Performance and scalability comparison of disk IO protocols for LHC

G. Donvito
INFN-BARI

Outlook

- **Tests set-up:**
  - XRootD
  - dCache
  - CASTOR (already presented by Luca Dell'Agnello)
  - GPFS (already presented by Luca Dell'Agnello)
- **Client software:**
  - Description
  - Optimization
- **Test results** (CASTOR, dCache, GPFS, XRootD)
- **Conclusions**

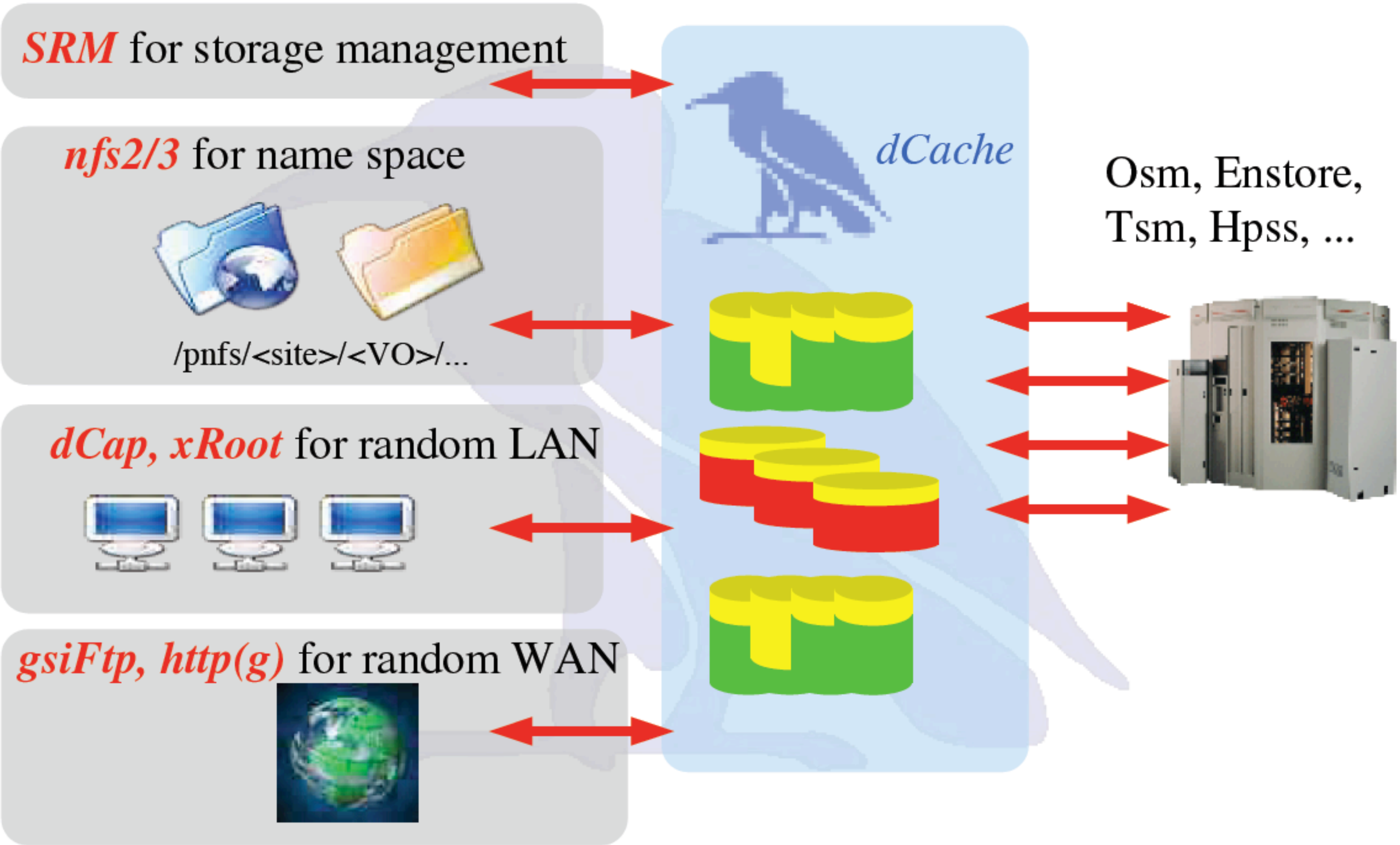
dCache overview

- It is developed in a large collaboration between Desy and FNAL (plus some other minor contributions)
- GOALS:
  - To make a distributed storage system to gain high performance and high-availability
  - To provide an abstraction of whole disk space under a unique NFS like file-system (just for metadata operations)
  - To possibly add the support for its own MSS system
    - They are needed only 2 or 3 scripts (put/get/remove)
- File access:
  - provides local and remote access (posix like) with many protocols (dcap, ftp, xrootd) both with and without authentication (gsi or kerberos)
- Access management: access priority and load balancing obtained trough the use of different queue
- Allows multiple copy of files spread over different pools to improve performance and HA
  - pool-2-pool automatic (or manually) transfers
- Allows dynamic "match-making" between pools
  - According to the parameters chosen by the administrator (they can be based on disk space, load, network, type of access etc.)

