Xrootd Monitoring for the CMS Experiment

L A T Bauerdick[1], K Bloom[3], B Bockelman[3], D C Bradley[4], S Dasu[4], I Sfiligoi[2], A Tadel[2], M Tadel[2,5], F Wuerthwein[2] and A Yagil[2] for the CMS collaboration

[1] Fermilab, Batavia, IL 60510-5011, USA
[2] UC San Diego, La Jolla, CA 92093, USA
[3] University of Nebraska – Lincoln, Lincoln, NE 68588, USA
[4] University of Wisconsin – Madison, Madison, WI 53706, USA

E-mail: mtadel@ucsd.edu

Abstract. During spring and summer of 2011, CMS deployed Xrootd-based access for all US T1 and T2 sites. This allows for remote access to all experiment data on disk in the US. It is used for user analysis, visualization, running of jobs at computing sites when data is not available at local sites, and as a fail-over mechanism for data access in jobs. Monitoring of this Xrootd infrastructure is implemented on three levels. Basic service and data availability checks are performed by Nagios probes. The second level uses Xrootd's "summary data" stream; this data is aggregated from all sites and fed into a MonALISA service providing visualization and storage. The third level uses Xrootd's "detailed monitoring" stream, which includes detailed information about users, opened files and individual data transfers. A custom application was developed to process this information. It currently provides a real-time view of the system usage and can store data into ROOT files for detailed analysis. Detailed monitoring allows us to determine dataset popularity and to detect abuses of the system, including sub-optimal usage of the Xrootd protocol and the ROOT prefetching mechanism.

1. Introduction

The CMS experiment [1] at the CERN LHC started taking data in the fall of 2009. The experiment data in various formats is distributed among participating Tier 0, Tier 1 and Tier 2 computing sites with the goal of optimizing the usage of available computing resources as well as to provide sufficient processing power to all physicists that require access to the data. As part of this arrangement all data that is required for physics analysis is available from the US Tier 1 and Tier 2 sites; a large part of it is even replicated to two or more sites. In 2010, several US Tier 2 centers started using Xrootd system [2] to expose their site storage to all CMS users. This provides the ability of direct, interactive access to data. It became immediately obvious that the availability of such service provides several possibilities to extend and improve the CMS computing model. The "*Anydata, Anytime, Anywhere*" project (AAA) [3] was started in Fall 2011 with the goal of exploring these possibilities and bringing them into production. The presented monitoring work is part of the AAA program.

One aspect of AAA is the deployment and operation of the Xrootd. This part has been completed and USCMS is currently operating an Xrootd data federation across eight sites and with a total of more than 750 data servers. The top-level redirector is operated at University of Nebraska-Lincoln

[5] To whom any correspondence should be addressed.

(UNL) and all other participating sites operate a local redirector in front of their severs. Other than providing users the direct data access, this infrastructure already operates two additional extended use-cases that are in fact already accounting for a majority of all Xrootd data-access:

i. *File-open fallback* allows a CMSSW job which fails to open a local file to fall-over to remote reading of the file via Xrootd, as opposed to failing the job.
ii. *Job overflow* from oversubscribed sites allows jobs to run on nearby sites with spare compute resources and read data remotely via Xrootd [4].

It was recognized from the start that monitoring of Xrootd operations and user activity would play an important role in transition from testing to production phase of operations. Extensive monitoring also allows detailed accounting and analysis of system usage. With rapid acceptance of fallback and overflow use-cases, its role as the guardian of production-level quality of service has become even more important.

This paper is divided into three sections. Section 2 describes service and data availability monitoring using Nagios. Collection of summary monitoring information from Xrootd servers and storage of data into MonALISA [5] repository is presented in section 3. Detailed monitoring stream processing with GLED [6] is then described in section 4.

2. Service and data availability monitoring

The monitoring of service and data availability is performed using the Nagios framework. Probes are setup for each site in the Xrootd federation to determine whether the site's Xrootd service is functioning and whether it is properly accessible via the federation. These checks were designed to use the same security protocol and client library as CMSSW jobs.

2.1. Checking of meta-manager, redirection and authentication

Users usually access the federation through the top-level redirector at UNL. We check that this top-manager is on the network and does indeed redirect users to an end-data server by attempting to read out a single file. This check is performed every ten minutes by Nagios probes.

