Journal of Physics: Conference Series **396** (2012) 042001 doi:10.1088/1742-6596/396/4/042001

An optimization of the ALICE XRootD storage cluster at the Tier-2 site in Czech Republic

D Adamova[1] and J Horky[2] (for the ALICE collaboration)

[1] Nuclear Physics Institute, Rez near Prague, Czech Republic

[2] FZU, Prague, Na Slovance 1999/2, Czech Republic

E-mail: adamova@ujf.cas.cz

Abstract. ALICE, as well as the other experiments at the CERN LHC, has been building a distributed data management infrastructure since 2002. Experience gained during years of operations with different types of storage managers deployed over this infrastructure has shown, that the most adequate storage solution for ALICE is the native XRootD manager developed within a CERN - SLAC collaboration. The XRootD storage clusters exhibit higher stability and availability in comparison with other storage solutions and demonstrate a number of other advantages, like support of high speed WAN data access or no need for maintaining complex databases. Two of the operational characteristics of XRootD data servers are a relatively high number of open sockets and a high Unix load. In this article, we would like to describe our experience with the tuning/optimization of machines hosting the XRootD servers, which are part of the ALICE storage cluster at the Tier-2 WLCG site in Prague, Czech Republic. The optimization procedure, in addition to boosting the read/write performance of the servers, also resulted in a reduction of the Unix load.

1. Introduction: ALICE Global Storage system

The ALICE experiment [1] distributed storage system (see Figure 1) uses the storage manager Scalla/XRootD [2], developed within a SLAC-CERN collaboration (originally, it was a common project of SLAC and INFN). After CERN got involved, the XRootD was bundled in ROOT [3] as a generic platform for distributed data access. This is very well suited for the ALICE data analysis. "xrootd" is just the name of the data access daemon, the complete suite is called Scalla/XRootD, Scalla meaning Structured Cluster Architecture for Low Latency Access.

According to the ALICE Computing model [4], several replicas of ESDs (Event Summary Data) and AODs (Analysis Object Data) are stored across the system. The jobs are sent to sites which publish at least some of the needed data, pulled by the local JobAgents. The file access is reliable, with auto discovery and failover in a case of temporary problems. When the jobs do not need to read files from distant Storage Elements (SEs) over high latency networks, the file access is quite efficient.

ALICE::NECTEC::SE	30 TB	142.7 GB	29.86 TB	0.465%	9,613	FILE	30 TB	5.801 TB	24.2 TB	19.34%
ALICE::NIHAM::FILE	895 TB	470 TB	425 TB	52.52%	12,489,240	FILE	894.9 TB	464.3 TB	430.6 TB	51.88%
ALICE::NSC::SE	400 TB	0	400 TB	-	0	FILE	-	-	-	-
ALICE::OSC::SE	30 TB	0	30 TB	-	0	FILE	-	-	-	-
ALICE::PNPI::SE	40 TB	22.33 TB	17.67 TB	55.82%	462,992	FILE	39.95 TB	24.67 TB	15.29 TB	61.74%
ALICE::Poznan::SE	88.59 TB	63.31 TB	25.28 TB	71.47%	885,935	FILE	88.59 TB	86.07 TB	2.521 TB	97.15%
ALICE::Prague::SE	538.7 TB	285.7 TB	253 TB	53.03%	2,851,047	FILE	538.7 TB	292 TB	246.7 TB	54.2%
ALICE::RAL::SE	188 TB	73.8 TB	114.2 TB	39.26%	500,720	CASTOR	-	-	-	-
ALICE::RRC-KI::SE	313 TB	200.3 TB	112.7 TB	64%	3,216,878	FILE	318.9 TB	227.5 TB	91.37 TB	71.34%
ALICE::SaoPaulo::SE	51 TB	1.264 TB	49.74 TB	2.479%	61,023	FILE	50.9 TB	4.375 TB	46.53 TB	8.595%
ALICE::SARA::DCACHE	200 TB	189.5 TB	10.51 TB	94.74%	1,989,035	SRM	-	-	-	-
ALICE::SPbSU::SE	61.7 TB	20.97 TB	40.73 TB	33.98%	495,047	FILE	-	-	-	-
ALICE::Strasbourg_IRES::SE	69.84 TB	55.24 TB	14.6 TB	79.09%	1,182,078	FILE	52.83 TB	51.56 TB	1.274 TB	97.59%
ALICE::Subatech::SE	313.3 TB	244.7 TB	68.64 TB	78.09%	4,517,682	FILE	313.3 TB	253 TB	60.27 TB	80.76%
ALICE::SUT::SE	30 TB	0.154 GB	30 TB	0.001%	54	FILE	30 TB	15.26 GB	29.99 TB	0.05%
ALICE::Torino::SE	469.5 TB	195.8 TB	273.7 TB	41.69%	3,059,567	FILE	469.5 TB	390.3 TB	79.2 TB	83.13%
ALICE::Trieste::SE	148 TB	16.5 TB	131.5 TB	11.15%	310,038	FILE	251 TB	160.1 TB	90.9 TB	63.79%
ALICE::Troitsk::SE	94.49 TB	50.39 TB	44.1 TB	53.32%	1,130,189	FILE	94.49 TB	62.86 TB	31.63 TB	66.52%
ALICE::Trujillo::SE	170 TB	26.62 TB	143.4 TB	15.66%	396,019	FILE	170.1 TB	65.12 TB	105 TB	38.29%
ALICE::Wuhan::SE	6.3 TB	6.599 GB	6.294 TB	0.102%	165	FILE	6.245 TB	586.8 GB	5.672 TB	9.177%
	15.93 PB	**8.66 PB**	**7.442 PB**		**135,125,027**		**17.93 PB**	**12.41 PB**	**5.519 PB**	

Figure 1. ALICE disk Storage Elements (a part of the list). Total disk capacity is ~ 15.93 PB.