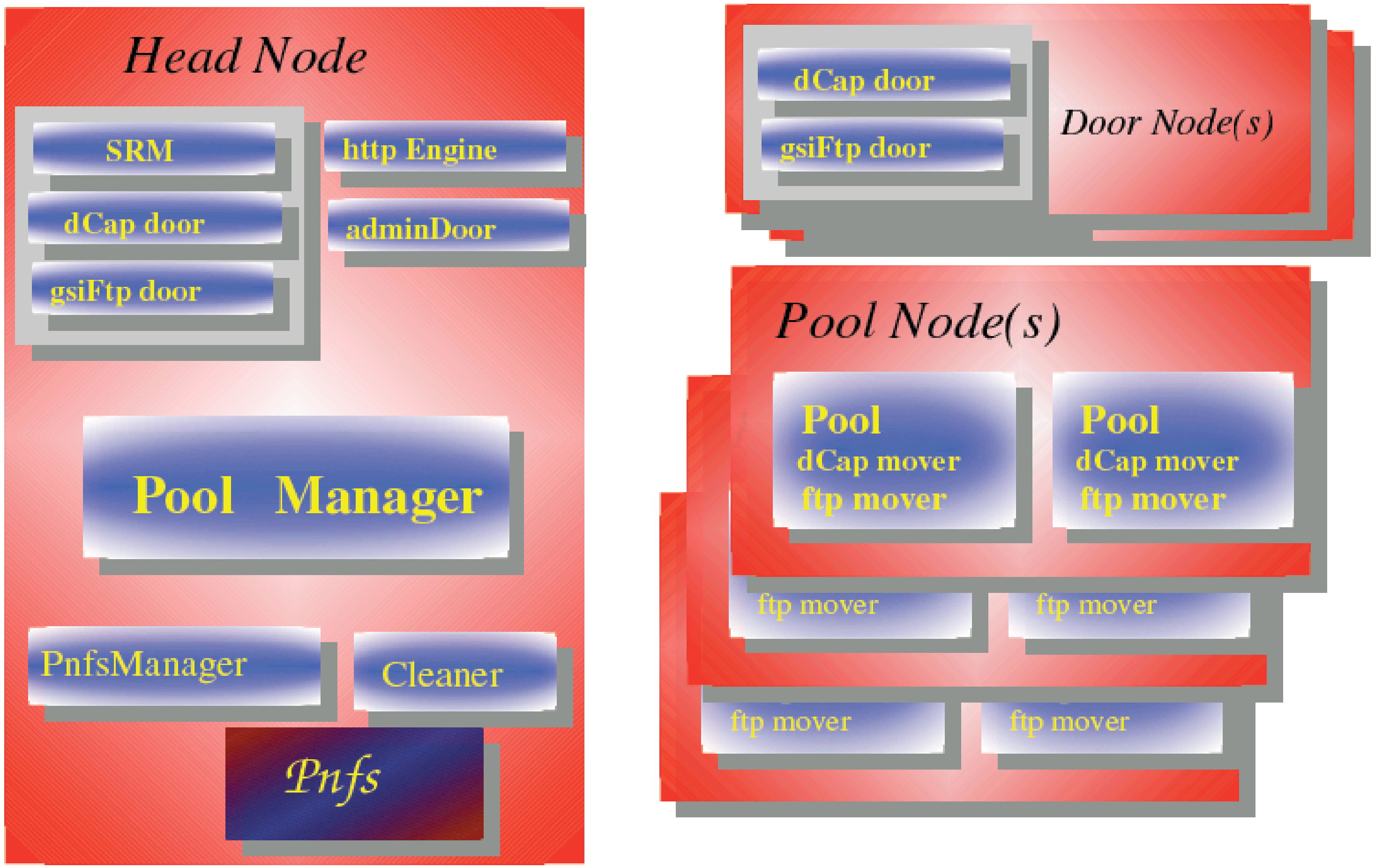
dCache overview (2)

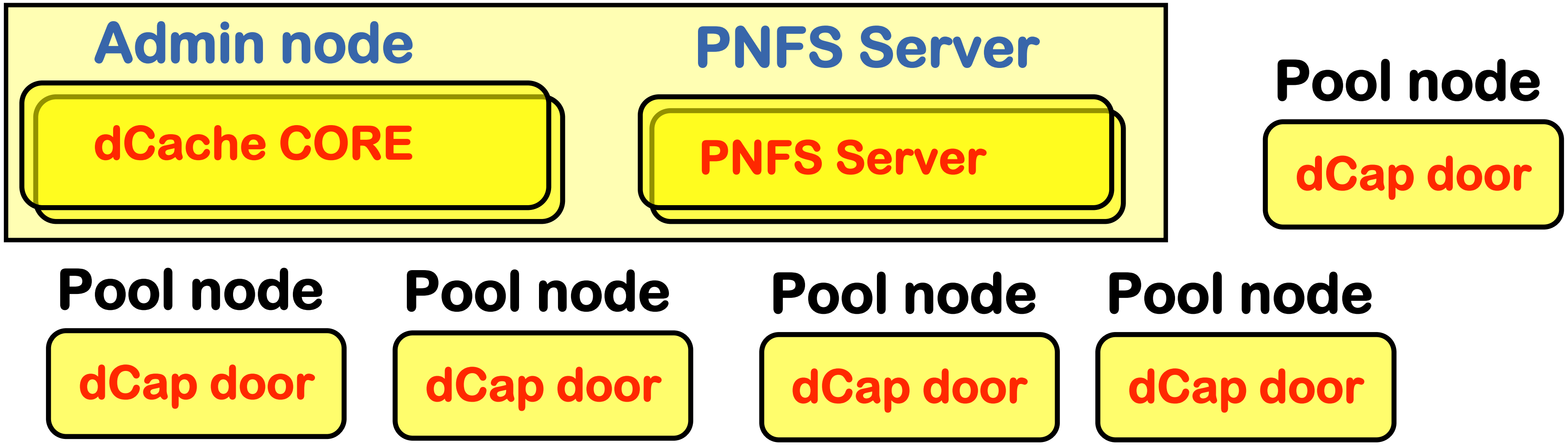
- It is possible to move all the files in a pool to put it in a "scheduled downtime"
  - Or just to choose which file you want to move and where.
- Also the "central services" can be split on different nodes to improve the scalability
- Pool management:
  - gives the possibility to create groups of pools named "storage class" (read, write, cache, or per VO and user bases or use bases)
  - Can be useful for quota management
- Web monitoring, statistical module (also with rate-plot)
- JAVA GUI for administration
- Both SRM v1 and v2 (in pre-production) is available
- Accounting system flat-files or DB based (not user friendly but there are many information) and space used per VO
- It is possible to use WN (or other "not reliable" space) disks to improve performance for local access

dCache overview (3)



dCache overview (4)



