Monitoring of large-scale federated data storage: XRootD and beyond.

J Andreeva[1], A Beche[1], S Belov[2], D Diguez Arias[1] , D Giordano[1], D Oleynik[2], A Petrosyan[2], P Saiz[1], M Tadel[3], D Tuckett[1] and I Vukotic[4]

[1] CERN (European Organization for Nuclear Research)
[2] JINR (Joint Institute for Nuclear Research)
[3] UCSD (University of California, San Diego)
[4] U. Chicago (University of Chicago)

Abstract. The computing models of the LHC experiments are gradually moving from hierarchical data models with centrally managed data pre-placement towards federated storage which provides seamless access to data files independently of their location and dramatically improve recovery due to fail-over mechanisms. Construction of the data federations and understanding the impact of the new approach to data management on user analysis requires complete and detailed monitoring. Monitoring functionality should cover the status of all components of the federated storage, measuring data traffic and data access performance, as well as being able to detect any kind of inefficiencies and to provide hints for resource optimization and effective data distribution policy. Data mining of the collected monitoring data provides a deep insight into new usage patterns. In the WLCG context, there are several federations currently based on the XRootD technology. This paper will focus on monitoring for the ATLAS and CMS XRootD federations implemented in the Experiment Dashboard monitoring framework. Both federations consist of many dozens of sites accessed by many hundreds of clients and they continue to grow in size. Handling of the monitoring flow generated by these systems has to be well optimized in order to achieve the required performance. Furthermore, this paper demonstrates the XRootD monitoring architecture is sufficiently generic to be easily adapted for other technologies, such as HTTP/WebDAV dynamic federations.

1. Introduction

The computing models of the LHC experiments are gradually moving toward federated storage such as XRootD [1]. A federation can be seen as an aggregation of storage of any kind into a global namespace using a single access protocol (both to query meta-data and to access data) to provide read-only access to world-wide replicated data via virtual entry points named redirectors. Today, the two main implementations of this federation are AAA [2] (Any data, Any time, Anywhere) for CMS and FAX [3] (Federated ATLAS XRootD) for ATLAS. The AAA federation is used daily by more than 600 distinct users accessing hundreds of thousands of files geographically distributed across 750 servers. Identifying access patterns and estimating data traffic is becoming more and more important to understand the dataflow and be able to propose optimal data placement strategies. For this goal to be achieved, the monitoring functionality of the federation should cover the status of all components. In addition, statistics mined in the collected monitoring data have to be well presented to provide a deep insight into new usage patterns of the storage resources. This paper will first cover the monitoring flow and the challenges it brings. Then the modular architecture of the monitoring chain will be described, explaining how it can scale and meet future expectations. Finally, a closer look will be taken at the current implementation of the XRootD Dashboards.

2. Monitoring dataflow

Monitoring a system at the scale of an XRootD federation is a challenge because of the complexity introduced by the large number of data-servers (O(1000)) distributed across the Internet. Furthermore, on the server level, monitoring should only impose a limited overhead and be non-intrusive. Limited overhead implies that the network resources dedicated to monitoring should not exceed a few percent while non-intrusive means that the monitoring activity should not block the system. The first section will demonstrate how the overhead on the system is limited thanks to the use of monitoring streams. The second will expose the effect on the reliability of monitoring data when a non-intrusive approach is used.

2.1. Monitoring streams

Many metrics on a data server can be monitored : user authentication failures, redirection rates, file accesses, etc. To group these metrics by category, XRootD introduces the concept of monitoring streams [4]. Depending on the quantity of data generated by a stream, the format will not be the same. Human-readable formats such as XML will be preferred for low event rate summary data while a much more concise binary encoded data format will be required for high event rate detailed streams. For the needs of our application - monitoring file accesses - detailed data-streams are required. Within these streams, every single UNIX IO operation is reported (open, read, write, ...) with the user mapping for an approximate maximum overhead of 3% [5]. All together the average incoming traffic is about 120kB/s for all the data-streams.