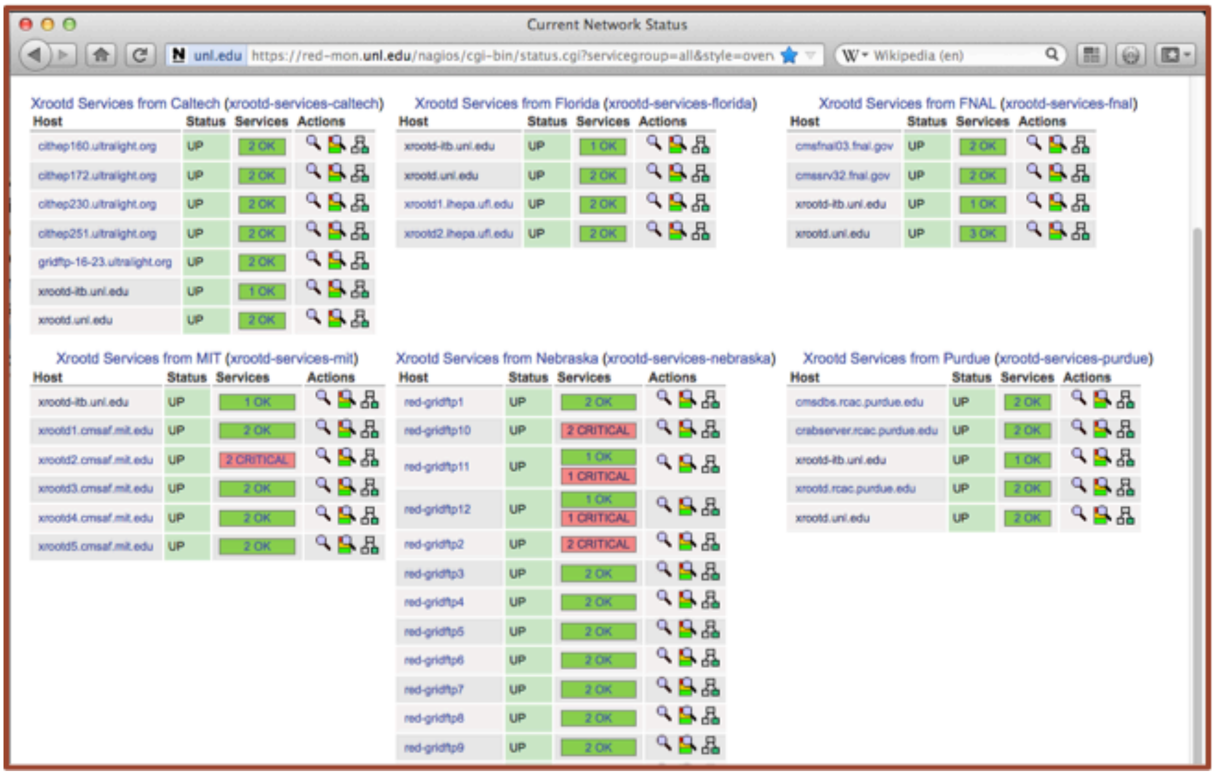
2.1.1. Checking redirection to all sites. An Xrootd meta-manager will only redirect requests to sites that have reported to possess the requested file. In order to check redirection to a specific site, one would in principle need to know about a file that only exists on the given site. This is impractical as it is error prone and fragile. We simulate unique files by performing the following namespace trick:

- The site's Xrootd service additionally exports its namespace inside a unique root based on the site name. For example, UCSD's file `/store/foo` is exported to the federation as both /store/foo and `/store/test/xrootd/T2_US_UCSD/store/foo`.
- Certain well-known datasets are required to be present at every CMS site.

Now, assuming `/store/foo` is in the required dataset, asking the meta-manager to serve `/store/test/xrootd/T2_US_UCSD/store/foo` will trigger a federation-wide name-space query. By construction, the file will only be found at UCSD. So, if the client can read data, we assume redirection to UCSD is functional.

In addition to the well-known file, a separate test queries CMS's data management services and randomly selects a file that is supposed to be present at the site.