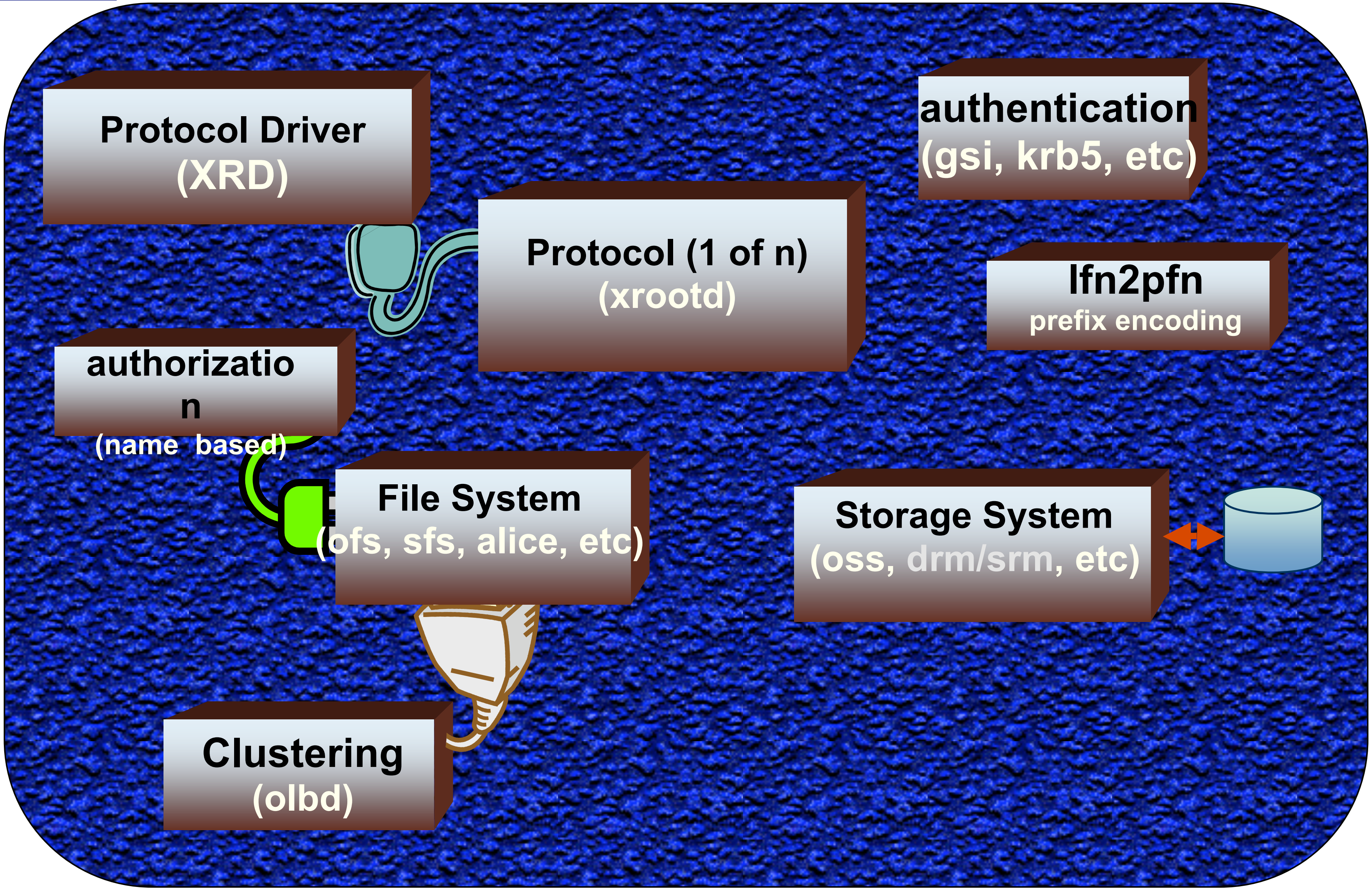


- The admin door is split in two machine: PNFS(Database) and dCache Admin-dcap door
    - The PNFS-server is on flat-files database (the lack in performance is not important since the number of files is small)
- On each machine we have a pool running with 2-3 partitions used
- Small tuning done in order to fit with the large requirements:
    - Number of allocable slots
    - Time-out in opening file

XRootD Overview

- Developed in a collaboration between SLAC, INFN, CERN, BNL and many other contributors
- Purpose:
  - to construct high performance data access systems by means of P2P-like clustering
  - develop a synergy between high performance, low latency servers, virtually unlimited clustering capabilities (up to 262K server nodes)
  - to build systems able to seamlessly ignore a server's failure, even through WANs
  - no central points of failure, no bottlenecks (e.g. file catalogs) except for the single disks performance
  - no 3rd party SW needed, no messy dependencies
  - to do it privileging simplicity, i.e. low admin cost

