HEPiX Storage Working Group
- progress notes 3.2010 -

Andrei Maslennikov

September 14, 2010 – Plzen

Summary

- What is this this group and what it does
- Storage situation at HEP
- Storage laboratory at KIT
- Some recent numbers
- Discussion

HEPiX Storage Working Group

- The HEPiX Storage Working Group was established in 2006. It monitors the available data archival and access technologies with a special accent on the data types specific for High Energy Physics (HEP).
- The group does regular assessments of data pools in the laboratories and institutions like CERN, FNAL, BNL, INFN, IN2P3, DESY, RAL and others, and performs various tests using the realistic test applications typical for HEP. Our current laboratory is set up at the Karlsruhe Institute of Technology (KIT).
- The group currently has 27 people on the list, but not all of them are active all the time.

Credits for the late period

- The test laboratory at KIT was built on the top of hardware kindly provided by Karlsruhe Institute of Technology (rack and network infrastructure, load farm) and E4 Computer Engineering (new disk server). CERN had contrubuted with some funds to cover a part of human hours.

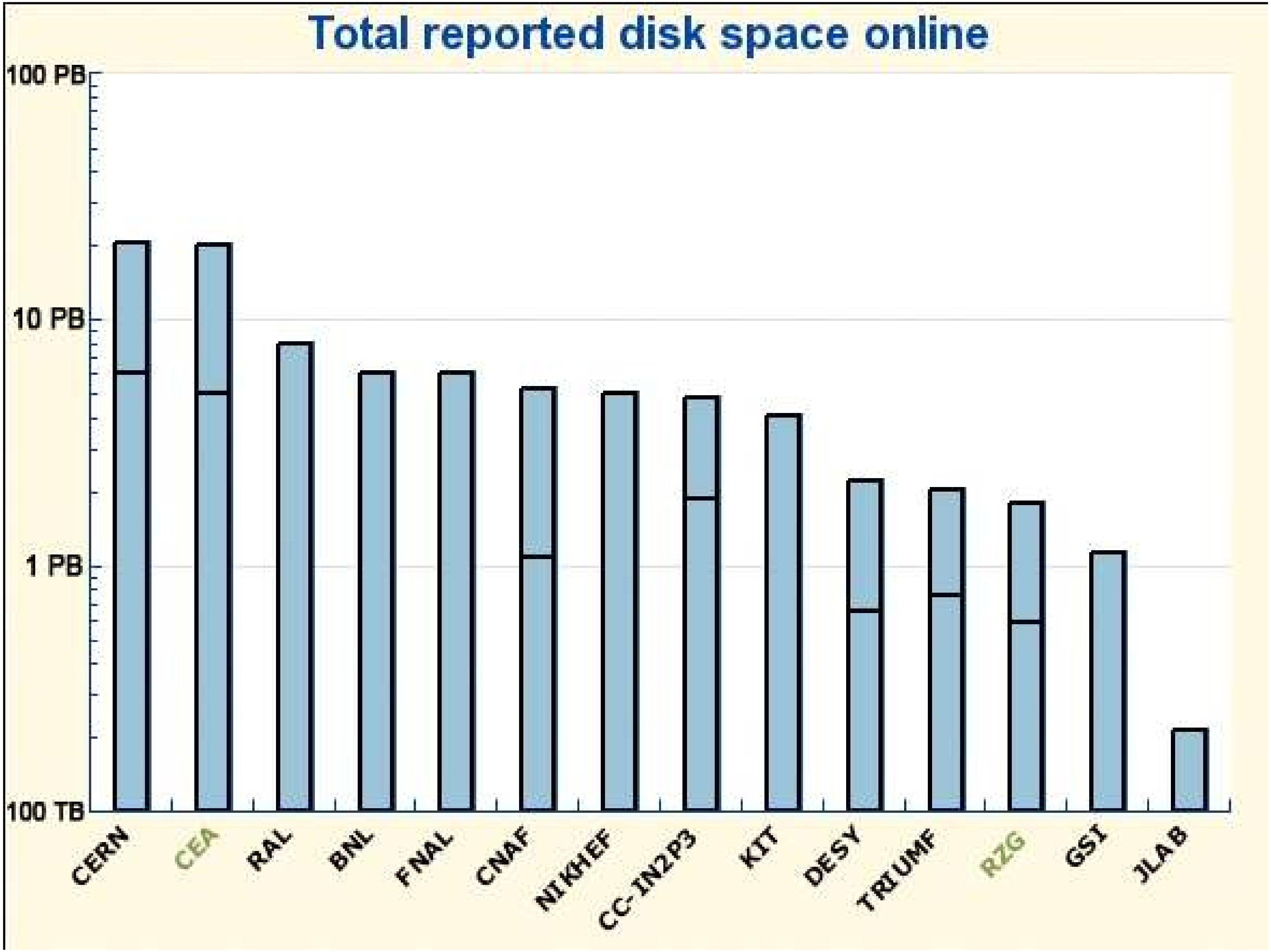
- These people participated in provisioning, funding, discussions, laboratory building, preparation of test cases and test framework, tests and elaboration of the results:

CASPUR	A.Maslennikov (Chair), M.Calori (Web Master)
CEA	J-C.Lafoucriere
CERN	B.Panzer-Steindel, D. van der Ster, R.Toebbicke
DESY	M.Gasthuber, P.van der Reest, D.Ozerov
E4	C.Gianfreda
INFN	G.Donvito, V.Sapunenko
KIT	J.van Wezel, A.Trunov, M.Alef, B.Hoeft
LAL	M.Jouvin
RZG	H.Reuter

Storage Situation at HEP (Feb 2010)

Storage Questionnaire 2010

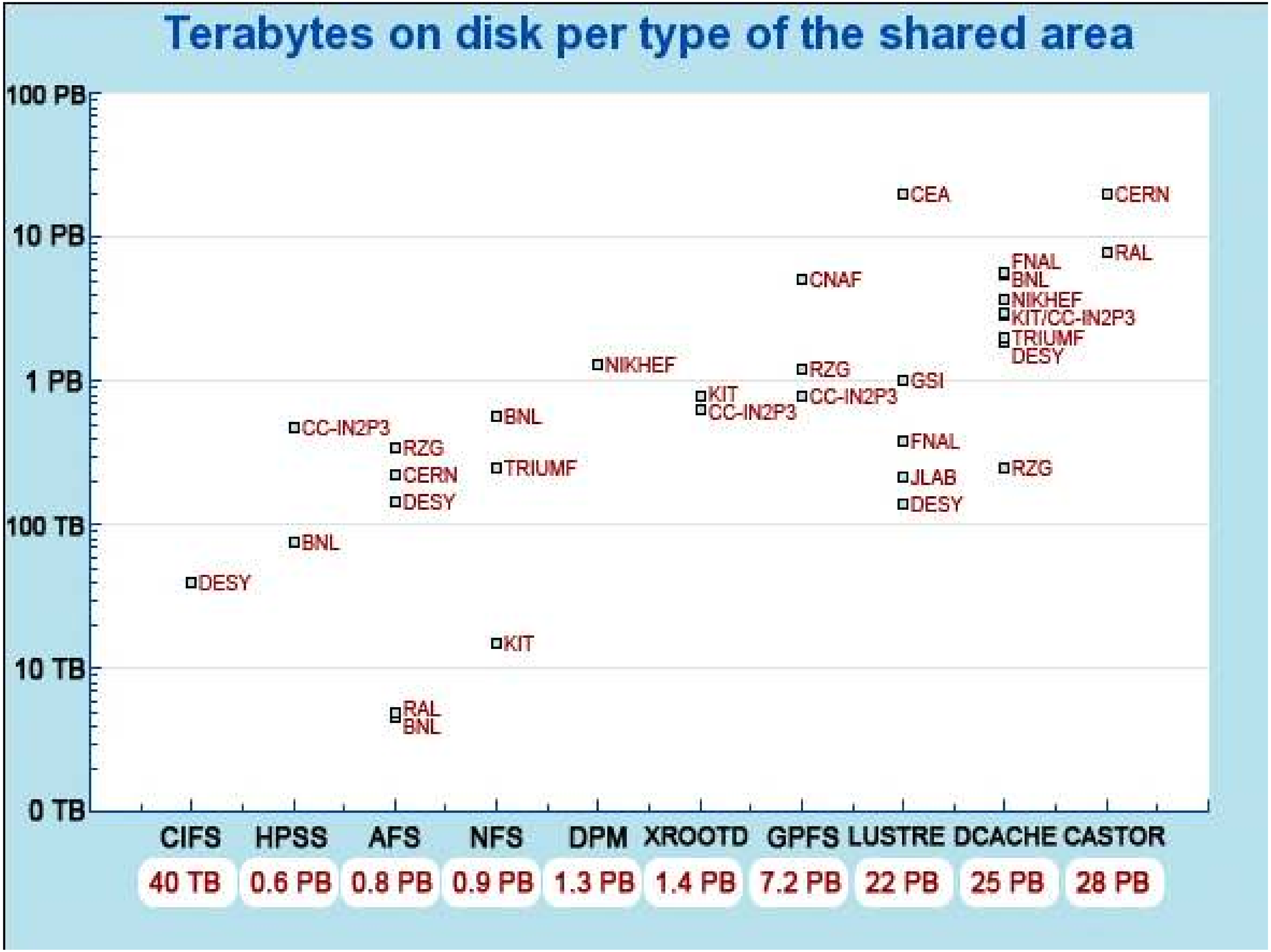
- **The 14 participating sites were mainly of the HEP origin, CEA and RZG being the only exceptions. The total described space online summed up to 87 PB, to be compared with 14 PB reported in 2007.**



