Cooperation among ALICE Storage Elements: current status and directions (The ALICE Global Redirector: a step towards real storage robustness).

Fabrizio Furano[1]
CERN
Geneve 23, CH-1211, Switzerland
E-mail: `fabrizio.furano@cern.ch`

In this talk we address the way the ALICE Offline Computing is starting to exploit the possibilities given by the Scalla/Xrootd repository globalization tools. These tools are quite general and can be adapted to many situations, without disrupting existing designs, but adding a level of coordination among xrootd-based storage clusters, and the ability to interact between them.

XII Advanced Computing and Analysis Techniques in Physics Research
Erice, Italy
3-7 November, 20

[1] Speaker

1.Introduction

In the ALICE computing model, the data access is performed through the so-called "Xrootd protocol", actually allowing sites to choose the preferred data access platform to deploy, among the ones supporting this data access protocol. [1]

A matter of fact is that the Scalla/Xrootd framework is getting a large success in the various deployments, thus making possible to think about how to make a coherent distributed storage of very large size evolve form the technological point of view, in order to provide a better service for the current and future times. Although the motivation for this kind of evolution could be considered largely academic, unfortunately this is not true. The reason for that is the obvious wait for the massive access to the ALICE data repository, but, on the other hand, generally there is a diffused concern about the evolution of the computing farms in the next years. This evolution might raise by a factor the data access rate for the whole repository, potentially turning the data access part into a serious bottleneck. In this environment, we started considering the fact that we believe that the typical Wide Area Network connections between sites could be used more efficiently, both for data access (eventually partially, as we will discuss) and for data movement across sites.

This major technological effort must be viewed as a serious attempt to prepare the data access technology in use for the times to come, in order to reduce the risk associated to the (largely expected) increase in the expectations dealing with quality, robustness and aggregate data throughput for the whole repository.

2.The starting point

In order to better understand the work done and the chosen design directions, we believe that it is useful to summarize the characteristics of the Scalla/Xrootd platform, some of which were considered as a starting point to decide the direction to exploit.

One point of view that sometimes, in our experience, is not strongly enough considered, is the overall complexity of most software tools and frameworks used in HEP computing, when it comes to considering the parts constituting them. From this perspective, the commonly used statement about the Scalla/Xrootd platform is "*no weird configuration requirements*"; the explanation of this is that, by design, the whole platform considers concepts like simplicity and self-contention as a major requirement, thus reducing drastically the number of dependencies from external packages. For this reason, a better explanation for the said statement could be that the complexity of a deployment typically "scales linearly" with the complexity of the requirements for that deployment. Moreover, the common deployment choices have been fully encapsulated in to the setup procedure [6], drastically reducing the need for subtle and complex setup/maintenance tasks. To this aspects, we must add the fact that, starting from the default configuration, the system is highly customizable, if needed, eventually exploiting the possibilities given by its internal configuration files or its modular architecture, fully based on software plugins.

One more very important feature, which we believe it's fundamental for the kind of usage that we are going to describe, especially when dealing with Wide Area Network connections is the set of fault tolerance-related procedures described in the Xrootd protocol [1][5] and implemented in its client side, namely *XrdClient*. [4]

Other interesting characteristics of the Scalla/Xrootd platform for data access are related to performance and scalability. One of the first aims of the project was to make it possible to scale the system theoretically in an indefinite way at the server side [2]. The basic and very simplified idea behind the architectural details is that, doubling the hardware resources should be an always-valid way to double the data throughput and/or the site of the repository. This is an interesting feature especially when it comes to thinking about repositories that are distributed among different sites, and its practicability is augmented by the very reduced (if not inexistent) set of external software dependencies. The reason for that is that a system that is not very scalable is not, in principle, a very good candidate for bigger and bigger deployments where the overall quality of service is not an option.

The fact that the whole platform does not make use of databases (e.g. to translate file names or to keep a catalogue of them), but instead exploits concepts which are closer to the peer-to-peer agents world [2], constitutes also the technological basis to another set of characteristics, all of them useful when it comes to distributing a large repository across different sites. In this case we are referring to the fact that opening a file towards an xrootd server has an additional latency which is very low with respect to the one given by the hardware where the server and the client run, here comprehending also the network between them [3] [7].

Moreover, various multiplexing techniques on the TCP connections and the file descriptors help in keeping low the resource usage at the server side, thus allowing more clients to connect to the same server, since the only present hard limit is the number of sockets and file descriptors in the machine.