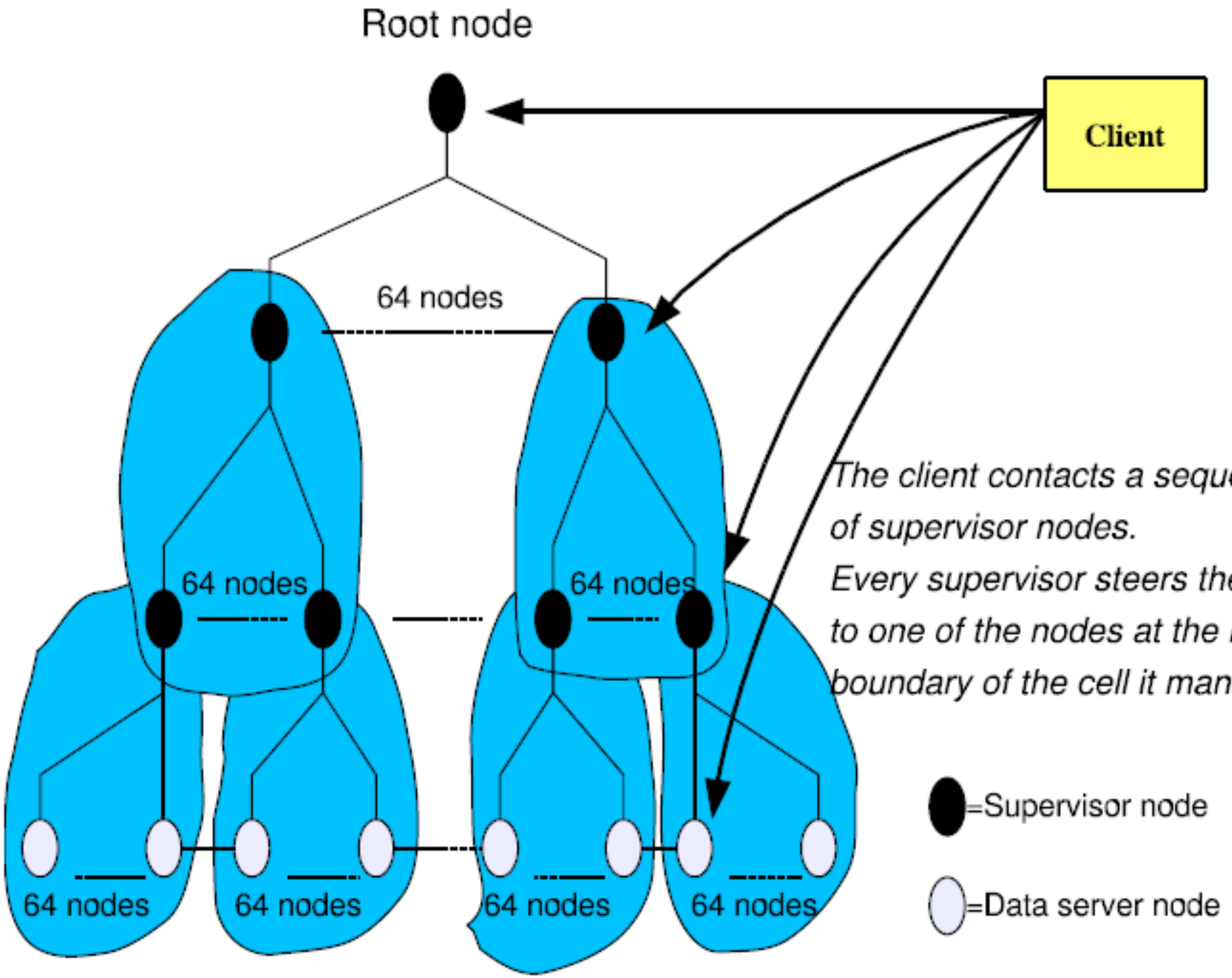
XRootD Overview (2)



XRootD Overview (3)

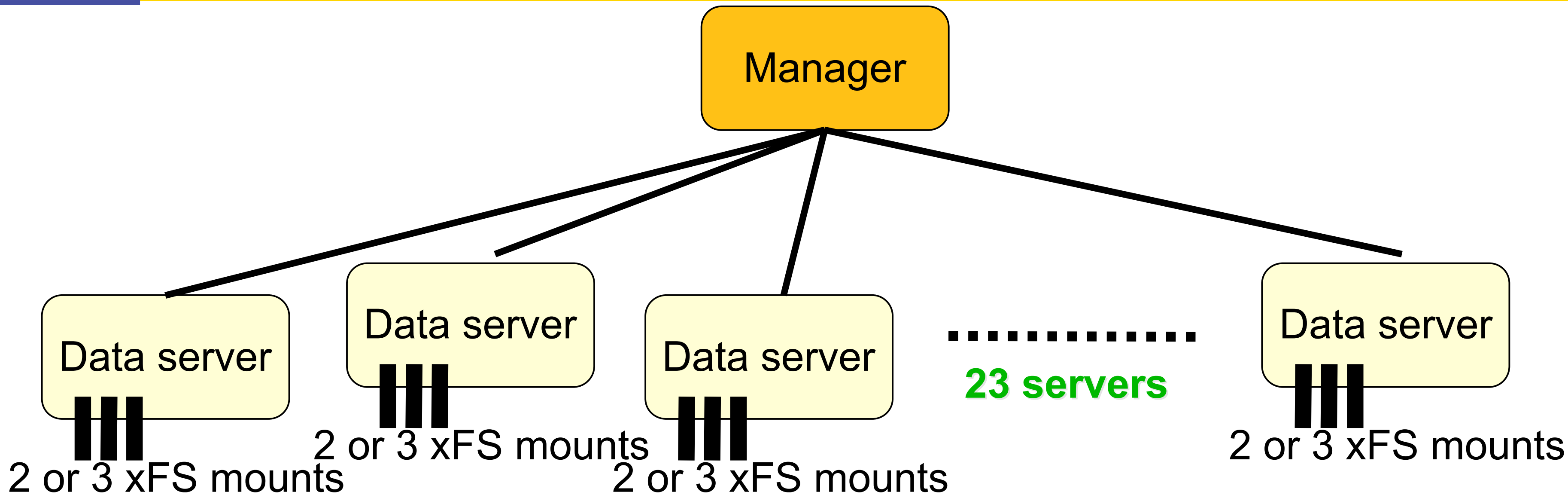
- File access: the storage granularity is at file level
- Plugin-based architecture, entirely POSIX and C++ based, many platforms available, very low dependencies on specific kernels, versions
- SRM compliance through external sw integrations (e.g. Castor), native SRM integration is on the way (STORM)
- Can aggregate different local namespaces into a global unique one
- Load balancing, resource allocation, access and fault tolerance achieved through
  - P2P-like mechanisms at the server side
  - An intelligent fault-detecting client which crawls the server clusters
  - In principle the app does not notice server failures
- Supports any number of file replicas to higher data availability and read performance
- Many interfaces (native, POSIX, ...) available through different sw layers
- Various MSS integrations (HPSS, CASTOR, ...)
- WAN-friendly, not limited to file copying
  - multiple clusters can cooperate through WAN
  - the client can be used to exploit high bandwidth WANs from the applications by hiding the data access latency