2.1.2. Checking authentication. Xrootd servers in the federation use GSI authentication [7]. As there are components in GSI security that change daily (CRLs), monthly (CA certificates) or yearly (server certificates) it is a common source of errors and important to keep under detailed surveillance. Two sets of identical Nagios probes are run; the first with certificate issued by CERN certificate authority (CA) and then with the second with DOEgrids CA. These CAs were selected because they are the most commonly used among CMS users. For authorization, the local site authorization services are


Current Network Status

https://red-mon.unl.edu/nagios/cgi-bin/status.cgi?servicegroup=all&style=overv

Xrootd Services from Caltech (xrootd-services-caltech)

Host	Status	Services	Actions
cithep160.ultralight.org	UP	2 OK	
cithep172.ultralight.org	UP	2 OK	
cithep230.ultralight.org	UP	2 OK	
cithep251.ultralight.org	UP	2 OK	
gridftp-16-23.ultralight.org	UP	2 OK	
xrootd-itb.unl.edu	UP	1 OK	
xrootd.unl.edu	UP	2 OK	

Xrootd Services from Florida (xrootd-services-florida)

Host	Status	Services	Actions
xrootd-itb.unl.edu	UP	1 OK	
xrootd.unl.edu	UP	2 OK	
xrootd1.ihepa.ufl.edu	UP	2 OK	
xrootd2.ihepa.ufl.edu	UP	2 OK	

Xrootd Services from FNAL (xrootd-services-fnal)

Host	Status	Services	Actions
cmsfnal03.fnal.gov	UP	2 OK	
cmssrv32.fnal.gov	UP	2 OK	
xrootd-itb.unl.edu	UP	1 OK	
xrootd.unl.edu	UP	3 OK	

Xrootd Services from MIT (xrootd-services-mit)

Host	Status	Services	Actions
xrootd-itb.unl.edu	UP	1 OK	
xrootd1.cmsaf.mit.edu	UP	2 OK	
xrootd2.cmsaf.mit.edu	UP	2 CRITICAL	
xrootd3.cmsaf.mit.edu	UP	2 OK	
xrootd4.cmsaf.mit.edu	UP	2 OK	
xrootd5.cmsaf.mit.edu	UP	2 OK	

Xrootd Services from Nebraska (xrootd-services-nebraska)

Host	Status	Services	Actions
red-gridftp1	UP	2 OK	
red-gridftp10	UP	2 CRITICAL	
red-gridftp11	UP	1 OK 1 CRITICAL	
red-gridftp12	UP	1 OK 1 CRITICAL	
red-gridftp2	UP	2 CRITICAL	
red-gridftp3	UP	2 OK	
red-gridftp4	UP	2 OK	
red-gridftp5	UP	2 OK	
red-gridftp6	UP	2 OK	
red-gridftp7	UP	2 OK	
red-gridftp8	UP	2 OK	
red-gridftp9	UP	2 OK	

Xrootd Services from Purdue (xrootd-services-purdue)

Host	Status	Services	Actions
cmsdbs.rcac.purdue.edu	UP	2 OK	
crabserver.rcac.purdue.edu	UP	2 OK	
xrootd-itb.unl.edu	UP	1 OK	
xrootd.rcac.purdue.edu	UP	2 OK	
xrootd.unl.edu	UP	2 OK	

Figure 1. Top-level Nagios status page for Xrootd service monitoring at UNL.

assumed to ensure that the user belongs to the CMS experiment. We do not currently test whether authenticated, but non-CMS users can access the infrastructure.

2.1.3. Checking file-system accessibility. For each file tested, only the first kilobyte of data is read. This ascertains that the storage system is minimally accessible via Xrootd, takes very little time, and does not require any bandwidth guarantees to the monitoring system.

If Nagios probe exits with an error condition or fails to complete within the one-minute interval, the service is considered to be failing and mail alarms are sent to system operators. Figure 1 shows a screenshot of the Nagios status page.

2.2. Checking of individual Xrootd servers

The tests described in section 2.1 test the functionality of the *site*, but does not reveal anything about the individual servers: the site-level redirector will always direct the client to a functioning, least loaded server. To test individual servers, probes must be directed at individual servers themselves.

Currently, the per-server tests are also done using Nagios. However, from the federation perspective, this should be a site-internal matter: the sites are supposed to run server-state checks themselves and ensure that a sufficient number of servers are functioning properly. Work is ongoing to have the health checks distributed to the sites as a part of the grid layer's monitoring tests.

2.3. Limitations of service availability tests

The probes only touch a very small part of namespace, use specific certificates and do not necessarily talk to all servers, even over a long period of time. Summary monitoring, described in the next section, provides complementary information related to server availability and performance, and also provides statistics about failure conditions that occur on every server.