Total reported disk space online
100 PB
10 PB
1 PB
100 TB
CERN
CEA
RAL
BNL
FNAL
CNAF
NIKHEF
CC-IN2P3
KIT
DESY
TRIUMF
RZG
GSI
JLAB

Some facts

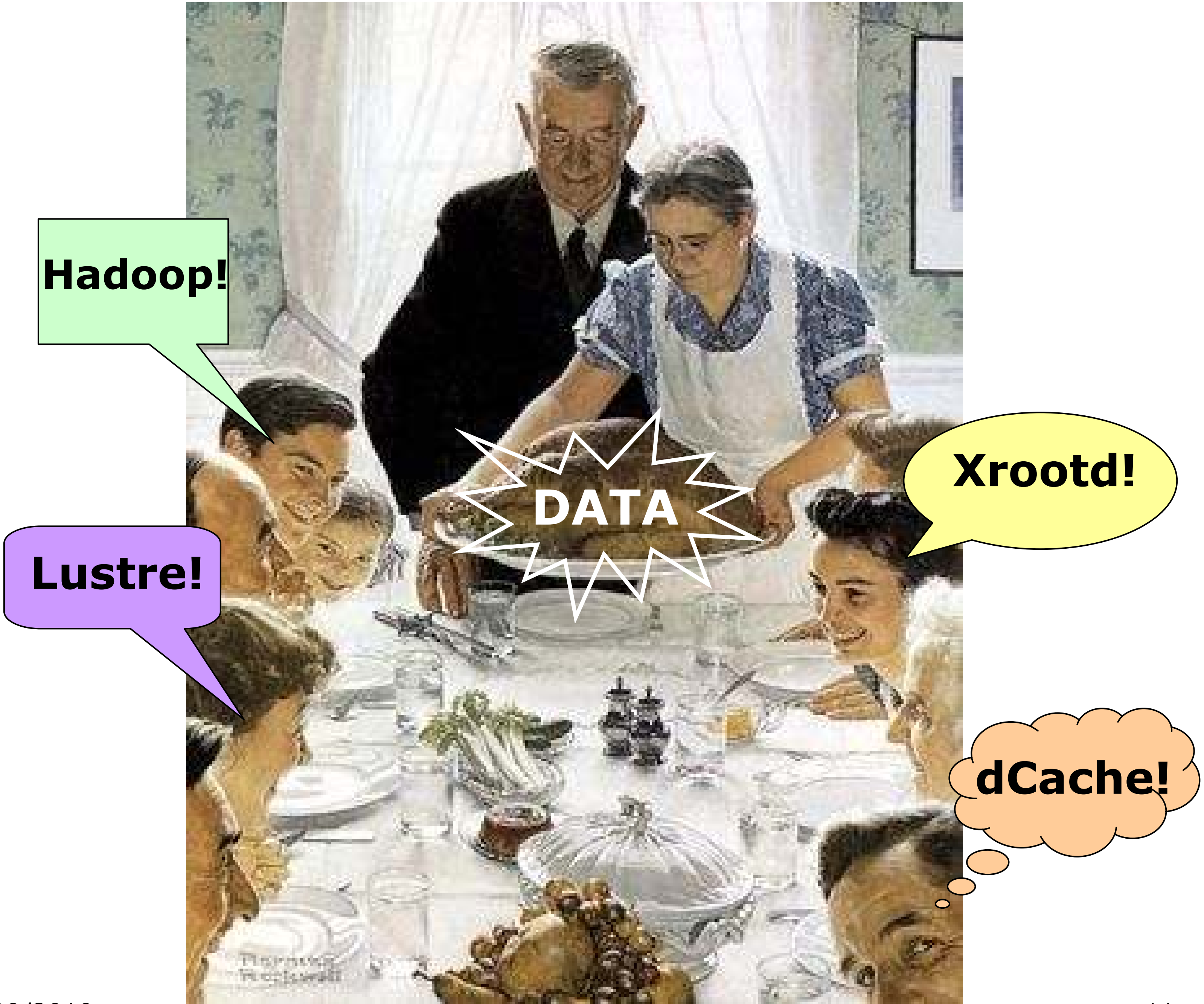
- Roughly one third of reported storage is in CASTOR, another third is in dCache and the remaining third is inside the shared file systems. CASTOR is only used at CERN and RAL, whereas dCache is in use at 8 sites out of 14.
- In 2007, no HEP data were stored inside Lustre. Today it is accounting for 50% of the shared file system space (another 50% is in GPFS). The shared file systems currently hold around 20% of HEP data, but they are visibly acquiring ground (new Lustre areas at GSI, DESY and FNAL etc). The recent migration from CASTOR to GPFS/STORM at CNAF demonstrated the feasibility of a large WLCG compatible archive built on the top of a shared file system.
- Currently observed ratio N-of-clients/N-of-servers oscillates around 10, over all participating HEP sites. Servers are still mostly with 1G outlets, so this ratio will likely be growing towards 50-90 for 10G based servers.