XRootD Overview (4)



The client contacts a sequence of supervisor nodes. Every supervisor steers the client to one of the nodes at the lower boundary of the cell it manages.

XRootD: Test Installation Schema



- A total of 24 servers
- 1 Manager + 23 data servers
- No data redundancy, No server redundancy
- Plain default configuration
- 2 or 3 mounts per data server, everything mapped to /store namespace
- Files have been distributed in a round-robin fashion by just writing them to /store
- No server local storage
- Many external mountpoints, pointing to a few disk controllers in the SAN

Client software

- The goal is to simulate a typical analysis job
- The schema for all the client is always the same:
  - There were 4 different implementation: CASTOR, dCache, GPFS, XRootD
- The operation executed by the client is logically simple:
  - It reads the required number of bytes at a given offset for each specified file
  - The list of "read operations" (offset and number of bytes) is given with an input file (named "tracefiles")
  - The list of files to be read is given with an input file:
    - 5 different files are opened by each job
  - The "tracefile" is chosen random between 10 different files:
    - 5 are taken from real BaBar jobs
    - 5 are composed by a random list of offset and number of bytes in order to simulate the worst possible case
    - Each "tracefile" contains 5000 "read operation"
  - It is enough a single read-failure makes jobs fail

Client software (2)

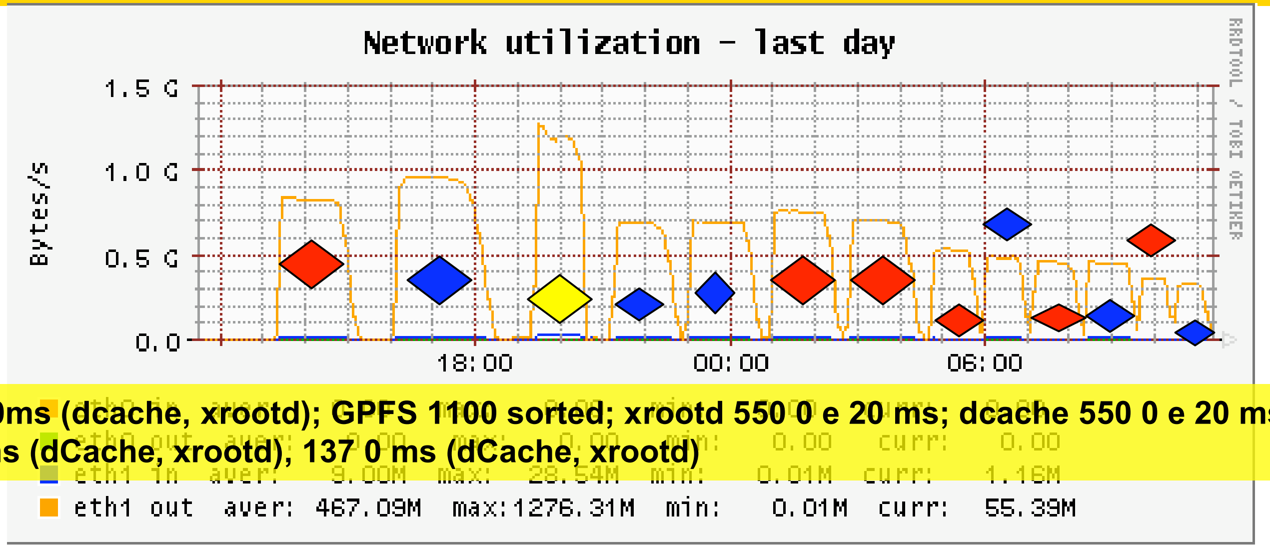
- Highlight on the tests:
  - All the jobs are synchronized in order to maximize the impact on the Storage Manager
    - This is a limit situation in order to simulate a higher number of concurrent jobs
  - All the files (5 file for each jobs) are opened before starting reading
    - Also 5 files per job are used in order to simulate a higher number of concurrent jobs
  - The access pattern is random in order to increase the stress on the storage system (both hardware and software)
  - It is possible to set a “think-time” in order to simulate the CPU time of a typical analysis-job (the CPU is really loaded)
  - Each file is red from several WN
  - Sorted “tracefiles” are used to reduce the load on the disk sub-system
  - A sequential access pattern is used in order to measure pick rate in case of concurrent file access
  - The client reports a lot of information useful for statistics:
    - open_elapsed, data_xfer_elapsed, close_elapsed, total_elapsed, totalbytesreadperfile, maxbytesreadpersecperfile, effbytesreadpersecperfile, throughputperfile, readscountperfile, openedokfilescount