2.2. Reliability

While monitoring is extremely important, this is not the main goal of the data server and should not impact it (or within a reasonable limit). At the same time, the aim of monitoring is to show the global picture of the system; a small monitoring data loss is then tolerated. In other words, it is not the responsibility of a data server to make sure that the monitoring data are correctly consumed, and the act of monitoring must not block a data server. These two conditions allow the UDP protocol to be used upstream of the UDP collector. Downstream of the UDP collector, a reliable protocol can be used without impact on the data server so no data loss is tolerated and all data sent by the UDP collector should reach the database within a reasonable time. To ensure this reliability, the stateful TCP protocol is used in addition to some acknowledgement methods between components along the chain.

3. Components of the monitoring chain

The monitoring chain is highly modular and relies on well-defined interfaces between each of its components. This architecture allows the scaling challenges to be met by adopting different strategies depending on the layer. In addition, it facilitates the technology watch in order to benefit from the latest advances. For example, investigation has started in the messaging layer with Apollo to replace ActiveMQ and in the storage layer with the Hadoop ecosystem or ElasticSearch to replace Oracle. The following schema (figure 1) shows the interaction between the components and this section describes each component.

3.1. GLED collector

As described previously, the monitoring system is based on streams. For our purpose, two data-streams are used: one reporting every single UNIX IO operation and one keeping track of the

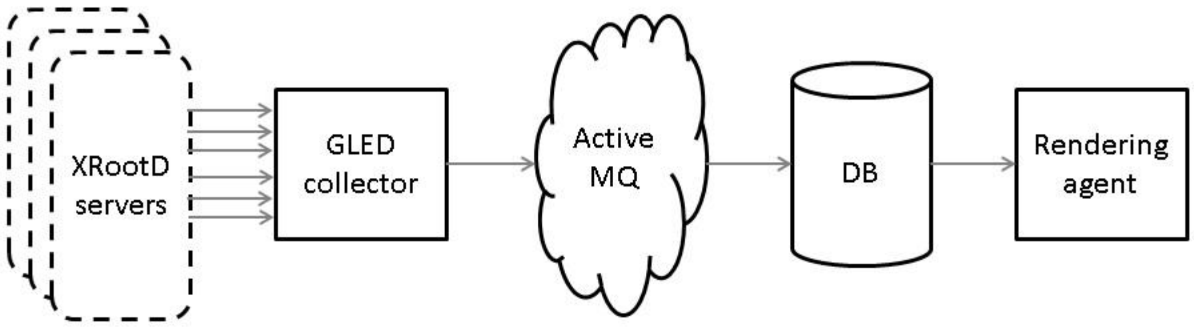


Figure 1. Workflow of monitoring data

user mapping to these operations. Individually, they are not understandable and the role of the GLED collector [6, 7] is to contextualize them. Because a simple read of a file could potentially generate a huge number of events (an open, many reads and a close), the GLED collector also needs to keep track of all IO events belonging to a single file operation. As an output, the GLED collector sends a JSON object containing the full details of the file operation. Example : User A has read file B on server C from site D, 10 GB of data have been read (out of 15 GB) using 1000 vector reads.

3.2. Transport Layer: ActiveMQ
From the GLED collector to the storage layer, data are transported using the ActiveMQ messaging technology. Monitoring data are transported asynchronously using the STOMP protocol over TCP, allowing for the producer and the consumer to be independent. When running smoothly, almost no delay is imposed by the messaging layer: data are transported in near real-time. On the contrary, the messaging layer can act as a buffer if a problem occurs downstream of it by saving all the incoming messages and keeping them until acknowledgement by the consumer.

3.3. Storage layer
The primary goal of the storage layer is to store the raw data and aggregate them into various kinds of statistics to speed-up the rendering on the user interface. The Oracle RDMBS used computes aggregations locally through PL/SQL procedures avoiding network activity and potential overhead. Different conservation policies are applied to the recorded data. As the primary goal of the raw data is to be aggregated and not accessed by the end user, keeping them for 3 months is enough while the statistics should be kept indefinitely.

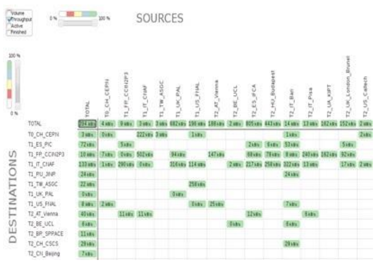
3.4. RESTful API
In this design, there is no direct access to the database from the user interface, all accesses must go through a well defined RESTful API. This API is provided by a Python application running on top of an Apache web server. Its main role is to trigger all the SQL queries to the database, enhance the results by adding meta-data such as topology (site name, country, VOs naming convention), and apply complex filtering and grouping. Then the API is able to return results in different human-readable standard formats such as JSON or XML.