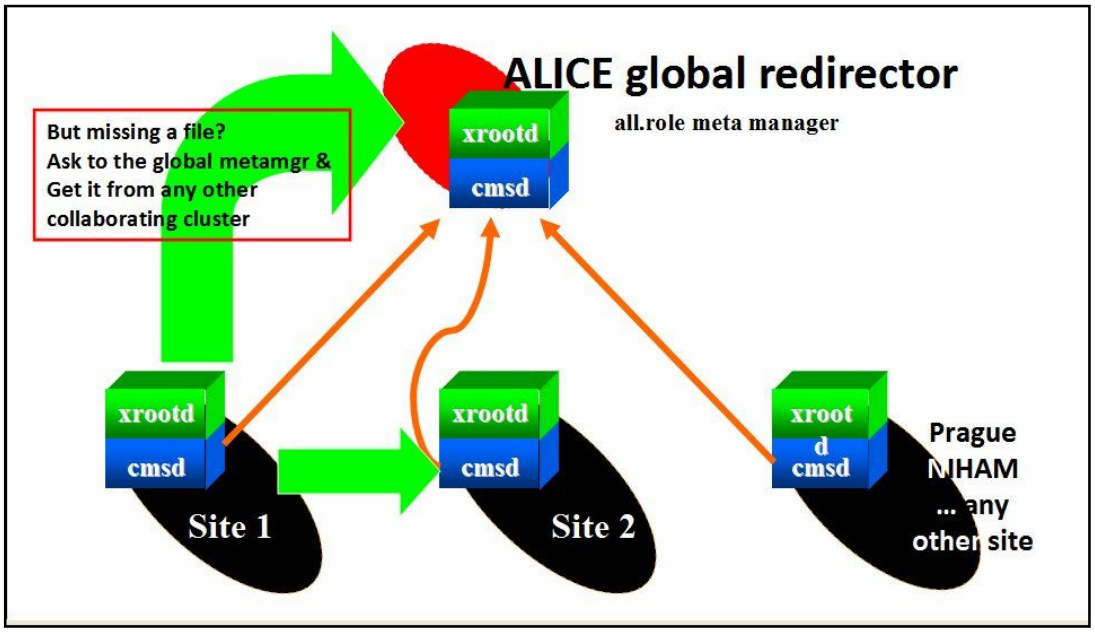


Figure 2. ALICE XRootD failover schema.

ALICE uses the system of one global and a set of regional redirectors which provide cache of files locations (see Figures 2 and 3). If a file is missing locally, an inquiry is done to the global redirector/metamanager and the file is accessed at another collaborating cluster, possibly the closest working SE. The processing of the job continues at the original site without a crash.

In this paper, we will describe the XRootD cluster providing services for ALICE at a Worldwide LHC Computing Grid (WLCG) [6] Tier-2 site in Czech Republic and a tuning/optimization of machines hosting the XRootD servers that we performed with the aim to boost their read/write performance. The optimization procedure, in addition to enhancing the performance of the servers, also resulted in a reduction of the Unix load [5].

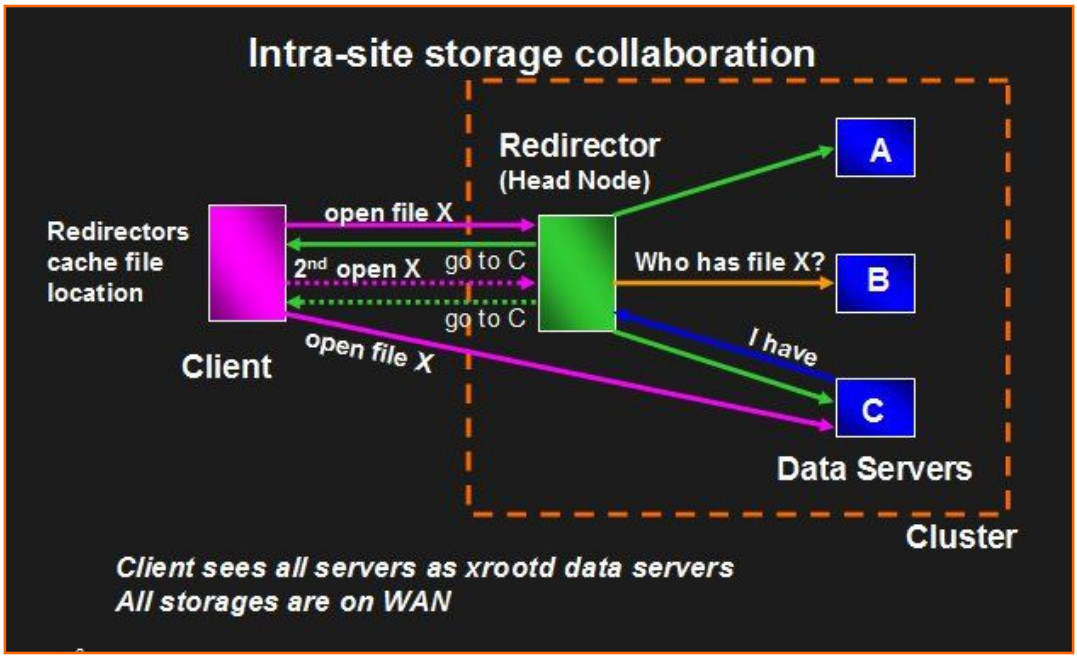


Figure 3. The function of regional XRootD redirectors. When a client wants to open a file, the redirector finds out which data server has the file and provides this information to the client. The actual data transfer is directly between the client and the data server.

2. ALICE experience with the XRootD performance

The choice of the pure XRootD as the storage solution for ALICE is based on experience with different types of storage managers gained by ALICE during 10 years of building its distributed computing and storage infrastructure. With hundreds of millions of files stored over its Storage Elements, ALICE is taking a full advantage of the primary goal of the Scalla/XRootD project, namely to create data repositories with no reasonable size limit, with a high data access performance and linear scaling capabilities [2]. The framework can serve any kind of data, organized as a hierarchical filesystem-like namespace, based on the concept of directory.

The XRootD manager exhibits many important features:

- High transaction rate with rapid request dispersement (fast opens, low latency)
- Write once read many times processing mode
- Integrated in ROOT
- Server clustering (up to 262000 servers per cluster) for scalability, supports large number of clients from a small number of servers
- High WAN data access efficiency
- Able to manage in real time distributed replicas in real time
- No database requirements (no consistency, backup/recovery issues, high performance)

- Simple installation
- No 3rd party software needed (avoids complicated dependencies)
- Low administration costs