Client side optimization

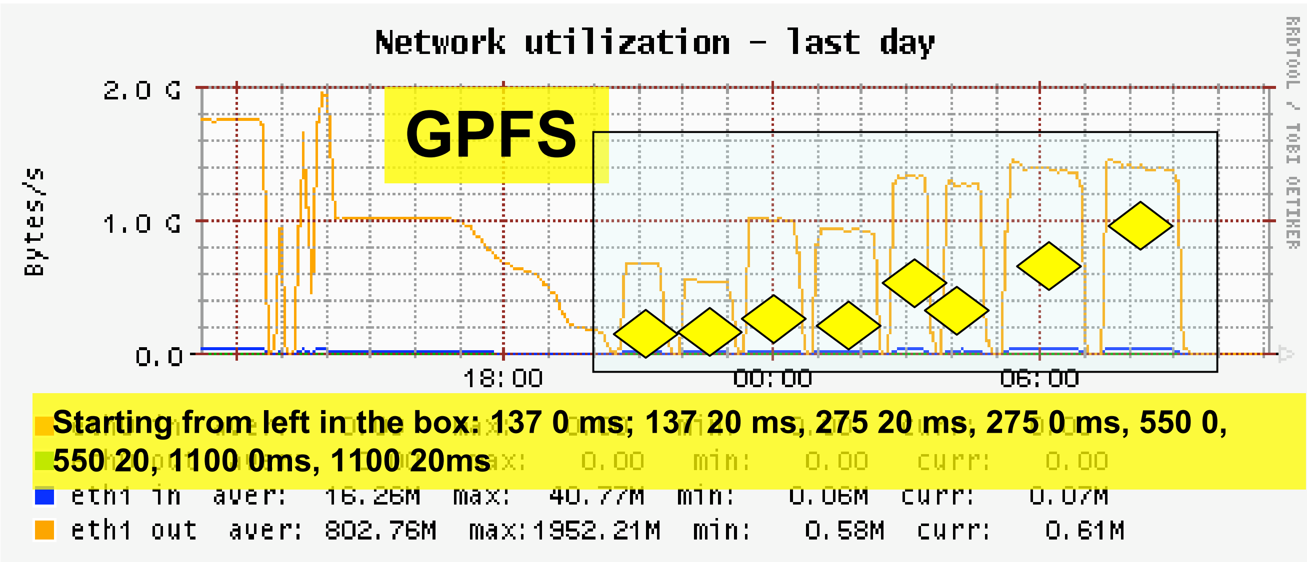
- dCache optimization:
  - Using ENV-Variables:
    - Reducing READHAED (export DCACHE_RA_BUFFER=1000)
      - Avoid overloading the system with data not used by the application
- GPFS optimization:
  - Using simple C “read” function
    - Avoid overloading the system with read action not needed (triggered automatically from C++ library)
- XRootD optimization:
  - Readahead switched OFF, using vectored asynchronous reads of 512 subchunks.
  - Set very high data xfer timeout (1200 secs) to efficiently deal with overloaded disk systems.

Results: Network views

- dCache
- Xrootd
- GPFS



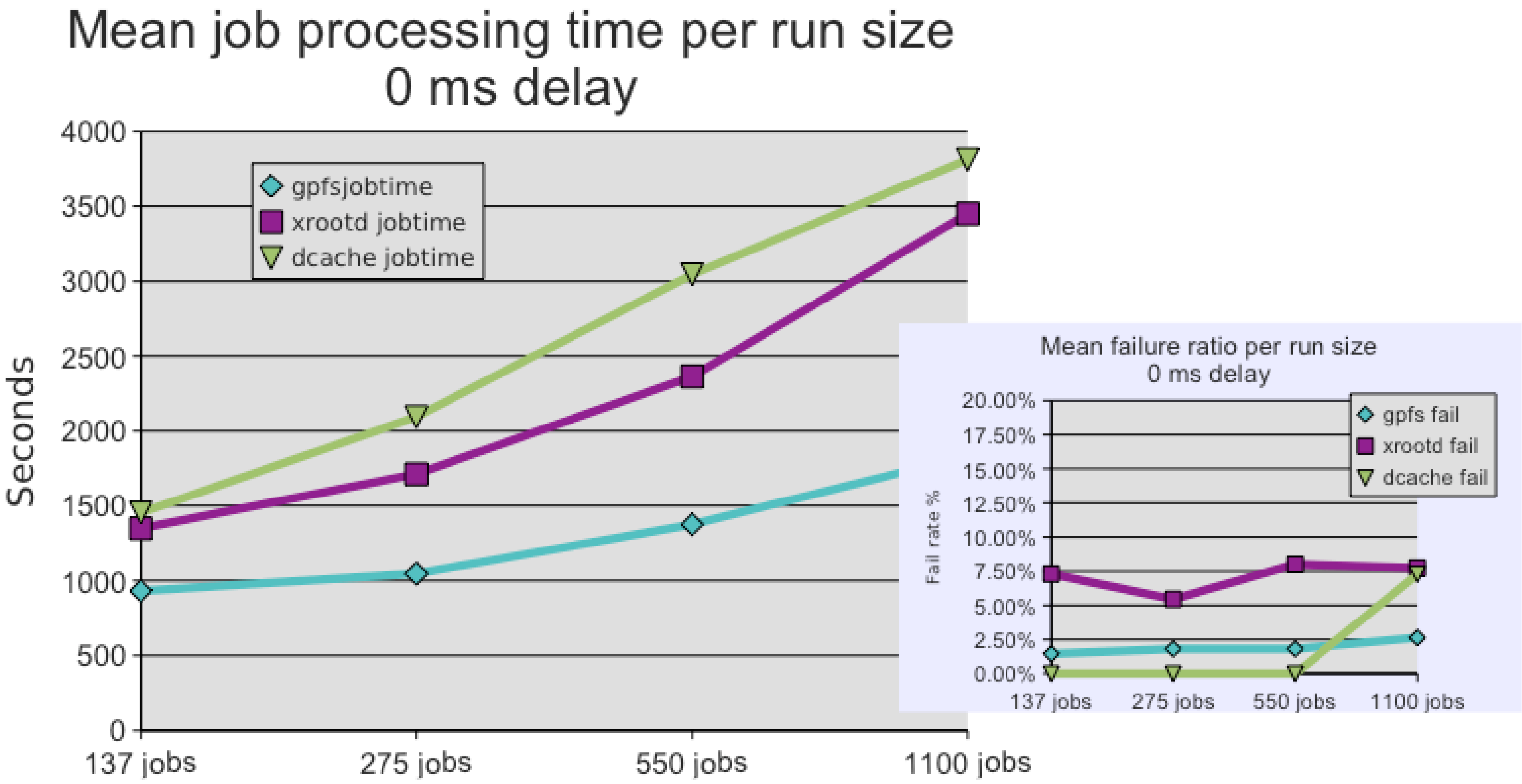
Starting from left: 1100 think-time 20ms (dcache, xrootd); GPFS 1100 sorted; xrootd 550 0 e 20 ms; dcache 550 0 e 20 ms; 275 0 ms (dCache, xrootd), 275 20 ms (dCache, xrootd), 137 0 ms (dCache, xrootd)