3.5. Visualisation interface
The XRootD monitoring data are finally displayed within a web browser using the latest JavaScript technologies: jQuery [8] and Highcharts [9] connected together by Dashboard in-house MVC μ-framework [10] (xbrowse).

4. Benefits of the architecture
4.1. Scalability
All together, the traffic of the AAA and FAX federations generate roughly 5 million monitoring events per day. All these events have to be recorded in the database and they represent 1.6GB per day, each event having an average size of 350 bytes. This activity will constantly grow together with the federations until the end of their full deployment. However, the presented system is ready to meet this scaling challenge. Upstream of the database, GLED collectors can be instantiated per region (or with a finer granularity) and the ActiveMQ servers scale horizontally. At the database level, traffic can be split into different objects and mined using different procedures (load-balanced over the whole cluster).

4.2. Common solution
Some applications such as XRootD monitoring and XRootD popularity may need to extract statistics for different purposes from the same raw data. Instead of relying on two different workflows as in the past, a unified approach is now used. This solution presents many advantages, no duplication at the raw data level releasing messaging and storage resources, but also ensuring the statistics are extracted from exactly the same data. Not only the workflow is shared but also the development effort between the applications. The level of customization for the two main experiments is very low allowing feedback from one to benefit the other.

5. AAA and FAX Dashboards
AAA and FAX Dashboards [11, 12] have been designed to cover a wide range of monitoring use cases. First, to have a global view of the federation as shown by the matrix or the map views in figures 2 and 3.



	TOTAL	T0_CH_CERN	T1_FR_CCIN2P3	T1_IT_CNAF	T1_TW_ASGC	T1_UK_RAL	T1_US_FNAL	T2_AT_Vienna	T2_BE_UCL	T2_ES_IFCA	T2_HU_Budapest	T2_IT_Bari	T2_IT_Pisa	T2_UA_KIPT	T2_UK_London_Brunel	T2_US_Caltech
TOTAL	[illegible]	4 kB/s	9 kB/s	3 kB/s	3 kB/s	682 kB/s	190 kB/s	188 kB/s	2 kB/s	805 kB/s	443 kB/s	14 kB/s	13 kB/s	182 kB/s	152 kB/s	2 kB/s
T0_CH_CERN	3 kB/s	0 kB/s		222 kB/s	3 kB/s		1 kB/s					1 kB/s				2 kB/s
T1_ES_PIC	72 kB/s		5 kB/s							2 kB/s	6 kB/s	53 kB/s			5 kB/s	
T1_FR_CCIN2P3	10 kB/s	7 kB/s	0 kB/s	502 kB/s		94 kB/s		147 kB/s		68 kB/s	78 kB/s	8 kB/s	240 kB/s	182 kB/s	92 kB/s	
T1_IT_CNAF	133 kB/s	1 kB/s	290 kB/s	0 kB/s		316 kB/s	114 kB/s		2 kB/s	217 kB/s	258 kB/s	322 kB/s	13 kB/s		17 kB/s	2 kB/s
T1_RU_JINR	24 kB/s											24 kB/s				
T1_TW_ASGC	22 kB/s						258 kB/s									
T1_UK_RAL	0 kB/s					0 kB/s										
T1_US_FNAL	8 kB/s	2 kB/s					0 kB/s	25 kB/s				7 kB/s				
T2_AT_Vienna	40 kB/s		11 kB/s	11 kB/s						12 kB/s			6 kB/s			
T2_BE_UCL	6 kB/s								0 kB/s			6 kB/s				
T2_BR_SPRACE	11 kB/s															
T2_CH_CSCS	29 kB/s											29 kB/s				
T2_CN_Beijing	7 kB/s															

Figure 2. Matrix representation of the transfers in the federation in a given time window.