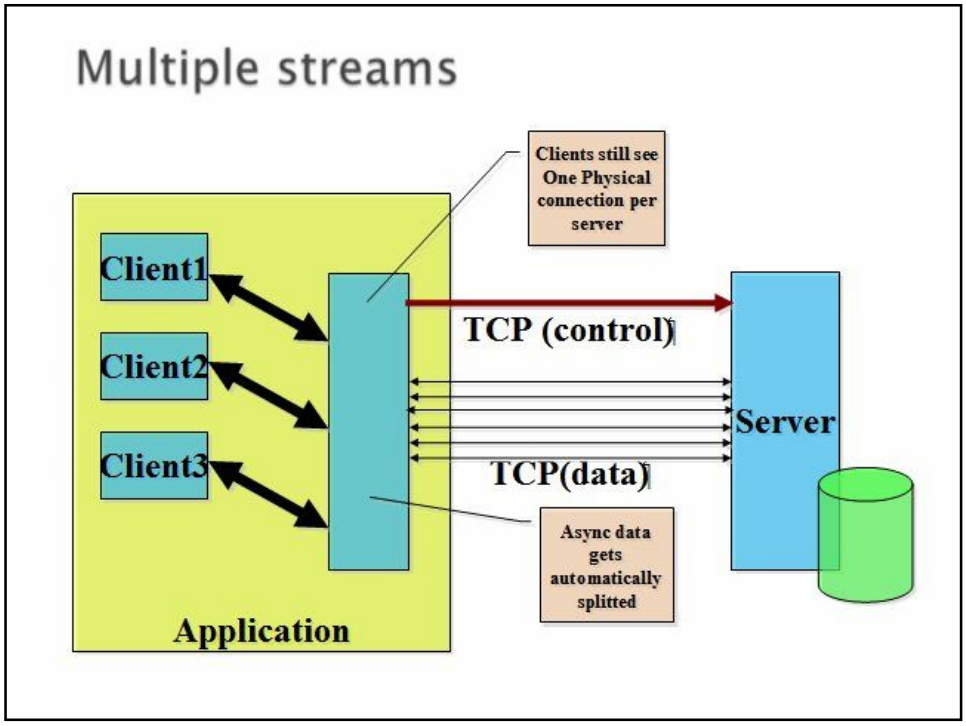


Figure 4. Data streaming from XRootD Storage Elements.

3. High Unix load

When a job requests processing of particular events included in different files stored at an XRootD SE, the data is streamed directly from the SE and only the requested events are read and processed on the fly. For this purpose, a number of sockets are opened, some of them inactive (see Figure 4). The number of open sockets can reach up to ~ 1000. The outgoing/read traffic from an SE is driven mainly by the (user) analysis jobs, thus the number of read requests can change chaotically.

In addition to the high number of open sockets, another feature showing up occasionally is a high Unix load. Normally, the high load happens together with a high throughput/traffic on the SE. The correspondence with the high number of open sockets is not so evident. Occasionally and irregularly, there occur special situations with an excessive load and very low/zero throughput (as the example in Figure 5).

With a careful configuration of the storage machines running XRootD, it is possible to substantially reduce the generation of too high load. In section 5, we describe an optimization procedure which we used to boost the performance of our XRootD storage machines, achieving also a substantial reduction of the Unix load on the optimized servers.

Recently, ALICE has launched a task to develop an extensive monitoring with MonALISA [7] to get more detailed information about the behavior of the XRootD servers and their communication with

running jobs. This is with the aim to find a way how to prevent the situations with a high load and a low throughput.

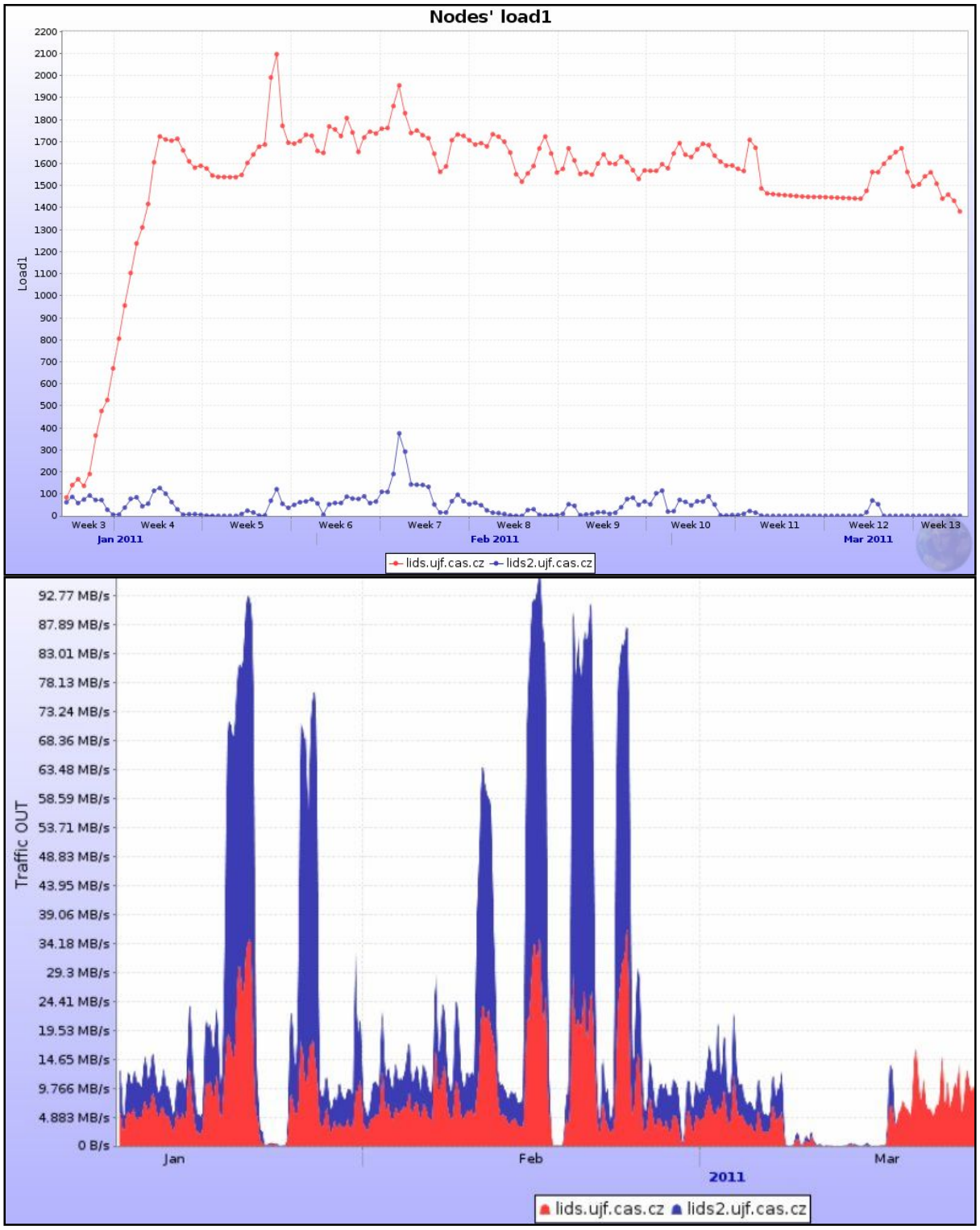
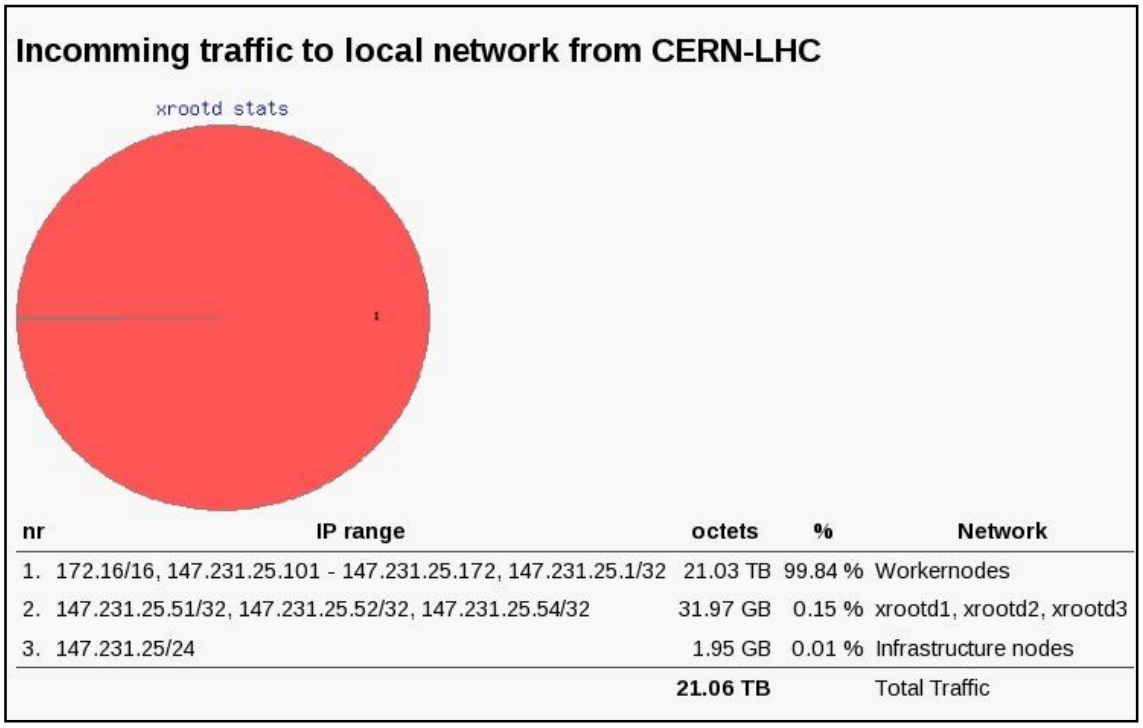