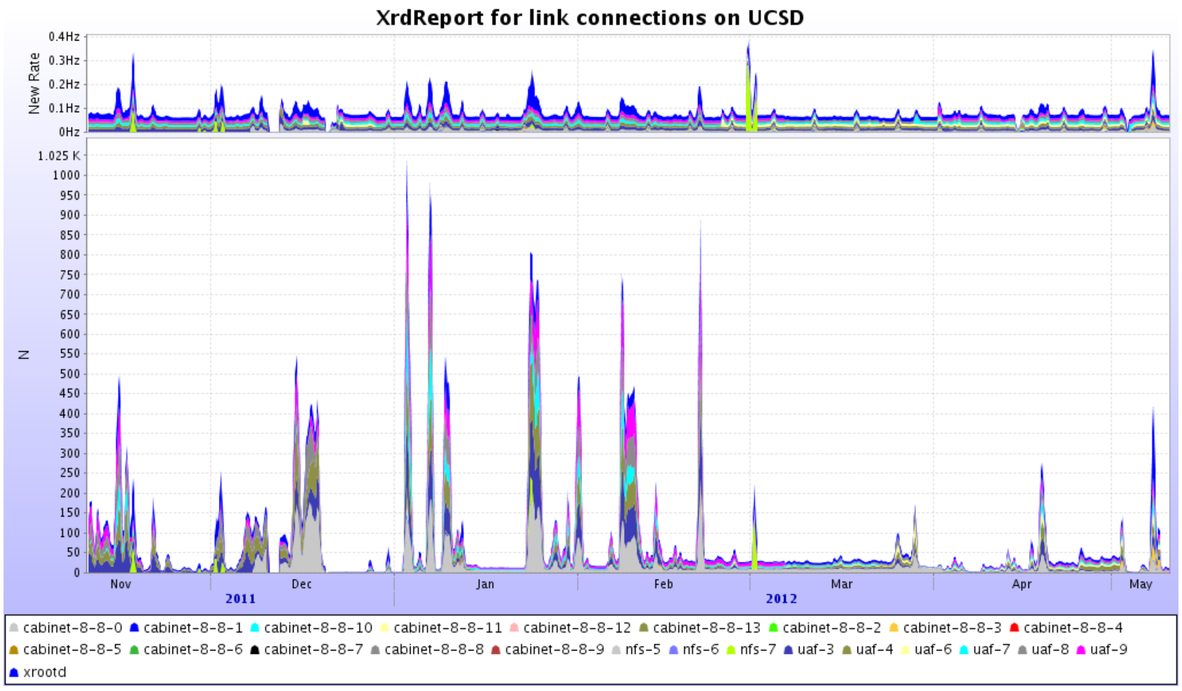


Figure 2. MonALISA graphs showing total number of connections and rate of new connections to servers at UCSD over a six-month period. Data is taken from Xrootd summary monitoring stream.

3. Xrootd summary monitoring

Each Xrootd server can be configured to send periodic reports about their operations and status to a specific network endpoint[6]. To prevent monitoring outages from adversely affecting the Xrootd service, UDP is used as a transport layer. This "summary monitoring" stream provides the data in a terse, XML-based format. It contains various counters and state variables, e.g., network traffic, usage of system resources, usage of daemon-internal resources (active connections, open files, used buffers), number of received requests, performed operations and errors. The values are mostly 32 and 64-bit integers or character strings. See [8] for a thorough description of the format.

3.1. Processing of summary monitoring packets, population of MonALISA repository

In the USCMS data federation, all redirectors and servers send their summary monitoring UDP packets[7] to a collector at UCSD. Depending on a site configuration, these are sent at a frequency of 30 to 120 seconds. The packets are received by a custom application, `xrd-rep-snatcher` [9], which normalizes the input and determines the canonical server and site names. For the variables which are cumulative counters, it calculates normalized rates of change. A subset of the values are passed from `xrd-rep-snatcher` to MonALISA for storage and presentation. Currently recorded values include:

- number of connected clients;
- rates of new connections, authentications and authentication errors;
- overall incoming and outgoing network traffic from by Xrootd;
- daemon's usage of system resources.

[6] In fact, two destination addresses can be specified.
[7] Data comes from both the xrootd and cmsd daemons.