The last topic about the characteristics of the Scalla/Xrootd system which can be useful to build new efficient ways to handle very large data repositories distributed through multiple sites is related to the fact that all the system can work with Wide Area Networks with a good level of efficiency, [8] because:

- The client-server connections allow an efficient remote POSIX-like direct data access (in read and write mode) through Wide Area Networks, by making use, when needed, of:
    - TCP multistreaming techniques, in order to be able to maximize the data throughput in high latency networks;
    - Latency-hiding mechanisms [8], which can play a major role in highering the data throughput through high latency networks, also for sparse data access patterns;
    - The ROOT [9][10] data analysis framework, used extensively in the ALICE software, is able to correctly trigger the latency-hiding mechanisms, because it is able to produce, in most cases, a list of pairs (offset, length) describing the future data access pattern, suitable for feeding the algorithms behind those latency-hiding mechanisms.

- The server clusters can be composed by servers located in different sites, or one can create a meta-cluster by aggregating several clusters, thus giving a unique entry point for their content.

3.The Virtual Mass Storage System

The set of Wide Area Network related features of the Scalla/Xrootd system, as described, plays a major role in giving ways to make a storage system evolve in the direction of being able to exploit Wide Area Networks in a productive way. The major design challenge is how to exploit them in order to augment the overall robustness of a large distributed deployment.
Nevertheless, in this perspective we must consider the fact that the ALICE computing model [11][12] refers to a production environment, which is used heavily and continuously. Any radical change, hence, was not to be considered a good idea (given also the high complexity of the whole ALICE software infrastructure).
Moreover, the idea of accessing remote data from data analysis jobs was not new in the ALICE computing framework. For instance, this is the default way that the analysis jobs have to access the so-called "conditions data", which are stored in a storage element at CERN. Even if each analysis job accesses 100-150 conditions data files (and 10,000 concurrent jobs are considered a normal activity), historically that choice always proved itself a very good one, with a good performance for that task, and a negligible failure rate.
On the other hand, the ALIEN software [11], together with its central services, already gives some sort of unique view of the entire repository, by faking the namespace coherence across different sites by mapping the file names using a central relational database. Its usage patterns also confirms the fact that it would be desirable to move towards a model where the storage coherence is given by the storage system itself and does not need computationally very heavy namespace translations.
One more very important consideration comes also from the fact that in the ALICE computing model any storage cluster can be populated only in two ways:

- Writing directly to it from an external machine, where a software takes responsibility of the transfer;
- Writing to the backend store, if present (e.g. a tape system behind the disk pools, which then act as a cache).

For instance, in the ALICE computing model these tasks are accomplished by a daemon called FTD (File Transfer Daemon), which runs in every site. Although functional and relatively performant, there are some issues which are inherent in this kind of design, and which we can shortly summarize as follows:

- All the data transfer load hits the same machine where FTD runs for the site
    - ➢ Several concurrent transfers can consume a high amount of system resources;
    - ➢ All the external traffic of the site goes through one machine, which by construction is in the site which is the destination of the data move;

- If a file is missing or lost for any reason (e.g. broken disks, errors in the central catalog, etc…),
    - All the processing jobs scheduled to run on that file instance will inevitably fail, and the trouble can be solved only with a careful human intervention.
    - With an order of 10^7 online files per site, finding the source of such a file miss can be extremely problematic.

An additional consideration is that, from a functional point of view, we considered as a poor solution the fact that, once a missing file is detected, nothing is done by the storage system to fetch it. This is especially controversial when, for instance, the file that is missing in a site is available in a well-connected one, and pulling it could be done without interrupting the requesting job.
A careful consideration of all these aspects led to thinking that there was a way to redesign the ALICE Scalla/Xrootd-based distributed storage in a way which automatically tries to fix this kind of problems, and at the same time does not disrupt the existing design, allowing for an incremental evolution of the storage, distributed across many almost independent sites contributing to the project. One more requirement was to design it in the most generic way, possibly agnostic with respect to the computing environment where it's placed.

The solution to the problem has been to apply the following statements to the overall storage:
- All of the xrootd-based ALICE storage elements are aggregated into a unique meta-cluster, which exposes a unique and namespace-coherent view of their content;
- Each site that does not have tapes or similar mass storage systems considers the meta-cluster (except itself) as its mass storage system. If an authorised request for a missing file comes to a storage cluster, then this tries to fetch it from one of the neighbours.

The host which manages the meta-cluster (which in the Scalla/Xrootd terminology is called a meta-manager) has been called *ALICE Global redirector*, and the idea for which a site considers the global namespace (to which it participates with its content) as its mass storage has been called *Virtual Mass Storage System (VMSS)*.