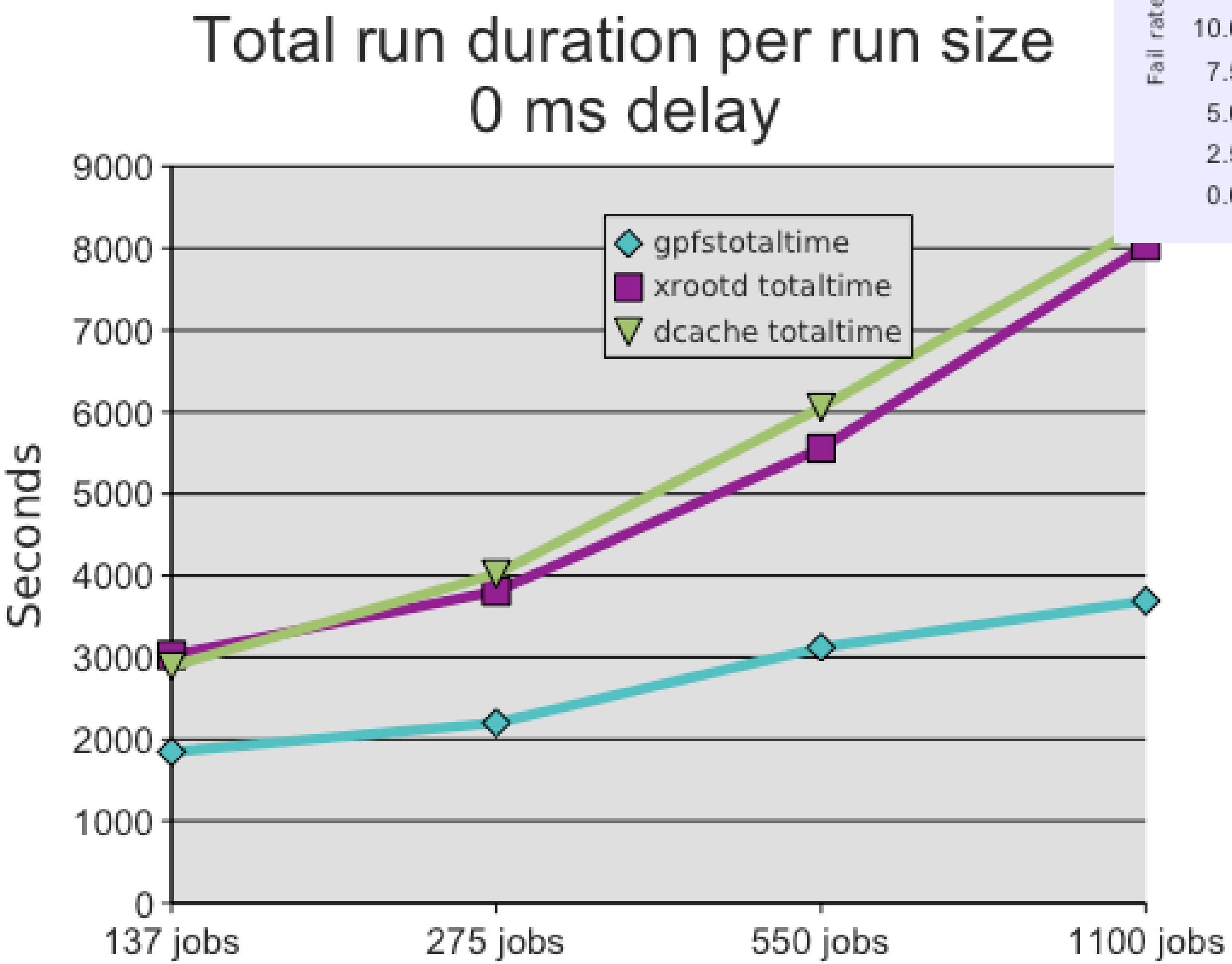
Starting from left in the box: 137 0 ms; 137 20 ms, 275 20 ms, 275 0 ms, 550 0, 550 20, 1100 0ms, 1100 20ms