Figure 5. An example of a high load vs. low/zero throughput (red curve and red plot).

4. ALICE XRootD storage cluster at the WLCG Tier-2 site in Prague, Czech Republic

The computing and storage resources and services provided for ALICE by Czech Republic are installed at the regional computing center Golias [8] in Prague at the Institute of Physics (FZU) of the Czech Academy of Sciences (AS) and a part of the ALICE storage cluster is situated at the Nuclear Physics Institute (NPI) of the Czech AS. The Golias site provides services for HEP experiments including ALICE and also for some non-HEP projects. It operates more than 2 PetaBytes on disk servers (DPM, XRootD, NFSv3) and publishes ~ 3500 cores on WorkerNodes.



nr	IP range	octets	%	Network
1.	172.16/16, 147.231.25.101 - 147.231.25.172, 147.231.25.1/32	21.03 TB	99.84 %	Workernodes
2.	147.231.25.51/32, 147.231.25.52/32, 147.231.25.54/32	31.97 GB	0.15 %	xrootd1, xrootd2, xrootd3
3.	147.231.25/24	1.95 GB	0.01 %	Infrastructure nodes
		21.06 TB		Total Traffic

Figure 6 Daily traffic stats from CERN to XRootD and WotkerNodes for 2012-05-04.

The site is certified as a WLCG Tier-2 center (praguelcg2), the WLCG MoU was signed in 2008. It has excellent network connectivity: multiple dedicated 1-10Gb/s connections to collaborating institutes, national and worldwide.

The batch system used is Torque + MAUI scheduler, the WLCG services include:

- Grid UI, MonBox, gLite-CE, CREAM-CE
- SE: DPM (head with 15 DPM disk nodes), XRootD (3 client machines for ALICE)
- ALICE VOBOX

The use of virtualization at the site is quite extensive, most of the services are running on virtual machines. One of the on-site developed monitoring services aims to provide a better understanding of the efficiency of the local ALICE jobs. In an ideal situation, the jobs running at the site would use only the data located at the Prague XRootD storage cluster described below. The mentioned service

monitors the data traffic between CERN and the site Golias and identifies the data flow going to the storage machines, caused e.g. by the replication campaigns, and the data going directly to the WorkerNodes, i.e. requested by the jobs. As shown in Figure 6, the absolute majority of the data flow from CERN during that particular day was caused by the requests of the jobs.

The ALICE XRootD storage cluster in Prague (ALICE::Prague::SE) consists of 2 parts:

- 3 client machines at the site Golias
- redirector plus 3 clients at the collaborating (remote) institute - NPI

There is a dedicated E2E optical link 1Gb/s connecting the two clusters, see Figure 7, to be upgraded to 10Gb/s soon. The total capacity of the Prague cluster is about 538.7 TiB.

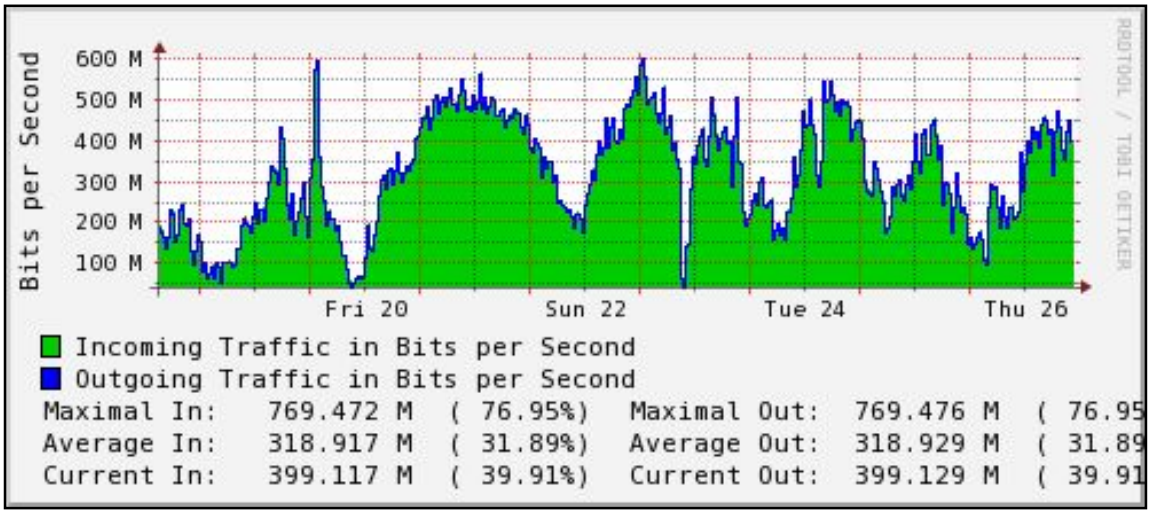


Figure 7. Weekly graph of traffic NPI <-> FZU (30 Minute Average).