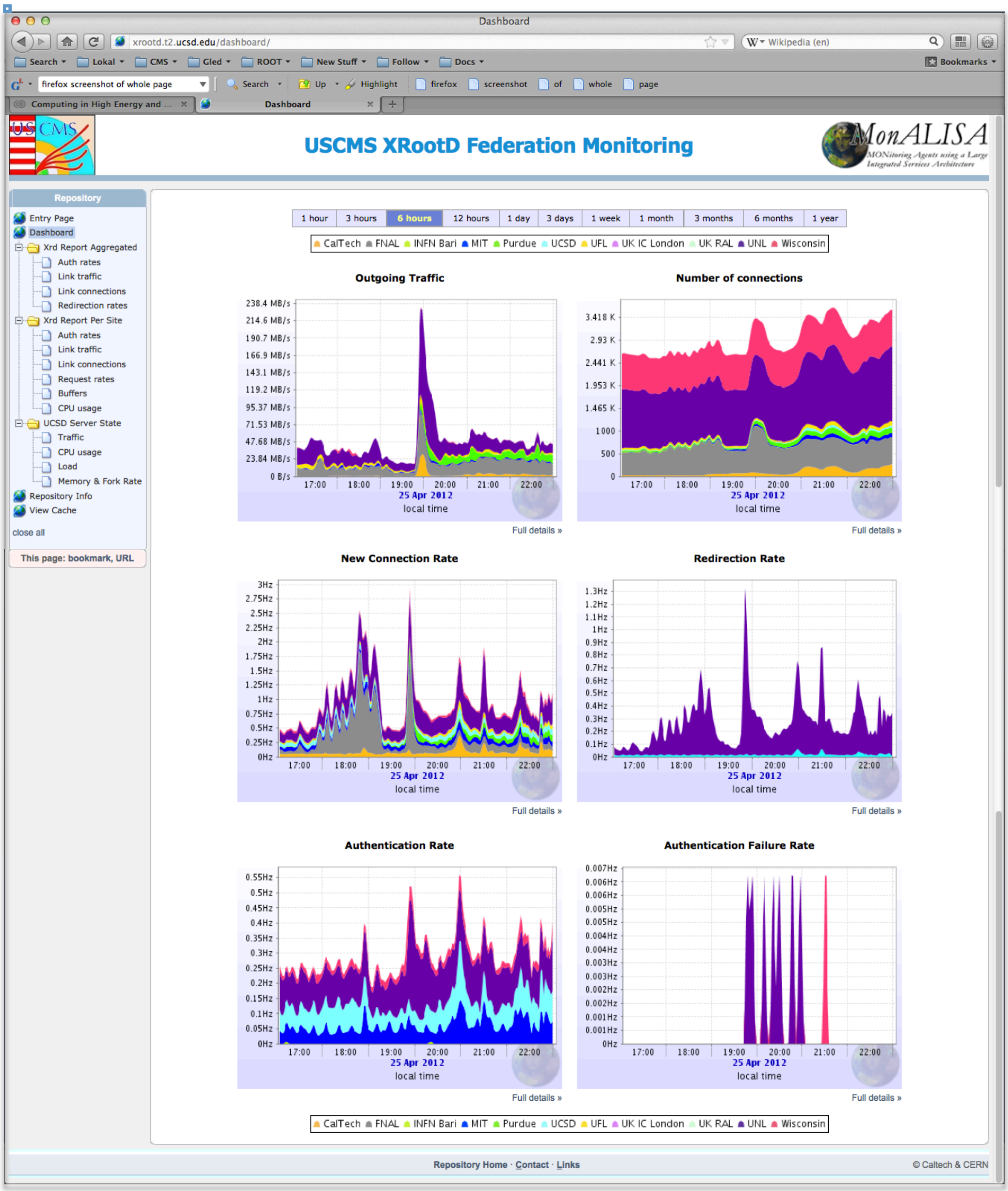
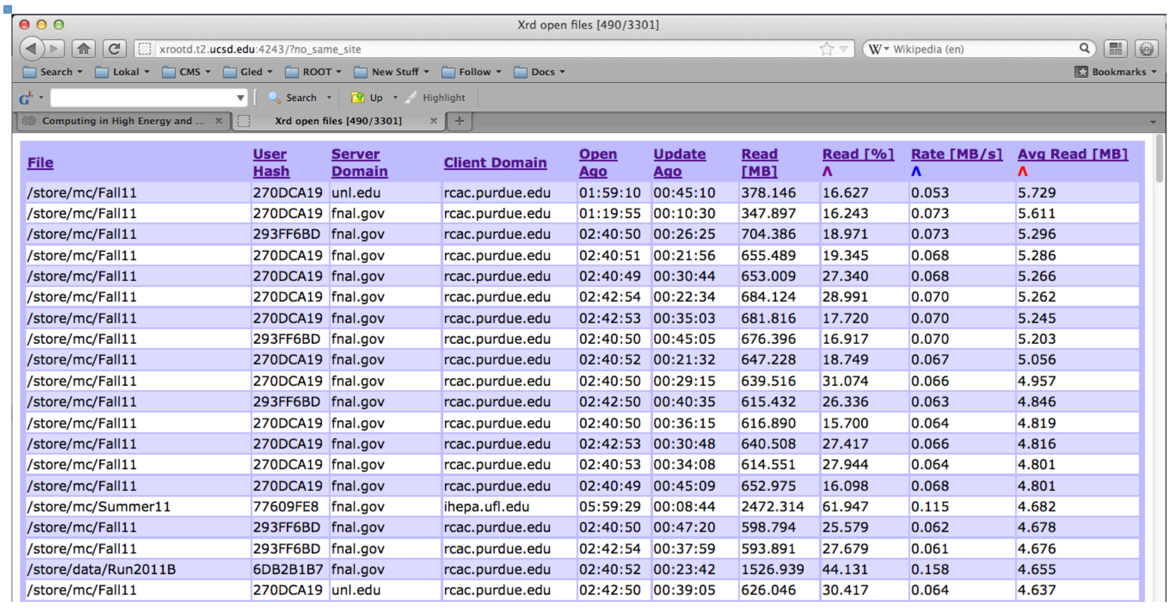