In this context, we consider as very positive statements the facts that:
- In the xrootd architecture, the process of location of a file is very fast (if the location is unknown to a manager server, finding it takes approximately only a network round-trip time plus one local file system lookup);
- No central repositories of data or metadata are needed for the mechanism to work, everything is performed through real-time message exchanges;
- All of the features of the Scalla/Xrootd suite are preserved (e.g. scalability, performance, etc…);
- The design is completely backward compatible, e.g. in the ALICE case no changes were required to the complex softwares which constitute the computing infrastructure;

- The system, as designed, represents a way to deploy and exercise a global real-time view of a unique multi-petabyte distributed repository in an incremental way. This means that the system is able to give a better service as more and more sites upgrade their storage clusters and join in.

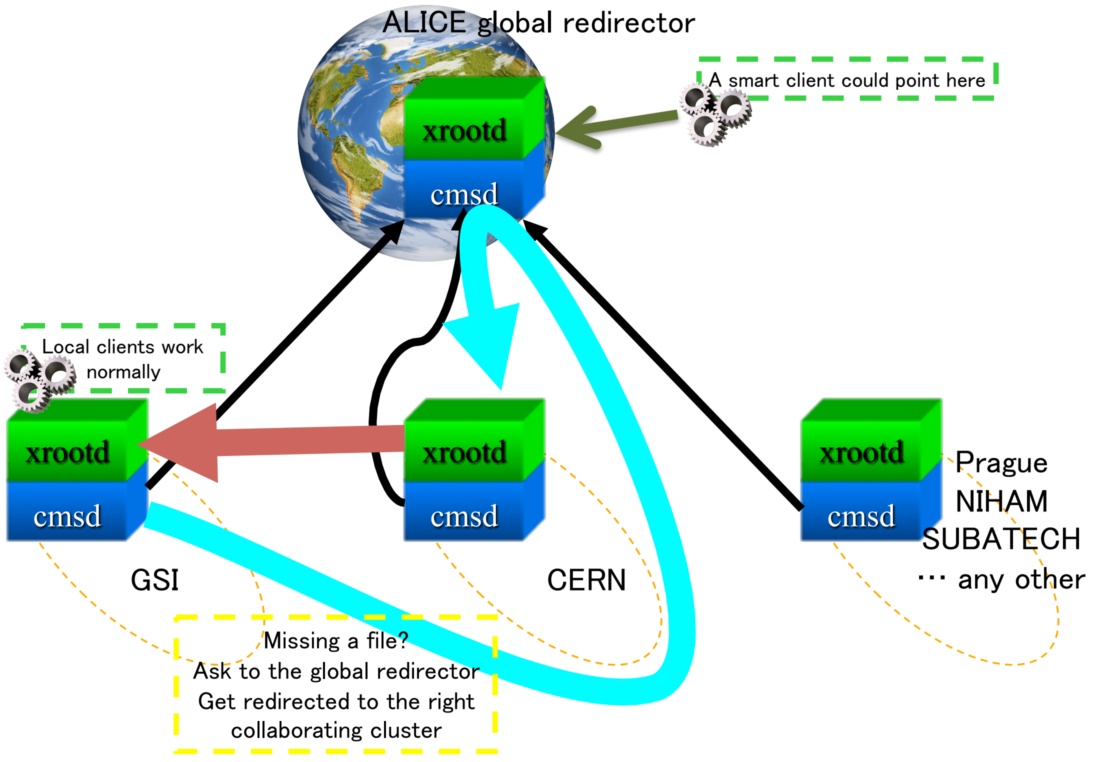
ALICE global redirector
xrootd
cmsd
A smart client could point here
Local clients work normally
xrootd
cmsd
GSI
xrootd
cmsd
CERN
xrootd
cmsd
Prague
NIHAM
SUBATECH
… any other
Missing a file?
Ask to the global redirector
Get redirected to the right collaborating cluster

Figure 1: An exemplification of the Virtual Mass Storage System

Figure 1 shows a generic small schema of the discussed architecture. In the lower part of the figure we can see three cubes, representing sites with a Scalla/Xrootd storage cluster. If a client instantiated by an analysis job (on the leftmost site) does not find the file it needs to open, then the GSI cluster asks the global redirector to get the file, just requesting to copy from it. This copy request (which is just another normal Xrootd client) is then redirected to the site that has the needed file, which is copied immediately to the leftmost site. The previously paused job can then continue its processing like if nothing particular had happened.