Terabytes on disk per type of the shared area
100 PB
10 PB
1 PB
100 TB
10 TB
0 TB
DESY
CC-IN2P3
BNL
RZG
CERN
DESY
RAL
BNL
BNL
TRIUMF
KIT
NIKHEF
KIT
CC-IN2P3
CNAF
RZG
CC-IN2P3
CEA
GSI
FNAL
JLAB
DESY
FNAL
BNL
NIKHEF
KIT/CC-IN2P3
TRIUMF
DESY
RZG
CERN
RAL
CIFS
HPSS
AFS
NFS
DPM
XROOTD
GPFS
LUSTRE
DCACHE
CASTOR
40 TB
0.6 PB
0.8 PB
0.9 PB
1.3 PB
1.4 PB
7.2 PB
22 PB
25 PB
28 PB

Observations

- So far, there seems to be no universally accepted data archival method for HEP data and situation continues to remain rather non-uniform. This non-uniformity in many cases has historical roots, and is often promoting a sane technological competition. However, one should never forget that all HEP sites have to deal with data of the same type and with similar access patterns.
- In this light, and in the view of the permanent growth of data volume, it is becoming more and more clear that a regular, methodical monitoring and comparison of TCO, reliability and efficiency of data archival and access solutions is and will be remaining a priority for HEP community.


Hadoop!
Lustre!
DATA
Xrootd!
dCache!

Storage Laboratory 2010

Current goals

- As in the previous years, we aim at the performance comparison of most diffused storage solutions (AFS, GPFS, Lustre, dCache, Xrootd etc)

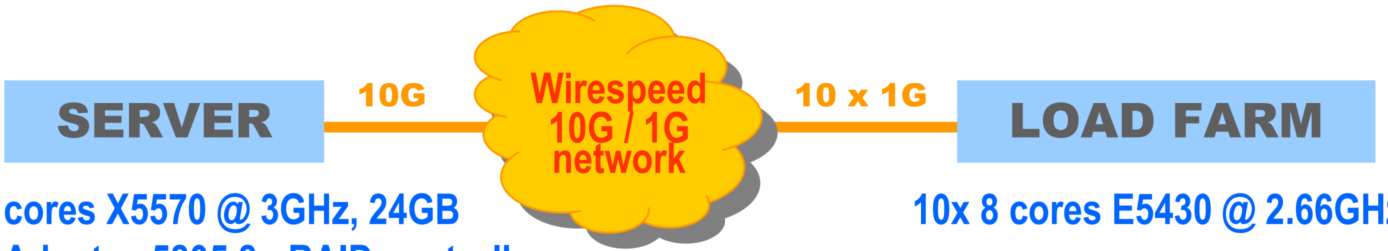
- Comparison is being done on the common hardware base, employing a set of realistic use cases relevant for the HEP community; one of our ancillary goals is thus to enlarge and keep up-to-date the use case library.

Disclaimer

- We are constantly dealing with the “moving target”: data formats and use cases are evolving, hardware base is changing, new versions of storage access and archival software replace the old ones. This implies that results obtained in the storage laboratory are and will always remain a subject to change.

- Whatever we report should hence aways be seen as “work in progress”. We are not trying to provide any final recommendations but are rather sharing with you our findings and are ready to accept any advice and feedback.

Hardware setup 2010 at KIT



8 cores X5570 @ 3GHz, 24GB
3 Adaptec 5805 8p RAID controllers
24 Hitachi drives of 1 TB
1 Intel 82598EB 10G NIC

10x 8 cores E5430 @ 2.66GHz,16GB

This setup reperesents well an elementary fraction of a typical large hardware installation and has basically no bottlenecks:

- **Each of the three Adaptec controllers may deliver 600+ MB/sec (R6)**
- **Ttcp memory-memory network test (1 server – 10 clients) shows full 10G speed**

(In 2009 we were limited by 4x 1G NICs and only one RAID controller)

Details of the test environment (June 2010)

- RHEL 5.4/64bit on all nodes (kernel 2.6.18-164.11.1.lustre / -164.15.1)
- Lustre 1.8.2
- GPFS 3.2.1-17
- OpenAFS/OSD 1.4.11 (trunk 984)
- dCache 1.9.7
- Xrootd 20100315-1007 with default settings
- Hadoop 0.20-1+169.89 from Cloudera

