DATA MANAGEMENT FOR ALICE, ATLAS AND VOCE IN THE CZECH REPUBLIC

L. Fiala, J. Chudoba, J. Kosina, M. Lokajicek, J. Svec [1,2], J. Krasova [1], J. Kmunicek, D. Kouril, L. Matyska, M. Ruda, Z. Salvet [2,3], M. Mulac [2,4]

[1] Institute of Physics, Academy of sciences of the Czech Republic
[2] CESNET, z.s.p.o.
[3] Institute of Computer Science, Masaryk University
[4] CIV, University of West Bohemia in Pilsen

Abstract:
We will present our experiences with grid data management for the purposes of Virtual Organizations ALICE, ATLAS and VOCE (general-purpose VO for Central Europe), with particular focus on transfers via FTS and issues with DPM installation and configuration.

1. Overview

In the scope of LCG[1] and EGEE[2] projects, the Czech Republic is supporting the following EGEE Virtual Organizations, allowing them to use computational and storage resources: ATLAS[3], ALICE[4] and VOCE[5]. The ATLAS and ALICE Virtual Organizations are corresponding to HEP experiments at LHC.

On the other hand, the main purpose of VOCE Virtual Organization is to provide already existing Grid facilities and infrastructure to scientific fields other than HEP, inside the Central Europe region (this is where the name comes from – VOCE stands for Virtual Organization for Central Europe) and to show new scientific communities how Grid could be useful for their work and research. Currently Austrian, Czech, Hungarian. Polish, Slovak and Slovenian resources are involved, and the design and implementation of the whole infrastructure (information systems, file catalogues, workload management systems) was done solely on Czech resources, using the LCG and gLite[6] middleware components.

2. Storage

As the Classical Storage Elements (in terms of LCG middleware) are being phased out, the decision regarding the storage system software to be used had to be made. As we are participating in Service Challenge 4, the need for a Storage Element which will provide a SRM [7] interface for accessing the data suddenly arose. Currently there are three Storage Elements which are supporting the SRM file access, namely **CASTOR** [8], **dCache** [9] and **DPM** [10]. The first two are mainly designed for grid-enabled file access to tape storage (also allowing file storage on disk pools, although this is not their primary aim). In contrast to this, DPM storage element is designed as a light-weight replacement both for CASTOR and dCache (both of them being quite complex and requiring non-trivial amount of configuration and management effort). DPM targets only those setups which do not have any tape storage, but the disk space (possibly collected from many distinct disk servers) is going to be used to store the data and provide them to users.

Currently, we do not provide any tape storage for LCG/EGEE Grid, so the DPM was the choice. Our current setup involves 1 head node and 1 disk server, which run both on the same physical machine. Even though this is not the recommended setup, we were able to gain reasonable performance, as will be demonstrated later.

The disk space provided consists of 4 filesystems, with 5 TB of total storage. Three of the filesystems are local, the fourth one is mounted over the network using the NBD [11] concept. The rationale for choosing NBD instead of NFS will be explained later in this article.

3. DPM issues

The deplyoment of the DPM was not without problems, the most serious we encountered are presented and discussed in the following sections, together with workarounds we used to make the whole concept usable.

3.1. srmCopy()

The problem we have faced almost immediately after installation and configuration of DPM was the fact that the current implementation supports only a subset of a SRM version 2.2 specification [12]. Probably the most notable function that is prescribed by the specification and not implemented by the DPM is `srmCopy()`. This function is used whenever 3rd party transfer takes place (i.e. the case in which the data are copied directly between two SRM-enabled storage interfaces, without first getting the data to the user's local disk and then putting them on the destination Storage Element).

This issue can however be solved quite easily (until the support for this method is implemented by DPM), as was described in [13]—in case when copying the data from non-DPM SRM Storage Element to DPM Storage Element, the pushmode=true flag must be used to workaround this problem. On the other hand, local temporary storage or direct transfer using `globus-url-copy` can be used to completely avoid direct SRM-to-SRM 3rd party copy using `srmCopy()` function.

3.2. Pools on NFS

Our original plan was to provide the space for DPM server through NFS protocol, from our NFS disk-array server. NFS server is 64bit Opteron, running 64bit version of 2.6 Linux kernel, providing the configuration needed to make our hardware to work properly.