Figure 3. USCMS Xrootd federation dashboard.

For complete list of possible values, see [8] and [9].

Currently, we operate one instance of MonALISA service and one instance of MonALISA repository, both of them run at UCSD. Considerable effort was used to customize the repository web front-end [10] for this project. See Figure 3 for an example screenshot.


Xrd open files [490/3301]
xrootd.t2.ucsd.edu:4243/?no_same_site
Wikipedia (en)
Search
Lokal
CMS
Gled
ROOT
New Stuff
Follow
Docs
Bookmarks
Search
Up
Highlight
Computing in High Energy and ...
Xrd open files [490/3301]

File	User Hash	Server Domain	Client Domain	Open Ago	Update Ago	Read [MB]	Read [%] ^	Rate [MB/s] ^	Avg Read [MB] ^
/store/mc/Fall11	270DCA19	unl.edu	rcac.purdue.edu	01:59:10	00:45:10	378.146	16.627	0.053	5.729
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	01:19:55	00:10:30	347.897	16.243	0.073	5.611
/store/mc/Fall11	293FF6BD	fnal.gov	rcac.purdue.edu	02:40:50	00:26:25	704.386	18.971	0.073	5.296
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:40:51	00:21:56	655.489	19.345	0.068	5.286
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:40:49	00:30:44	653.009	27.340	0.068	5.266
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:42:54	00:22:34	684.124	28.991	0.070	5.262
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:42:53	00:35:03	681.816	17.720	0.070	5.245
/store/mc/Fall11	293FF6BD	fnal.gov	rcac.purdue.edu	02:40:50	00:45:05	676.396	16.917	0.070	5.203
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:40:52	00:21:32	647.228	18.749	0.067	5.056
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:40:50	00:29:15	639.516	31.074	0.066	4.957
/store/mc/Fall11	293FF6BD	fnal.gov	rcac.purdue.edu	02:42:50	00:40:35	615.432	26.336	0.063	4.846
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:40:50	00:36:15	616.890	15.700	0.064	4.819
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:42:53	00:30:48	640.508	27.417	0.066	4.816
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:40:53	00:34:08	614.551	27.944	0.064	4.801
/store/mc/Fall11	270DCA19	fnal.gov	rcac.purdue.edu	02:40:49	00:45:09	652.975	16.098	0.068	4.801
/store/mc/Summer11	77609FE8	fnal.gov	ihepa.ufl.edu	05:59:29	00:08:44	2472.314	61.947	0.115	4.682
/store/mc/Fall11	293FF6BD	fnal.gov	rcac.purdue.edu	02:40:50	00:47:20	598.794	25.579	0.062	4.678
/store/mc/Fall11	293FF6BD	fnal.gov	rcac.purdue.edu	02:42:54	00:37:59	593.891	27.679	0.061	4.676
/store/data/Run2011B	6DB2B1B7	fnal.gov	rcac.purdue.edu	02:40:52	00:23:42	1526.939	44.131	0.158	4.655
/store/mc/Fall11	270DCA19	unl.edu	rcac.purdue.edu	02:42:50	00:39:05	626.046	30.417	0.064	4.637

Figure 4. Html table of currently opened files. This information is served from a http server embedded in detailed monitoring processing application.