- Use Case 1: CMS “Datascan” standalone job - fw v.3.4.0
  (Giacinto Donvito) – scans, almost serially, through the root data structures

- Use Case 2: ATLAS “Hammercloud” standalone job – fw v.15.6.1
  (Daniel van der Ster) – scans and randomly navigates inside the root data files

How the tests were performed

In all cases with the only exception of Hadoop/serverless, the method was as follows:

- Configure the server and client parts of a solution under test;
- Load the ATLAS and CMS data files into the data area under test;
- Run 20,40,60,80 jobs per 10-node cluster (2,4,6,8 jobs per node); each of the jobs is processing a dedicated non-shared set of event files;
- In each of the measurements start all the jobs simultaneously and then kill them simultaneously, after some predefined period of smooth running;
- Count the total numbers of events processed in each of the runs; These numbers may be compared directly for all solutions under test.
- While the jobs are running, measure also the average incoming MB/sec on each of the 10 Ethernet interfaces of the worker nodes;
- Try to tune each of the solutions under test to get the largest possible numbers of events processed per predefined period;

Hadoop/serverless configuration:

All 10 worker nodes all acted as data providers and data clients. Each of the nodes had 2 disk drives, so in the end we had 20 data drives. As in the case of server we had 18 data drives after R6 formatting, it made sense to compare the Hadoop/serverless test results with those of the server-based configurations.

Tunables

We report here, for reference, some of the relevant settings that were used so far.

Diskware: three stanadlone RAID-6 arrays of 8 spindles, stripe size=1M;
played a lot with disk readaheads, negligible influence on final results

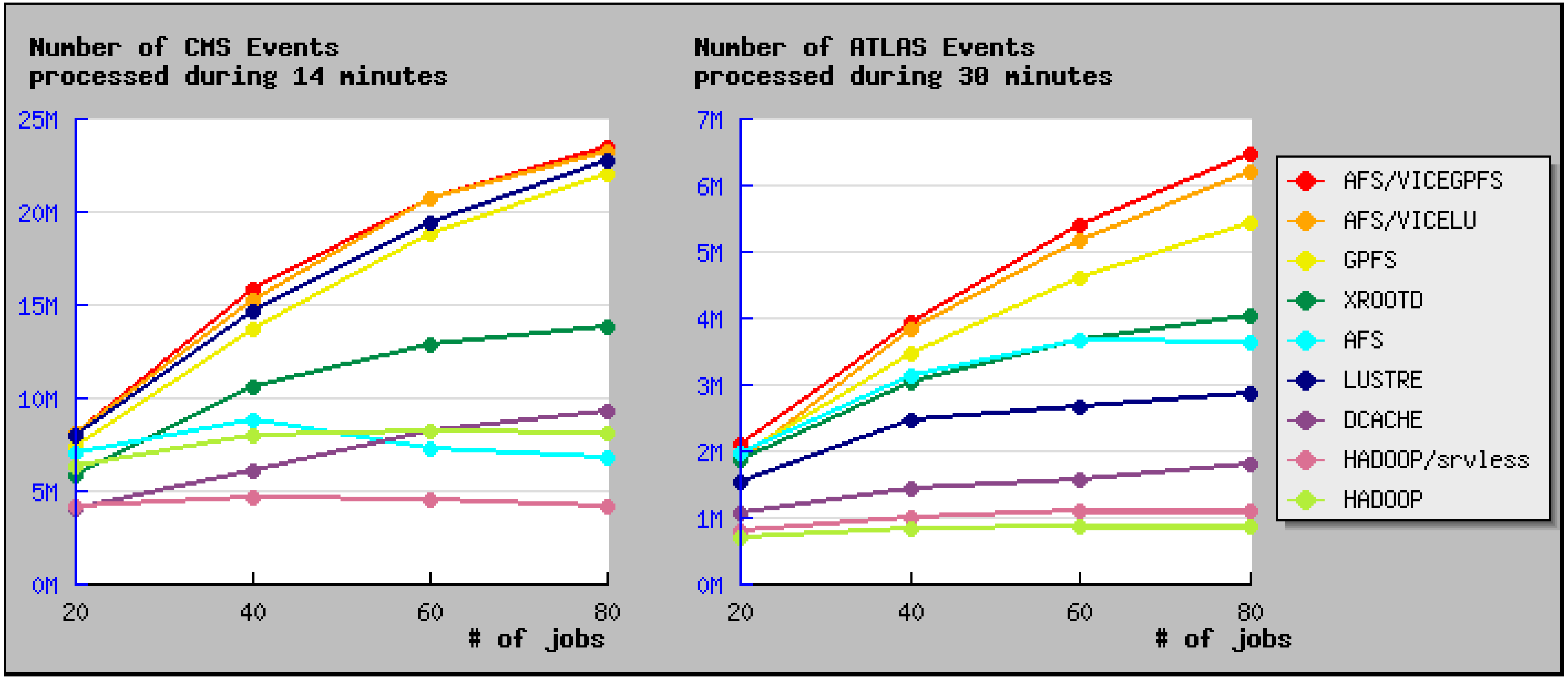
Lustre: No checksumming, No caching on server
Formatted with: "-E stride=256 -E stripe-width=1536"
Data were spread over 3 file systems (1 MGS +3 MDT)
OST threads: "options ost oss_num_threads=512"
Read-aheads on clients: 4MB (CMS), 10MB (ATLAS) later converged on 4MB