However, this setup (having DPM pools located on NFS mounted partitions), didn't turn out to work well—we have experienced silent truncations of files transferred from server to client. There was no error message emitted during the transfer. We contacted DPM developers, but they were not aware of this problem. The `strace` debugger showed us the following (only the part relevant to the file copying operation is shown):

```
alarm(1200)                 = 1200
gettimeofday({1141997636, 424588}, NULL) = 0
read(13, "e\364\354a\256#\214=\243\31=\273\3332\234=3\30\316\356"..., 65536)
= 65536
gettimeofday({1141997636, 424753}, NULL) = 0
gettimeofday({1141997636, 424801}, NULL) = 0
select(15, [4], [14], [], {45, 905900}) = 1 (out [14], left {45, 900000})
gettimeofday({1141997636, 427129}, NULL) = 0
write(14,"\37\334\214\276?\270y)9\237\203z\344S\217|\356#\237\373"...,
65536) = 37648
write(14,"\224\257\264F\253M\v\211\210|\224\233\26BmB\310NC\202\20"...,
27888) = -1 EAGAIN (Resource temporarily unavailable)
gettimeofday({1141997636, 427377}, NULL) = 0
gettimeofday({1141997636, 427424}, NULL) = 0
select(15, [4], [14], [], {45, 903277}) = 1 (out [14], left {45, 910000})
gettimeofday({1141997636, 430129}, NULL) = 0
write(14,"\224\257\264F\253M\v\211\210|\224\233\26BmB\310NC\202\20"...,27888
) = 27888
rt_sigaction(SIGALRM, {0x807220f, [ALRM], SA_RESTORER|SA_RESTART, 0x92f0c8},
{0x807220f, [ALRM], SA_RESTORER|SA_RESTART, 0x92f0c8}, 8) = 0
```

This algorithm repeats for every 65 kilobyte block of the file (data are read from NFS server and then sent to the network file descriptor).

The last block (which is not last in fact in the file, but last in the connection—and then the file is truncated), however looks like this:

```
alarm(1200)                 = 1200
gettimeofday({1141997636, 449720}, NULL) = 0
read(13, 0x9c642d8, 65536)  = -1 EACCES (Permission denied)
gettimeofday({1141997636, 450174}, NULL) = 0
gettimeofday({1141997636, 450221}, NULL) = 0
select(15, [4], [14], [], {45, 880480}) = 1 (out [14], left {45, 890000})
gettimeofday({1141997636, 450356}, NULL) = 0
write(14,"\336\3440\331\244\22\303E|Z:\27?!K\377\tsA\313\247\245"..., 65536)
= 36200
write(14,"\3532\220\27\36-!e\374\365\35\0036\202--Z\32\342\270tF"..., 29336)
= -1 EAGAIN (Resource temporarily unavailable)
```

As can be observed on the bold line, the `read()` system call returned `EACCES` error for the very same file descriptor (number 13, in our demonstrated case), on which previously many `read()` calls succeeded and there was no system call in between (like `ioctl()`, `close()`, etc.) which would cause the change of the file descriptor state.

This problem occurs only when the file is located on the NFS mount, and we were not even able to reproduce it using standard Unix utilities like `/bin/cp`, `/usr/bin/dd`, simple test C programs just performing read()/write() operations (we were using the same block sizes as DPM did when received the error, i.e. 65536 bytes). We also found out that the problem occurs only in cases when 2.4-kernel based NFS client (in the role of DPM disk server) accesses 2.6-kernel based NFS server. 2.6-2.6 and 2.4-2.4 transfers worked well (we didn't test the variant with 2.6-kernel based NFS client accessing 2.4-kernel based NFS server). The cause of this problem is not yet fully understood, as it is quite difficult to trigger (so far only DPM

is able to trigger the conditions provoking this bug, we were not able to make kernel to return `EACCES` in this situation by any other means). In addition to this, we also verified that this problem occurs also with new VDT [15] 1.3 package, which is soon to be incorporated into the LCG middleware release (currently 1.2 is used).

The workaround solution we used was replacing NFS with NBD, which is the solution we use for providing the partition from the disk array to the DPM disk server. This setup works well and no file truncation happens any more. It has to be explicitly pointed out, that DPM requires every filesystem in the pool to be located on separate partition, due to the way how DPM calculates and updates the internal free space counter. This is fine with NBD, as NBD doesn't allow for sharing the same block device to multiple clients (which is the disadvantage of NBD compared to NFS).

3.3. Rate limiting

We came across this problem when our DPM Storage Element was under a high load of data transfers (mainly in the scope of Service Challenge 4). The current SRM server daemon implementation contained in DPM doesn't support rate limiting of possible concurrent new SRM requests (dCache and CASTOR2 storage systems provide this functionality).

We were informed by the DPM developers that this is a planned feature for one of the coming releases.

4. Performance in the real life tests

We tested our setup in various performance tests. We had to use the same server to support ATLAS production, therefore conditions for different tests could vary. For monitoring we used CERN FTS [16] server monitoring GRIDVIEW [17], and IP accounting (ipac) package [18] installed on our DPM server. The first available FTS channel to our DPM server was setup from CERN. We demonstrated that our server can accept data for several hours at the level reaching 50 MB/s (see Figure 1).