5. Tuning procedure for a better performance and reduction of the load

The Prague XRootD cluster started in 2007 with one disk server at the NPI (already decommissioned). Gradually, 7 seven other disk machines have been set into operation (see the previous section). In 2011, we decided to perform an extensive tuning of the cluster motivated by remarkably different performance of the individual machines.

The optimization procedure was done as follows:

- data was migrated from the machine to be tuned to free disk arrays at another machine of the cluster.
- the migration procedure was done so that the data was accessible all the time.
- the empty machine was re-configured using RAID5, for its higher read data transaction rate. Before, the RAID6 setup was used.
- the number of disks in one array was reduced, resulting in more disk bundles than with the previous RAID6.
- the lower data protection level is compensated by an extensive local monitoring for a case of a disk failure.

- the raid controller cache was carefully configured.
- the readahead option was set to a multiple of (stripe_unit * stripe_width) of the underlying RAID array.
- no partition table was used to ensure proper alignment of the file systems: they were created with right geometry options ("-d su=X, sw=YYk" mkfs.xfs switches).
- mounting was performed with the noatime option.

As a result, the read/write performance of the machines was substantially enhanced, see an example in Table 1. Another result of the tuning was that the load on the re-configured servers dropped. In Figure 8, the read traffic during the period before and after the tuning is shown, in Figure 9 is the load on the machines during the same time.

	iozone-write	**iozone-read**	**ioapps-simul**
before tuning	**89 MB/s**	**22 MB/s**	**2799 s**
after tuning	**174 MB/s**	**398 MB/s**	**584 s**

Table 1. Parameters of one of the optimized XRootD servers before and after tuning.

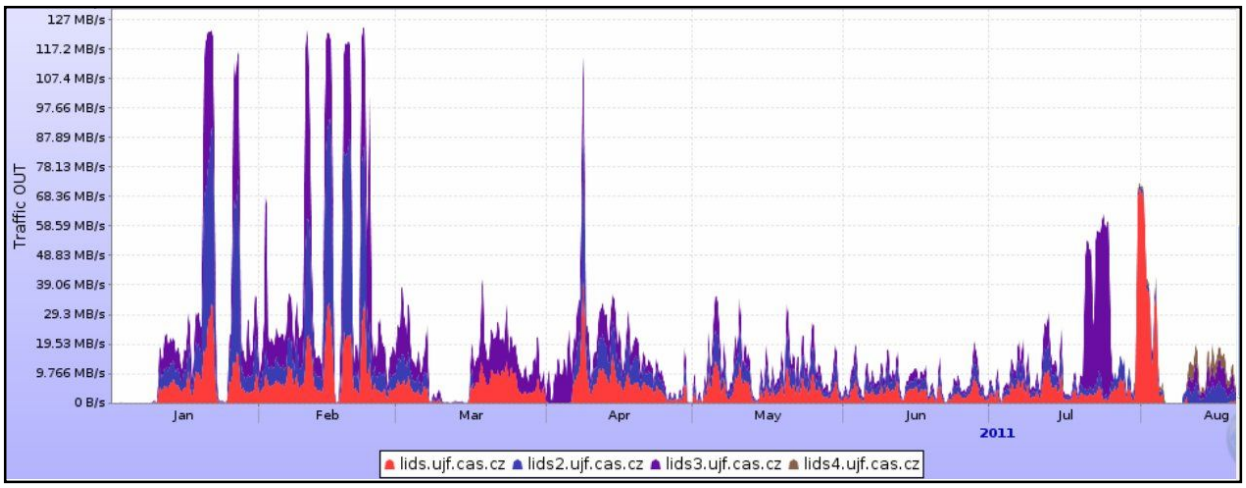


Figure 8. Outgoing traffic on the NPI XRootD machines 01.01.2011 - 01.09.2011.

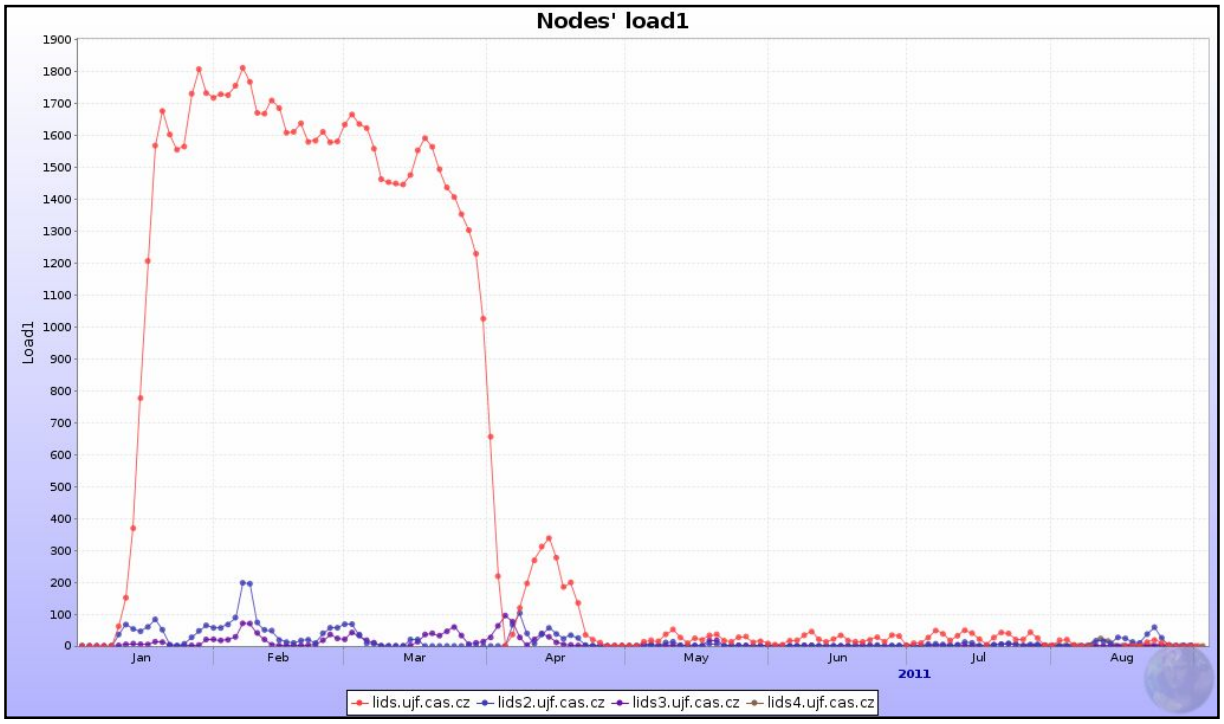


Figure 9. Load on the NPI machines 01.01.2011 - 01.09.2011. Tuning was done after April 2011.