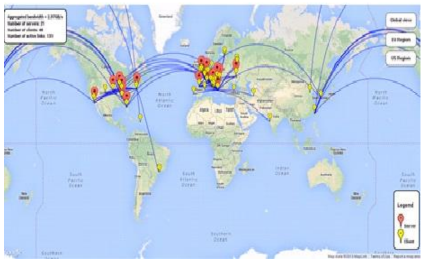


Figure 3. Map representation of the transfers in the federation in a given time window.

The second goal of this application is to allow site administrators to make deep analyses. Many federation characteristics can be understood such as site access patterns, user behaviour and data locality optimisation. From figure 4, a site administrator is able to understand which remote users are mainly accessing his site, while figure 5 would allow him to understand which are the most read files from his site. The application now consists of 7 distinct views with associated filters and different plotting options.

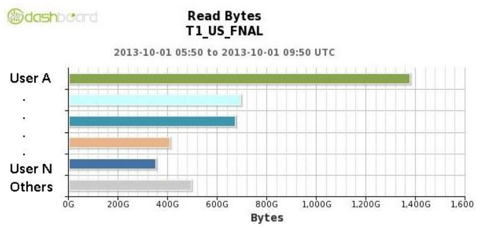


Figure 4. User accessing FNAL.

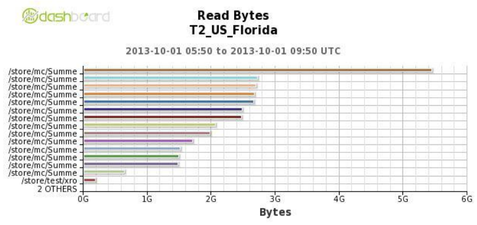


Figure 5. File accessed by Florida users.

6. Beyond XRootD federations

While XRootD is today the most used federated storage technology, other technologies such as the HTTP/WebDAV dynamic federation [13] will be used in the future. Monitoring this new technology will be a hot topic in the future. The monitoring tool described throughout this paper is built using common building blocks and distinct layers communicating through well-defined APIs. This generic design greatly simplifies the implementation of monitoring in different contexts: for example, FTS and ATLAS DDM monitoring are based on exactly the same framework. Thanks to this experience, monitoring the future HTTP/WebDAV federation will be relatively fast and easy to implement as soon as the system is correctly instrumented and monitoring use-cases understood.

7. Conclusion

The AAA and FAX federations are currently actively used and growing quickly. All together, about 5 million monitoring events are recorded and mined daily and the visualization interfaces are used by an increasing number of users. The various monitoring views in the application cover a wide-range of use cases allowing both in-depth analysis as well as a global overview of the federation. By design, the application is ready to meet the challenges imposed by the exponential growth of federation usage, following horizontal scaling strategies where applicable and analysing new technologies.

References

[1] XRootD project page: http://www.xrootd.org/
[2] AAA project page: https://twiki.cern.ch/twiki/bin/view/Main/CmsXrootdArchitecture
[3] FAX project page: https://twiki.cern.ch/twiki/bin/view/AtlasComputing/AtlasXrootdSystems
[4] Xrootd system monitoring documentation: http://xrootd.slac.stanford.edu/doc/prod/xrd_monitoring.htm
[5] Hanushevsky A, Presentation on High Performance Monitoring, 2013
https://indico.cern.ch/getFile.py/access?contribId=0&resId=1&materialId=slides&confId=220304
[6] Gled web page: http://www.gled.org/cgi-bin/twiki/view/Main/XrdMon
[7] Tadel M, *Gled - an Implementation of a hierarchic Server-client Model*, 2004 *Nova Science Publishers Vol.* **16** 1-59454-174-4
[8] JQuery web page: http://http://jquery.com/
[9] Highcharts web page: http://www.highcharts.com/
[10] Andreeva J et al, *Designing and developing portable large-scale JavaScript web applications within the Experiment Dashboard framework*, 2012 *J. Phys.: Conf. Ser.* **396** 052069
[11] AAA Dashboard: http://dashb-cms-xrootd-transfers.cern.ch
[12] FAX Dashboard: http://dashb-atlas-xrootd-transfers.cern.ch
[13] Furano F, Brito da Rocha R, Devresse A, Keeble O, Alvarez Ayllon A and Fuhrmann P, *The Dynamic Federations: Federate Storage on the fly using HTTP/WebDAV and DMLite*, 2013 *The International Symposium on Grids and Clouds*