GPFS: 3 NSDs, one per RAID-6 array, 3 file systems (one per NSD)
-B 4M –j cluster - maxMBpS 1250 - maxReceiverThreads 128
nsdMaxWorkerThreads 128 - nsdThreadsPerDisk 8 - pagepool 2G

AFS, dCache, Xrootd: 3 XFS partitions (one per RAID array)
Formatted with: "-i size=1024 -n size=16384 -l version=2 -d sw=6,su=1024k"
Mounted with: "logbsize=256k,logbufs=8,swalloc,inode64,noatime"
Afsd options: "memcache, chunksize 22, cache size 500MB" (Vice/Lu, Vice/GPFS)
"memcache, chunksize 22, cache size 4GB" (Native)
Xrootd caching suffix (effective only in ATLAS case):
?cachesz=200000000&readaheadsz=100000000&readaheadstrategy=2&rmpolicy=1&readtrimblksz=65536
dCache library: libdcap++ from Ganga was used

Hadoop fuse 2.7.4-8, rdbuffer=131072, /dev/sdX readaheads of 16M
3 XFS partitions (with server) like in dCache test, or 20 ext4 partitions (serverless)
(*) Unstable under heavy load (write aborts on massive writes, few crashes on reads)

Current results



- Storage Efficiency (events processed / minute) may vary a lot from one solution to another. By simply changing the data archival technology on the same hardware base, as much as a factor of 4-5 in efficiency increase may be obtained
- Some of the solutions look universally good for both (very different) use cases
- Posix file systems in general look more efficient compared with the special solutions. They also require less tuning effort.

More detail (ATLAS test case)

For completeness, we quote here the numbers of events observed, along with the average number of MBs per second entering all the client network interfaces during the test job execution.

	20 threads	40 threads	60 threads	80 threads
Hadoop	329 MB/sec 707290 evs	386 MB/sec 823822 evs	423 MB/sec 871783 evs	437 MB/sec 870047 evs
Hadoop srvless	354 MB/sec 808221 evs	453 MB/sec 1013770 evs	499 MB/sec 1089265 evs	517 MB/sec 1090403 evs
dCache	111 MB/sec 1050184 evs	158 MB/sec 1420947 evs	176 MB/sec 1562001 evs	196 MB/sec 1805839 evs
LUSTRE	113 MB/sec 1543639 evs	188 MB/sec 2464774 evs	201 MB/sec 2682563 evs	225 MB/sec 2850484 evs
AFS native	140 MB/sec 1960132 evs	232 MB/sec 3144659 evs	275 MB/sec 3659608 evs	279 MB/sec 3628869 evs
Xrootd	445 MB/sec 1855726 evs	745 MB/sec 3034830 evs	913 MB/sec 3659365 evs	1035 MB/sec 4024395 evs
GPFS	185 MB/sec 1923523 evs	386 MB/sec 3466926 evs	548 MB/sec 4593836 evs	689 MB/sec 5438793 evs
AFS/VLU	132 MB/sec 1856441 evs	279 MB/sec 3849246 evs	388 MB/sec 5156440 evs	475 MB/sec 6190785 evs
AFS/VGPFS	151 MB/sec 2092590 evs	297 MB/sec 3949517 evs	422 MB/sec 5404200 evs	541 MB/sec 6479655 evs

More detail (CMS test case)