In the time of writing this article we decided to perform an additional test with a tuning of other XRootD disk servers' parameters, namely the readahead and the I/O scheduler. For each disk array on a machine, we set the readahead to a multiple of (number_of_data_disks * stripe_size) for each array so that the final value was ~ 1MiB (compared to the default of 128 KiB). The I/O scheduler was set to 'noop' value instead of the default 'cfg' value. For the default and optimized parameters, we performed simulated XRootD-like-reading tests with data files generated artificially in each of the disk arrays. These tests, however, had to be done during the full operation of the XRootD machines, at this time we had no free space where to migrate the data. An example of the test on one of the servers is shown in Table 2. By the optimization of the selected parameters, the reading of files got about 25% faster.

filesystem	**default runtime (s)**	**optimized runtime (s)**
fs #1	**4087.5**	**3225.1**
fs #2	**3852.3**	**3008.2**
fs #3	**2461.4**	**1788.2**
fs #4	**3965.6**	**3050.8**
fs #5	**3977.7**	**3058.3**

Table 2. Times needed to read generated files (in seconds) before and after tuning of parameters.

The performance improvements shown during synthetic tests were better than shown during the real XRootD operations. We ascribe this behavior to the read pattern we observe on our machines. A typical read pattern is shown in Figures 10-12 for one particular file. It has features of a random access pattern.

Figures 11 and 12 represent details of Figure 10. The x-axis in Figure 10 shows the offset of the file being read and on the y axis is a time since the opening of the file. Although it might not be visible at the first glance, the picture is composed of about 20000 small lines (= number of read() calls). This is better shown in Figure 11, which is a detail of figure 10. It looks like the read() operations use 3 different parts of the file. The read() requests are relatively small, mostly less than 3 kB, as shown in figure 12. If reading with this pattern is happening for many (several hundreds) files at the same time, the disk system must inevitably be completely saturated.

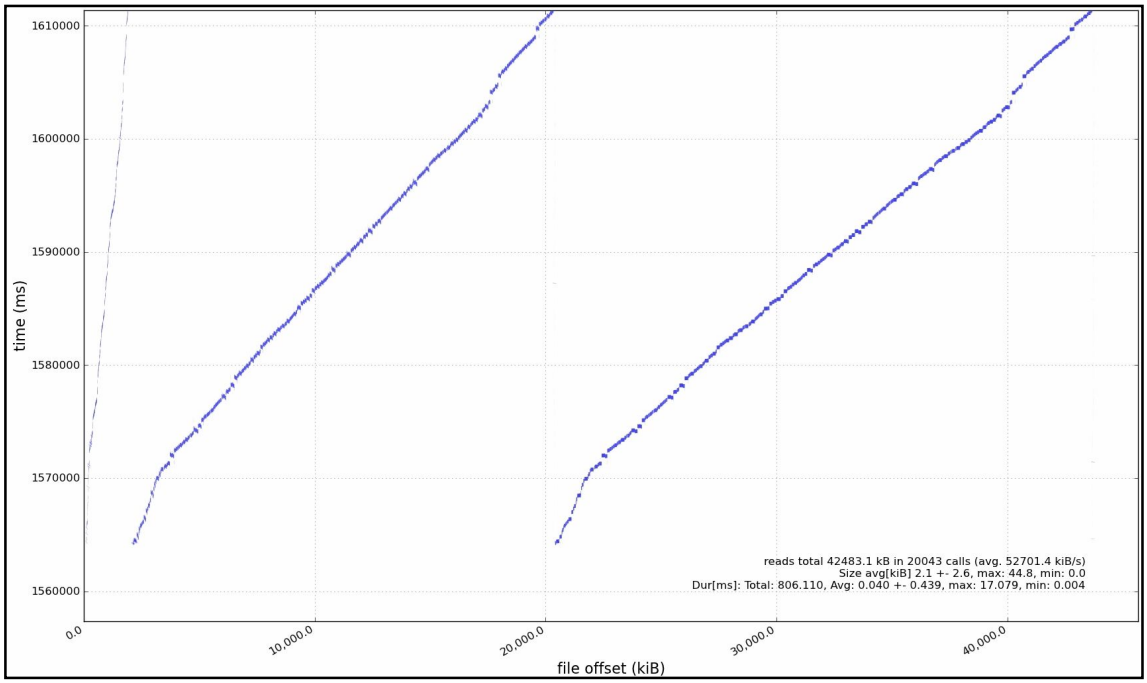


Figure 10. The course of a file reading.

6. Summary and concluding remarks

In this paper, we described in detail the optimization procedure we completed to enhance the performance of our XRootD cluster. The tests done after the first stage of the tuning, during which the data from the servers under re-configuration was temporarily migrated elsewhere, shown a remarkably high read/write performance improvement: up to 20 times better values for the 'iozone-read'. Another result of this tuning was a reduction of the Unix load on the re-configured servers.

In addition to this extensive optimization performed on the machines with the data temporarily migrated elsewhere, we decided to do more tuning - of other disk servers' parameters. This tuning and the tests of its results had to be done during the full operation of the machines. The tuning resulted in about 25% speed-up in the file reading tests.