3.2. *Processing of data with custom MonALISA plugins*

MonALISA provides a natural place to do further, more complex, processing of the data from `xrd-rep-snatcher`. This custom user code is implemented as Java classes and is invoked at different data processing stages. We use two kinds of plugins:

i. Data producers for calculation of per-site quantities, e.g. summing up total traffic for each site.
ii. Data filters for detecting error conditions and sending notification emails.

3.3. *Data presentation options*

From the MonALISA web frontend two types of visualizations are provided:

- *Standard MonALISA graphs*. MonALISA has a standard generated graph page that can be used to show all data. This provides users with various filtering capabilities, such as site, nodes and time intervals. Figure 2 shows an example of a graph produced by MonALISA.
- *Overall Dashboard*. As a special case, frequently viewed graphs that show status of the whole federation at a glance has been combined into a single dashboard page (shown in Figure 3; see also [10]).

4. Xrootd detailed monitoring

In contrast to summary monitoring, the detailed monitoring contains details about individual user sessions, open files, data transfers. For redirectors, monitoring data is produced for each client redirection and query. The information is sent in a highly encoded binary format and also transmitted via UDP. Every effort is made to reduce the size of the detailed monitoring packets; otherwise the size of the monitoring could rival the data transfers themselves. User sessions and open files are identified by a *dictionary-id* assigned at resource allocation and immediately reported in a dedicated UDP packet. Details about individual file transfers are collected independently for each session and are sent in a large UDP packet when the server-side buffer is full[8] or after a specified amount of time elapses. Each read's size and offset are recorded independently; vector reads are marked and reported

[8] Buffer sizes are between 1 and 64kB. For USCMS servers we use between 1.5 and 4kB.

as one read with read offset being replaced with the information about the number of elements requested within the vector read[9].

When using GSI authentication, the servers can be configured so that the user session mapping records contain full details about the user certificate and VOMS extensions [11]. Almost all USCMS servers are configured this way. Using this information, we can correlate user sessions across multiple sites and identify the user responsible for each transfer. We maintain a detailed accounting database to detect excessive or abusive usage of the system.

4.1. Collection of detailed monitoring packets

All the federation servers report detailed monitoring directly to UCSD. At UCSD, the UDP streams are collected and redistributed by a custom UDP-to-TCP replicator[10]. This allows the single stream to be fed to several analyzing programs (including the testing infrastructure). The implementation is general enough to allow installation of replicators at multiple sites to convert the stream to a reliable delivery protocol[11], provide compression and, as desired, local analysis.

4.2. Description of detailed stream processing with GLED

The processing of detailed monitoring packets is implemented in the GLED framework [6] as the XrdMon package. To re-assemble information scattered throughout the UDP monitoring packets, a complete in-memory representation of all servers, sessions and open files is built. Lookup tables based on the dictionary-id are maintained for each server.

At present, the following derived representations are currently implemented:

i. An HTML table showing currently open files and ongoing data transfers. The data is served via an HTTP server embedded in the XrdMon package. Filtering based on contents of all string columns is supported. To protect privacy of users, the page with all details is only available to the members of CMS experiment. A public page with sensitive information removed is also available (shown in Figure 4).

ii. When a message that a file has been closed is received (or a file closed due to an error condition or time-out), a detailed file-read report is generated. It contains all details about the session, connection, user, and file transfer. Information about single reads and vector reads is represented statistically. For non-vector reads, the information contains the number of read operations and total amount of data read. For the size of individual reads the minimum, maximum, average and the standard deviation are reported.

Several external reporters can be fed the information from (ii). One implementation sends the data into OSG Gratia accounting database [12]. This is used as the basis for generation of detailed daily usage emails that contain the site-to-site matrix and breakdown of all traffic by user and by site. The second implementation writes the data into a ROOT tree for further analysis. Example of a ROOT histogram derived from this information is shown in Figure 5.

[9] Optionally, each vector read can be fully unpacked and reported with individual element offsets and read sizes
[10] Also implemented using GLED; this allows users to combine packet servers or clients with stream analyzers.
[11] Packet loss can be rather fatal for detailed monitoring. If a dictionary-id mapping packet for a user session is lost, it becomes impossible to process all subsequent packets referring to this session.

