. The bolded log lines show where the error was detected and when another option was given to the client. At this moment the monitoring was not aware yet of the problems at FZK, so it was still being considered valid option. At a later time (third case) the functional tests were already showing the problem at FZK so it was excluded from the list of options, even for jobs running on the same site. To compensate, GSI and Prague were promoted and a third best storage element – CNAF, ALICE Tier1 in Italy – was presented to the client. So a pointless attempt to upload to a broken storage element was avoided. Figure 3 below shows the status of functional tests in the monitoring pages at a later time when the problem was still not solved.



	Functional tests					
SE Name	add	ls	get	whereis	rm	Last OK test
1. FZK - SE	Oct ...	Last...	Last...	Last...	Last...	01.10.2010 16:01
2. FZ ALICE::FZK::SE	🟩	🟩	🟩	🟩	🟩	02.10.2010 10:00

Figure 3 – Status of the functional tests in the monitoring pages, with the FZK storage element showing operational problems leading to it being excluded from discovery

Last example shows a job of the same type running at the Russian ALICE site Troitsk. All storage elements that were returned to the client are from Russia as well, keeping the data close to where the job ran.

Before having this discovery mechanism the only option was to specify a fixed list of storage elements for writing. This list was manually entered in the job description, based on the status at the start of a production cycle. The production would then stress these few elements, often at far away locations relative to where the jobs ran, and in some cases even fail to write if the storages developed any problems while running the production. This was particularly visible in the case of job log files that are typically stored in one location only, and so one failing storage element halted the entire production.

Another advantage of the automatic discovery is the load balancing of the resources. With such a blanket production all storage elements contribute to the traffic. Moreover, because the produced data is spread in the entire Grid, analysis jobs for this data can then run practically everywhere because some chunks of data can be found in every storage element, and in volumes correlated with how many production jobs were executed there, so the distribution of analysis jobs will practically follow the distribution of resources among sites.

8. Conclusion

This paper has presented how the storage elements and network monitoring information is fed back into the system to obtain a maintenance-free and highly optimized system for data access. From user perspective, the only parameter required is the QoS tag, specifying the type of storage and number of replicas, while the location and status of the storage elements themselves are completely opaque. New storage resources are automatically discovered and configured and the access is optimized depending on where the code is running. The QoS-aware method has increased substantially the efficiency of all job types, since by design, the failures for writing are only possible if all storage elements are inaccessible, and in the case of data reading, the client will only run close to a working storage element.

9. References

[1] : Bagnasco S, Betev L, Buncic P, Carminati F, Cirstoiu C, Grigoras C, Hayrapetyan A, Harutyunyan A, Peters A J and Saiz P 2007 *AliEn : ALICE environment on the GRID*, *J. Phys.: Conf. Ser.* **119**

[2] : Legrand I, Newman H, Voicu R, Cirstoiu C, Grigoras C, Toarta M and Dobre C 2004 *MonALISA : An Agent Based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications*, *Computing in High Energy Physics and Nuclear Physics 2004, Interlaken, Switzerland* 907

[3] : Cirstoiu C, Legrand I, Voicu R and Stratan C 2006 *ApMon* http://monalisa.caltech.edu/monalisa__Download__ApMon.html

[4] : *The Scalla/XrootD Software Suite. July2009* http://savannah.cern.ch/projects/xrootd/, http://xrootd.slac.standford.edu/

[5] : *MonALISA Repository for ALICE*, 2006 http://alimonitor.cern.ch/

[6] : *IETF RFC 1930*, http://tools.ietf.org/html/rfc1930