4.Conclusion

As very often happens, the described effort is an ongoing task on a system in production, composed by more than 60 sites all over the world. Hence, the overall activity was aimed at giving to the whole storage system a way to evolve technologically, without disrupting a design that is known to work reasonably well from the functional point of view.

One of the most interesting features of the new design is that the new storage clusters are more autonomous, and are able to initiate a "file pull" in a few milliseconds, which is a couple of orders of magnitude faster than the current system, based on a centralised scheduling of the data transfers to perform, and on database lookups to check for new transfers to initiate.

Even if this new mechanism is proving itself very good in fetching automatically the missing files in a repository, there are some aspects that are even more interesting from the technological and functional points of view.

One is that in the near future, this highly robust and performant way to move files could be used as a transport mechanism, always coordinated by FTD in the ALICE computing (or any other system with similar functionalities), but demanding the transfers to the storage cluster, in order to get better performance and stability.

Another aspect is that, even if right now it is treated as a secondary feature, the *ALICE global redirector* is just a plain xrootd server able to give robust, real-time access to a fully distributed repository through Wide Area Network. Hence, if an application is able to correctly trigger the latency-hiding mechanisms of the Scalla/Xrootd client, by pointing it to the global redirector it can run with a good performance even through WAN, since each file open it performs will be redirected to the proper site, and the latencies of the WAN will be almost completely cancelled, provided that the network can give a sufficient throughput.

At this time we are not proposing or foreseeing to leave the well-tested design for which an analysis job preferably accesses data locally to the site where it runs; instead, we believe that this kind of feature could be one more step in the direction of allowing a physicist willing to analyse some data interactively to perform quickly and efficiently his task without having to schedule an analysis job.

References

[1] *The Scalla/xrootd Software Suite*, http://savannah.cern.ch/projects/xrootd and http://xrootd.slac.stanford.edu/

[2] Fabrizio Furano, Andrew Hanushevsky, "Managing commitments in a Multi Agent System using Passive Bids," iat,pp.698-701, 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'05), 2005

[3] Hanushevsky, A. and Weeks, B. 2005. Designing high performance data access systems: invited talk abstract. In Proceedings of the 5th international Workshop on Software and Performance (Palma, Illes Balears, Spain, July 12 - 14, 2005). WOSP '05. ACM, New York, NY, 267-267. DOI= http://doi.acm.org/10.1145/1071021.1071053

[4] Dorigo, A., Elmer, P., Furano, F., and Hanushevsky, A. 2005. XROOTD/TXNetFile: a highly scalable architecture for data access in the ROOT environment. In Proceedings of the 4th WSEAS international Conference on Telecommunications and informatics (Prague, Czech Republic, March 13 - 15, 2005). M. Husak and N. Mastorakis, Eds. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, 1-6.

[5] A. Dorigo, P. Elmer, F. Furano, and A. Hanushevsky, Xrootd - A highly scalable architecture for data access, WSEAS Transactions on Computers, Apr. 2005.

[6] XRootd explained, Computing seminar at CERN, http://indico.cern.ch/conferenceDisplay.py?confId=38458

[7] Hanushevsky, A. Are SE architectures ready for LHC? In Proceedings of ACAT 2008: XII International Workshop on Advanced Computing and Analysis Techniques in Physics Research, http://acat2008.cern.ch/, http://indico.cern.ch/contributionDisplay.py?contribId=227&sessionId=23&confId=34666

[8] F Furano and A Hanushevsky, Data access performance through parallelization and vectored access. Some results. CHEP07: Computing for High Energy Physics. Journal of Physics: Conference Series 119 Volume 119 (2008) 072016 (9pp)

[9] ROOT: *An Object-Oriented Data Analysis Framework*, http://root.cern.ch

[10] *Distributed Parallel Analysis Framework with PROOF*, M. Ballintijn, R. Brun, F. Rademakers and G. Roland, http://root.cern.ch/twiki/bin/view/ROOT/PROOF .

[11] A L I C E: Technical Design Report of the Computing, June 2005. ISBN 92-9083-247-9, http://aliceinfo.cern.ch/Collaboration/Documents/TDR/Computing.html

[12] L. Betev, F. Carminati, F. Furano, C. Grigoras, P. Saiz. The ALICE computing model: an overview. Third International Conference "Distributed Computing and Grid-technologies in Science and Education", GRID2008, http://grid2008.jinr.ru/

[13] *Authorization of Data Access in Distributed Storage Systems*, D. Feichtinger, A.J. Peters, The 6th IEEE/ACM International Workshop on Grid Computing, 2005, http://ieeexplore.ieee.org/iel5/10354/32950/01542739.pdf?arnumber=1542739