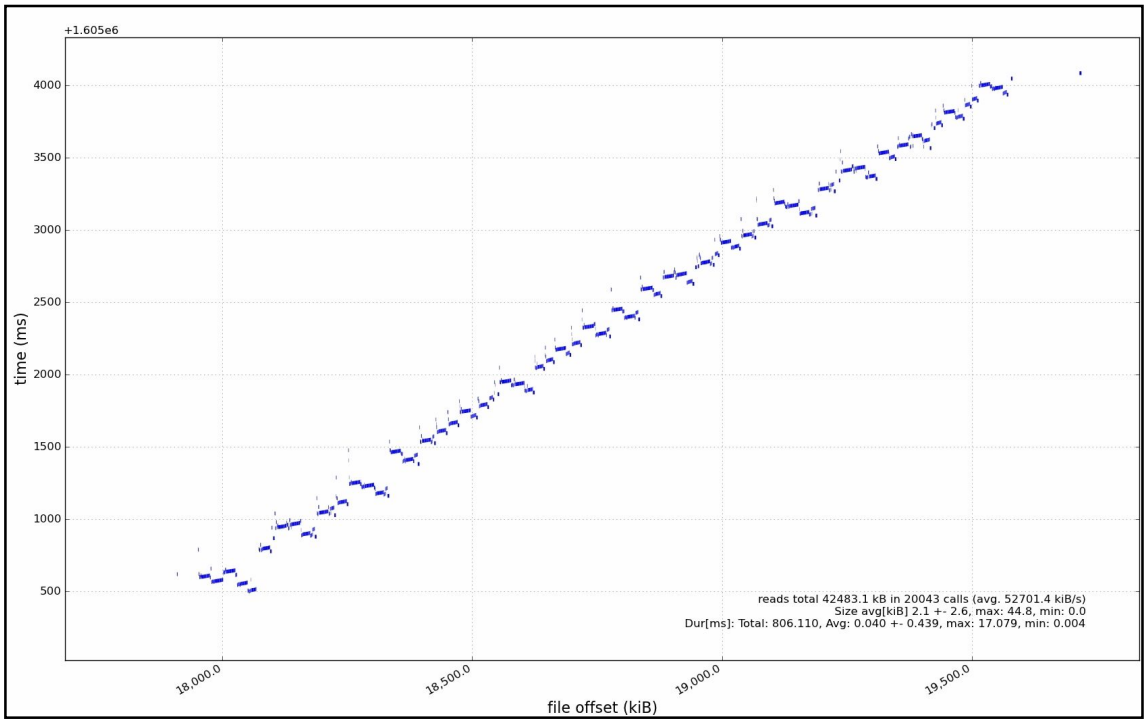


Figure 11. Detail of the read file procedure.

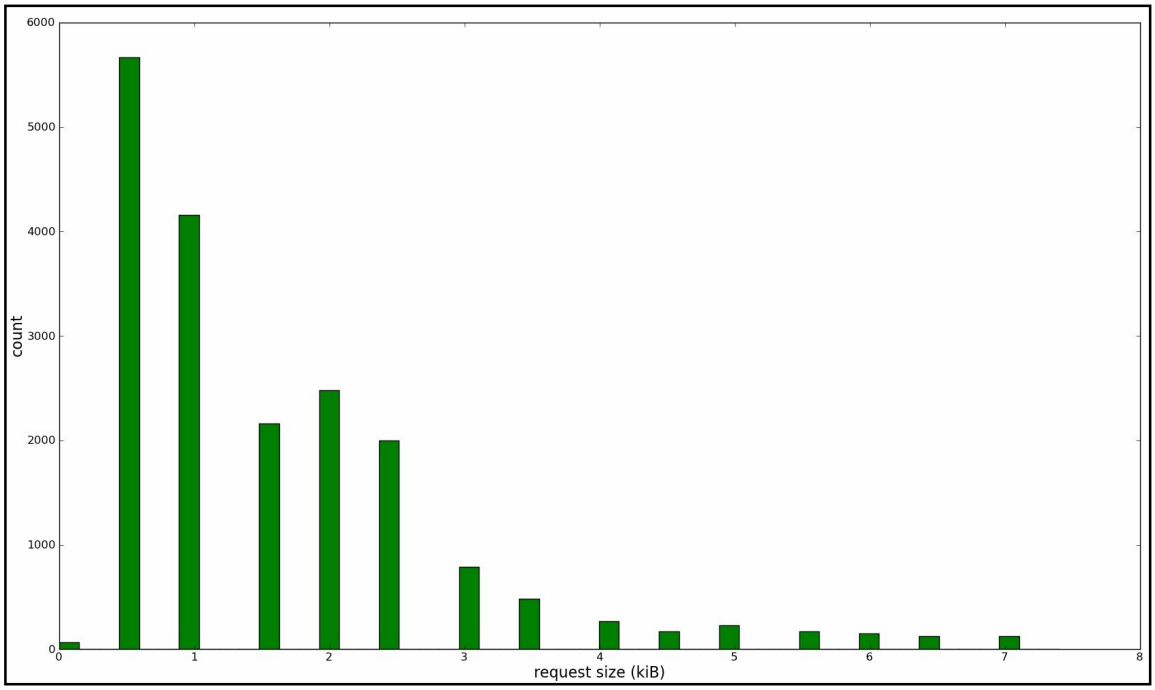


Figure 12. The individual read() requests are small.

The synthetic tests of the performance improvements shown better results than what we have observed during the real XRootD operations. We explain this behavior as a consequence of the XRootD random access read pattern. We suppose that e.g. usage of the ROOT TTreeCache and optimized ROOT baskets during the file creation process would optimize the read behavior and boost the reading speed.

In addition to the improvement of the read/write performance of our optimized XRootD machines, the tuning/optimization produced another clearly positive effect: the load on the optimized machines was in general reduced and the ratio between the throughput and the load on the machines got increased. Although the load still might get to ~ 100 in peaks, the servers are remarkably stable and the situations with high Unix load and low/zero throughput disappeared, see Figures 13 and 14.