	20 threads	40 threads	60 threads	80 threads
Hadoop srvless	166 MB/sec 4218626 evs	214 MB/sec 4637679 evs	222 MB/sec 4513393 evs	213 MB/sec 4133450 evs
AFS native	192 MB/sec 7015157 evs	279 MB/sec 8758298 evs	277 MB/sec 7243828 evs	277 MB/sec 6784062 evs
Hadoop	225 MB/sec 6222536 evs	313 MB/sec 7030508 evs	364 MB/sec 7232530 evs	391 MB/sec 7240599 evs
dCache	178 MB/sec 4025939 evs	297 MB/sec 6027288 evs	409 MB/sec 8194695 evs	502 MB/sec 9272238 evs
Xrootd	112 MB/sec 5859415 evs	206 MB/sec 10553195 evs	280 MB/sec 12713531 evs	341 MB/sec 13835426 evs
GPFS	203 MB/sec 7397335 evs	388 MB/sec 13677869 evs	557 MB/sec 18756221 evs	711 MB/sec 21969636 evs
LUSTRE	169 MB/sec 7921081 evs	330 MB/sec 14274838 evs	451 MB/sec 19023629 evs	554 MB/sec 22544025 evs
AFS/VILU	213 MB/sec 8149266 evs	414 MB/sec 15789180 evs	580 MB/sec 20549956 evs	710 MB/sec 22670236 evs
AFS/VGPFS	214 MB/sec 8044773 evs	411 MB/sec 15851773 evs	604 MB/sec 20729597 evs	740 MB/sec 23510218 evs

Observations - GPFS

- This time we were able to obtain excellent GPFS results, much better than those that were seen before. Most probably, this improvement may be explained by the elimination of the network bottleneck that we had in our previous setup (we stepped to 1000 MB/sec from 450 MB/sec). As well, we are now running a more recent version of GPFS software which is known to be more performing.

- GPFS is hence looking quite attractive. IBM had recently changed its licensing policies, and the product became more affordable. As of the next quarter, they promise to propose the even more convenient site licenses.

- GPFS technology allows for smooth addition and removal of storage devices which makes it much more manageable in comparison with Lustre. Its principal drawback today is the lack of the fragmented file system layout. Striping may not be switched off, thus a loss of just one NSD may result in a visible data outage across the file system.

Observations – AFS/Vicep-Lustre

- Somehow an amalgam of AFS and Lustre transport presented itself as one of the most efficient solutions for the two extreme cases (CMS use case with its modest random I/O component vs ATLAS/OldFormat with high random I/O).

- Running AFS with a speed of Lustre is especially attractive because of the value added features of AFS. It provides the fine-grained security level, and adds the possibility to add/remove Lustre OSTs without interrupting the file system activity. It is available at no cost; even if it is true that Lustre management on a large scale may require more human resources, this hybrid solution is definitively deserving more attention.

- NB: The AFS/VILU tests were run as superuser. Some small overhead may be necessary to support the non-privileged user access.

Observations – remote Root protocol

- The CMS and ATLAS frameworks under test were assembled using the production version of Root of 2009 (5.22.00d).
- Both CMS and ATLAS frameworks must be sensitive to caching policies at the client level, however only that of ATLAS was reacting visibly to the Root caching parameters passed via the file name suffix. In particular, we were able to increase 4+ fold the efficiency for ATLAS/Xrootd using these parameters as was suggested by Fabrizio Furano.
- For instance, this is an example of how ATLAS/Xrootd framework behaved in vanilla variant, and after feeding in the client caching instructions:

```
+---------+--------------------------------------------------+
|Vanilla  |     985 MB/sec   1132 MB/sec   1153 MB/sec  1156 MB/sec  |
|         |     808374 evs    913080 evs    910937 evs   895540 evs  |
+---------+--------------------------------------------------+
|Best     |     445 MB/sec    745 MB/sec    913 MB/sec  1035 MB/sec  |
|Caching  |    1855726 evs   3034830 evs   3659365 evs  4024395 evs  |
+---------+--------------------------------------------------+
```

- BTW, we have also tried the “root door” of dCache (the suffix trick here was of no help, it even led to crashes of some of the threads..):

```
+---------+--------------------------------------------------+
|Root door|     664 MB/sec    949 MB/sec   1108 MB/sec  1069 MB/sec  |
|(Vanilla)|     557061 evs    764518 evs    837213 evs   824075 evs  |
+---------+--------------------------------------------------+
```

July - September 2010

- In July two more use cases from FNAL were being configured so no new tests took place. In August the DESY team was further tuning dCache.

- Next, we used the first 10 days of September to upgrade the laboratory to the new level of the OS and file system software. The new use cases are due to be ready at the beginning of October, so in September we have an option to take a closer look at AFS. We are especially interested in trying the latest 1.5.xx release (xx==77 at the moment).