Tests using srmcp command showed the advantage of FTS. A bulk transfer of 10 files between Prague and SARA (Tier1 centre in Netherlands) fully succeeded, a transfer of 200 files mostly failed with an error "Too many transfers". FTS server can limit number of simultaneous connections per channel and protect SRM storage elements from overloads.

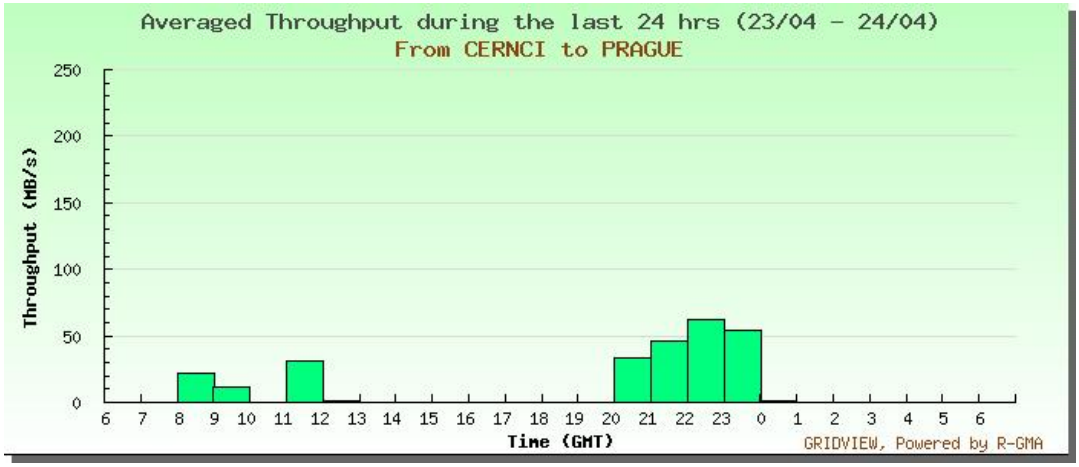

Figure 1: 3 tests of transfers via FTS were done during a weekend when there were no other activities on the DPM server but CERN server was fully used for Service Challenge tests to Tier-1 centres. During the last test 700GB were transfered within 4 hours, additional 25% of transfers failed.

ATLAS computing model associates smaller Tier2 centers to a predefined Tier1 centre, where simulated data from Tier2 are stored on tapes. Prague Tier2 at Institute of Physics (FZU) is associated to Forschungszentrum Karlsruhe (FZK) in Germany. A shared 1 Gbps connection was available during 2006. Data transfer tests indicated that limitation on transfer speed does not come from a network bandwidth but from disk server performance. Figure 2a shows that we could achieve almost 50 MB/s when disk servers at Tier1 were not overloaded, figure 2b shows the same transfers when disk servers were simultaneously used for other transfers and their load increased. A similar behavior is observed when DPM server is heavily loaded with by many file retrievals.

5. Conclusion

We used a DPM server running on 1 machine to support data transfers for ATLAS production and for tests in the scope of Service Challenge 4. For several days the server sustained continuous rate of data export above 400GB/day with a peak of 1.5 TB/day (Figure 3). During intensive data transfer tests we observed performance bottlenecks but not always we could uniquely assign them to the DPM server. It was not possible to have DPM pools located on NFS-mounted filesystem. This problem has been reported and consulted with DPM developers, but exact cause was not yet fully understood.

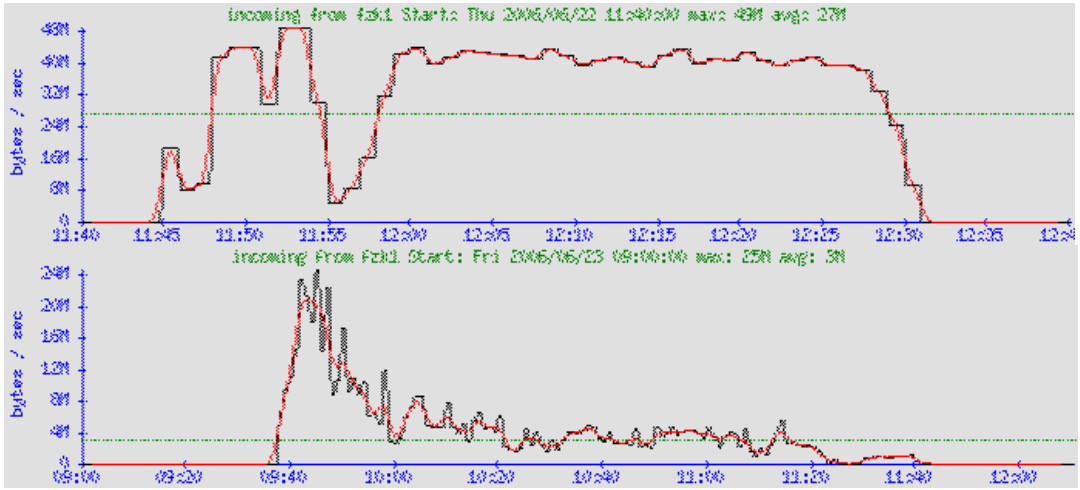

Figure 2: a) A time behavior of transfer of 100 GB from FZK to FZU when disk servers are not overloaded. b) The same transfer fails when the load increases, many errors due to "transfer time-out" occur. Plots were obtained by IPAC package [18].