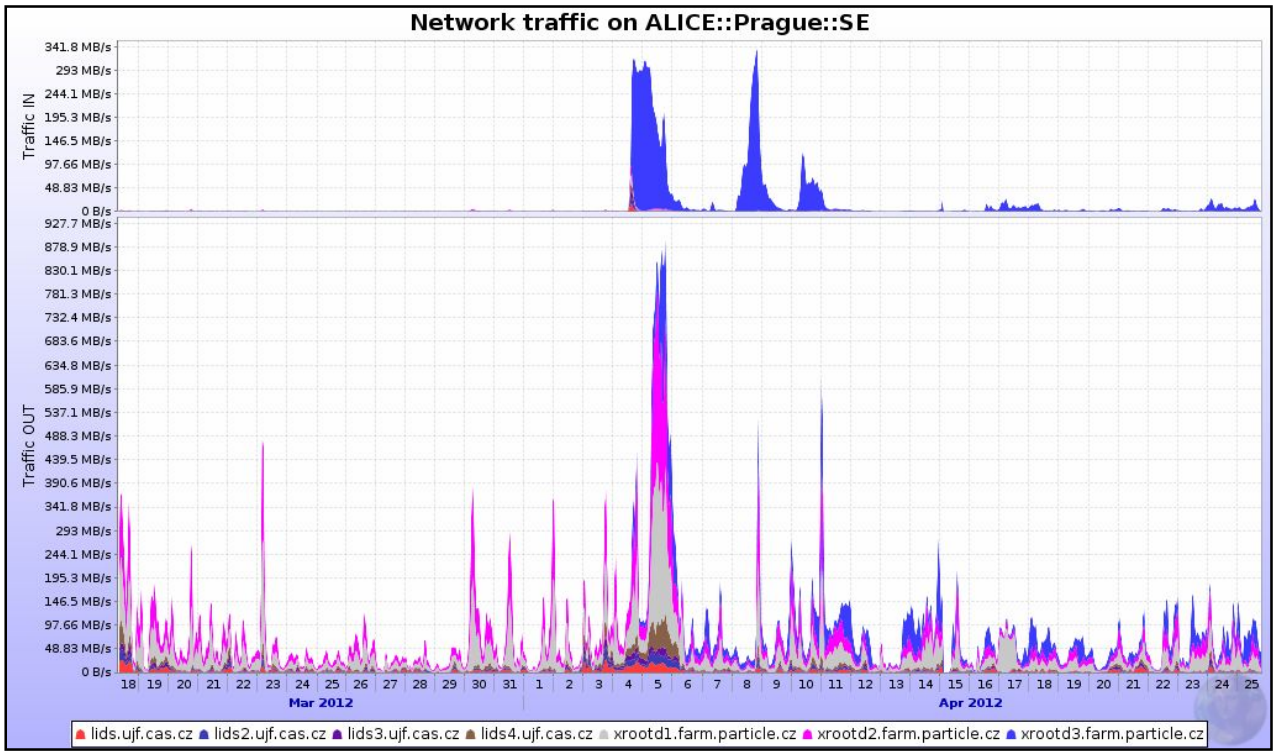


Figure 13. Traffic on our XRootD machines 18.03.2012 – 25.04.2012.

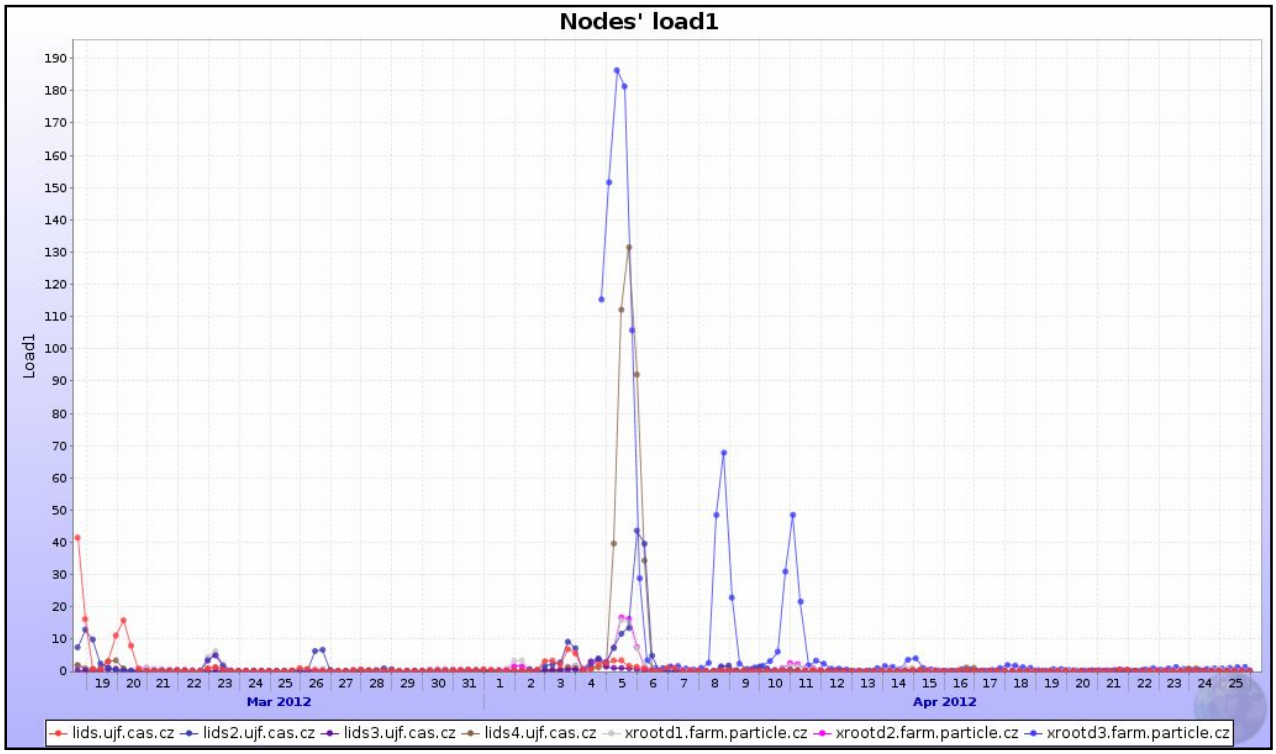


Figure 14. Load on our XRootD machines 18.03.2012 - 25.04.2012. The peaks correspond to a higher traffic.

Acknowledgements
The work was supported by the MSMT CR contracts No. 1P04LA211 and LC 07048.

References

[1] ALICE Collaboration: http://aliceinfo.cern.ch/Public/Welcome.html.

[2] XRootD,http://project-arda-dev.web.cern.ch/project-arda-dev/xrootd/site/index.html, http://xrootd.slac.stanford.edu/.
Scalla Cluster Management Service, http://xrootd.slac.stanford.edu/doc/prod/cms_config.htm.

[3] Brun R and Rademakers F 1997 ROOT: An object oriented data analysis framework,

Nucl. Instrum. Meth. **A389** 81; http://root.cern.ch/.

[4] ALICE Experiment Computing TDR, http://aliceinfo.cern.ch/Collaboration/Documents/TDR/Computing.html.

[5] Gunther N J *UNIX Load Average. Part 1: How It Works*, http://www.perfdynamics.com/Papers/la1.pdf.

[6] Worldwide LHC Computing Grid: http://public.web.cern.ch/public/en/lhc/Computing-en.html.

[7] Legrand I, Monitoring *and Control the xrootd servers in ALICE,* https://indico.cern.ch/getFile.py/access?contribId=50&sessionId=2&resId=0&materialId=slides&confId=175968.

[8] Regional Computing Center for Particle Physics, http://www.particle.cz/farm/public.aspx.