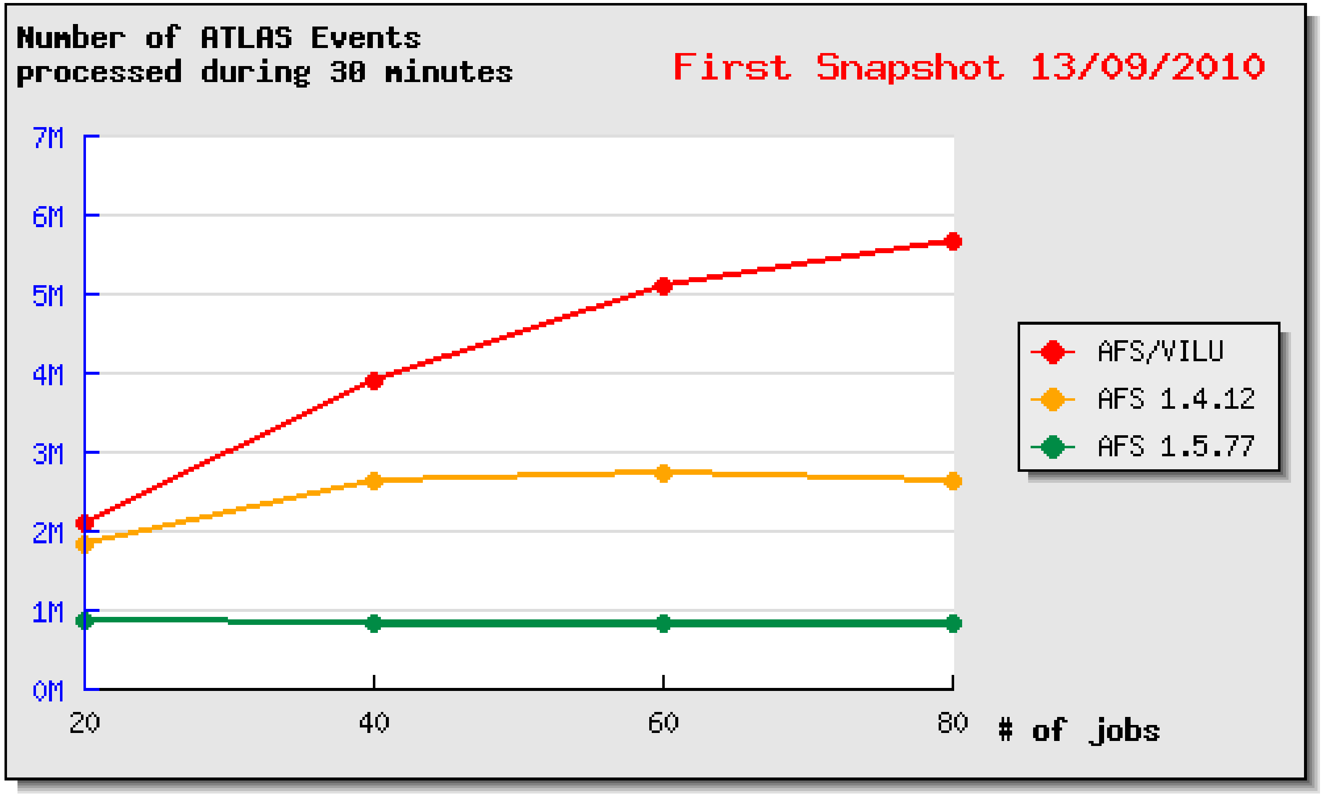
- The machines were all upgraded to RHEL55/2.6.18-194.11.3.el5 kernel series. AFS/OSD was moved to trunk 1037 and Lustre was updated to 1.8.4. The RAID arrays were rebuilt as well, so we ended up with quite a renewed setup.

OpenAFS 1.5.77 – first exposure

On advice of Hartmut Reuter, we prepared the first 1.5.77 test in the following way:

- **AFS_NRXPACKETS was set up to 4096 in afs.h**
- **Configure was run with these switches :**
  --enable-largefile-fileserver --enable-fast-restart --disable-optimize-kernel --disable-optimize-lwp --disable-optimize
- **Some relevant network related kernel parameters on the clients were set as follows:**
  net.core.rmem_max=1048575
  net.core.wmem_max=1048575
  net.core.rmem_default=1048575
  net.core.wmem_default=1048575
  net.core.optmem_max=1048575
  net.core.netdev_max_backlog=10000

1.5.77 ☹ vs 1.4.12 vs VICEP/Lustre



- NB: This is our first ever try of 1.5.77, it definitively yet has to be understood and tuned
- We certainly are interested to look into the 1.5.xx series, especially since 1.6 release seems to be imminent.

Immediate plans

- The group is planning to run the lab tests at KIT in September and October, and then to present its next progress report at Cornell in the beginning of November.
- The test program includes migration to the updated ATLAS (15.6.6) and CMS (3.7) use cases and hopefully inclusion of a new use case from ALICE and/or LHCb. We shall also be adding NFS 4.1 to the list of storage solutions under test. Storage software and the OS will be upgraded to the latest available levels.
- Finally, our plans include a study of the aggregate performance degradation due to rebuild in progress and evaluation of some new disk hardware.

Discussion