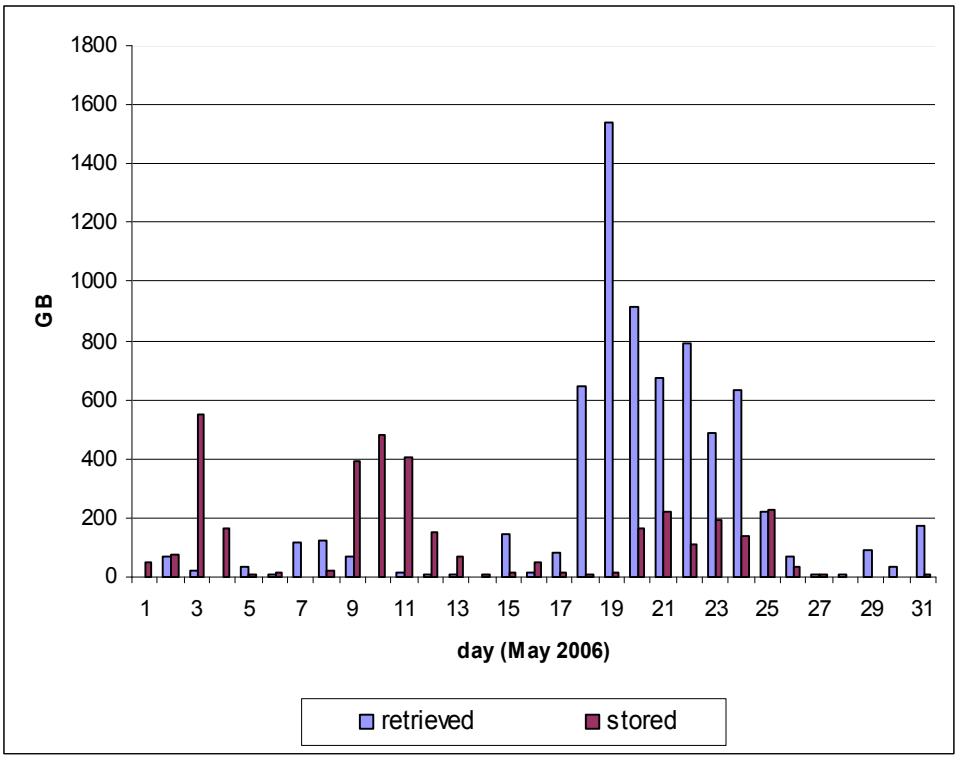

Figure 3: Data volumes transferred to and from the DPM server golias100 during May 2006.

References

[1] Worldwide LHC Computing GRID, http://cern.ch/lcg/
[2] Enabling Grids for E-sciencE, http://cern.ch/egee/
[3] The ATLAS Experiment, *http://cern.ch/atlas/*
[4] A Large Ion Collider Experiment at CERN LHC, *http://aliceinfo.cern.ch/*
[5] VOCE Virtual Organization, *http://egee.cesnet.cz/en/voce/*
[6] gLite – Lightweight Middleware for Grid Computing, *http://cern.ch/glite/*
[7] Storage Resource Management Working Group, http://sdm.lbl.gov/srm-wg
[8] CERN Advanced Storage Manager, http://castor.web.cern/ch/
[9] dCache Project, http://www.dcache.org
[10] DPM, http://twiki.cern.ch/twiki/bin/view/LCG/DataManagementDocumentation/
[11] The Enhanced Network Block Device Kernel Module, http://enbd.sf.net/
[12] SRM v 2.2 Specification, http://sdm.lbl/gov/srm-wg/DOC/SRM.v2.2.pdf
[13] James Casey, DPM and SrmCopy,
http://twiki.cern.ch/twiki/bin/view/LCG/DpmAndSrmCopy/
[14] CERN Linux Pages, http://linux.cern.ch/
[15] Virtual Data Toolkit, http://vdt.cs.wisc.edu/
[16] FTS server, *http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/FTS/download.htm*
[17] GRIDVIEW, http://gridview.cern.ch/GRIDVIEW/
[18] IP Accounting for linux, *http://sourceforge.net/projects/ipac*