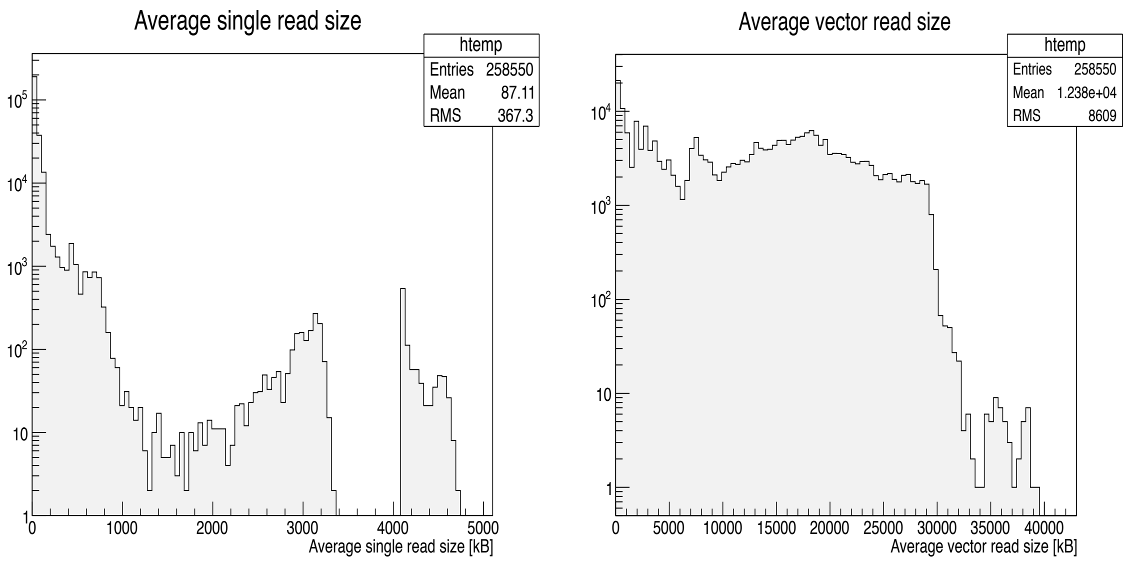


Figure 5. Average read size per session, separated between single reads (left) and vector reads (right). Information was extracted from detailed monitoring file-read reports from April 26 to April 30 2012

5. Conclusion

Xrootd monitoring for the needs of USCMS Xrootd data federation is implemented on several complementary levels. While each of them uses a fundamentally different approach, the derived results do have enough overlap to allow for cross-checking of the results and to assure that all monitoring components are functioning properly. Dashboard-like web-pages are available on all levels and automatic e-mail alerts get sent when error conditions are detected. All monitoring results are archived and can be used for detailed diagnostics of problems or for analysis of service performance and user access patterns. As an additional benefit, the Xrootd monitoring sub-systems have been thoroughly validated and improved during integration of monitoring infrastructure and services.

Future work will focus on extensions of Xrootd monitoring system to support fast detection of data-transfer bottlenecks. This information will be used by the Xrootd redirectors and clients so that the problematic servers or routes can be avoided or, if already taken, substituted with better alternatives.

Acknowledgments

This work is partially sponsored by the US National Science Foundation under Grants No. PHY-0612805 (CMS Maintenance & Operations), PHY-1104549, PHY-1104447, and PHY-1104664 (AAA), and the US Department of Energy under Grant No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG).

References

[1] CMS Collaboration 2008 *JINST* **3** S08004
[2] Xrootd project page: http://www.xrootd.org/
[3] Bauerdick L *et al* 2012 Using Xrootd to Federate Regional Storage *in these proceedings*.
[4] Sfiligoi I *et al* 2012 Controlled overflowing of data-intensive jobs from oversubscribed sites *in these proceedings*.
[5] Legrand I *et al* 2009 *Comput. Phys. Commun.* **180** 2472
[6] Tadel M 2004 Gled - an Implementation of a hierarchic Server-client Model *Applied Parallel and Distributed Computing (Advances in Computation: Theory and Practice* vol 16) ed Pan Y and Yang L T (Nova Science Publishers) ISBN 1-59454-174-4
[7] GSI: http://www.globus.org/security

[8] Xrootd system monitoring documentation:
http://xrootd.slac.stanford.edu/doc/prod/xrd_monitoring.htm
[9] xrd-rep-snatcher source code: http://svn.gled.org/var/trunk/xrd-rep-snatcher/
[10] US CMS Xrootd federation MonALISA web front-end: http://xrootd.t2.ucsd.edu/
[11] VOMS http://www.globus.org/grid_software/security/voms.php
[12] Gratia project home-page: https://twiki.grid.iu.edu/bin/view/Accounting/WebHome