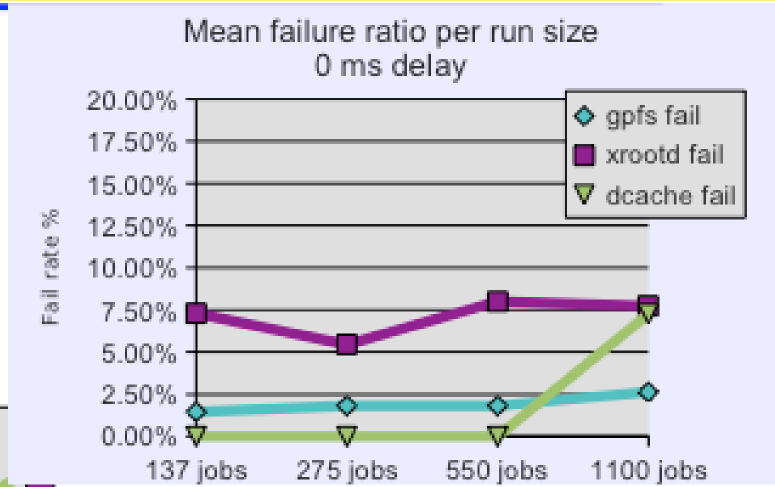
“APPLICATION” VIEWS

Results: JOBtime vs size vs failures - 0ms

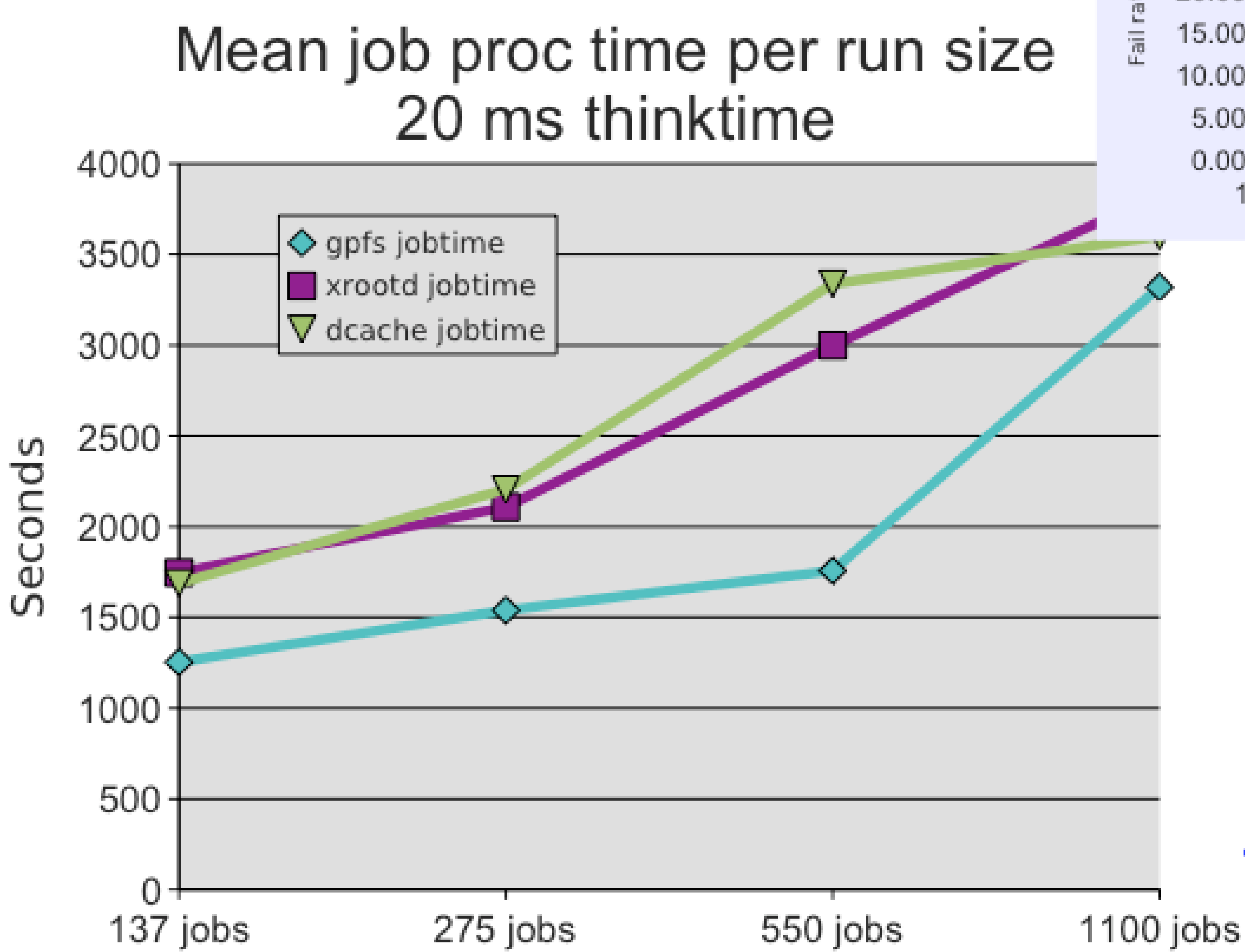


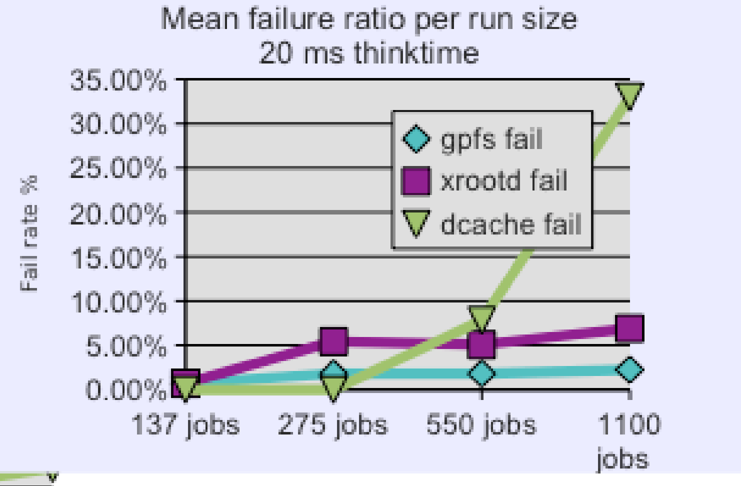
Results: RUNtime vs size vs failures - 0ms



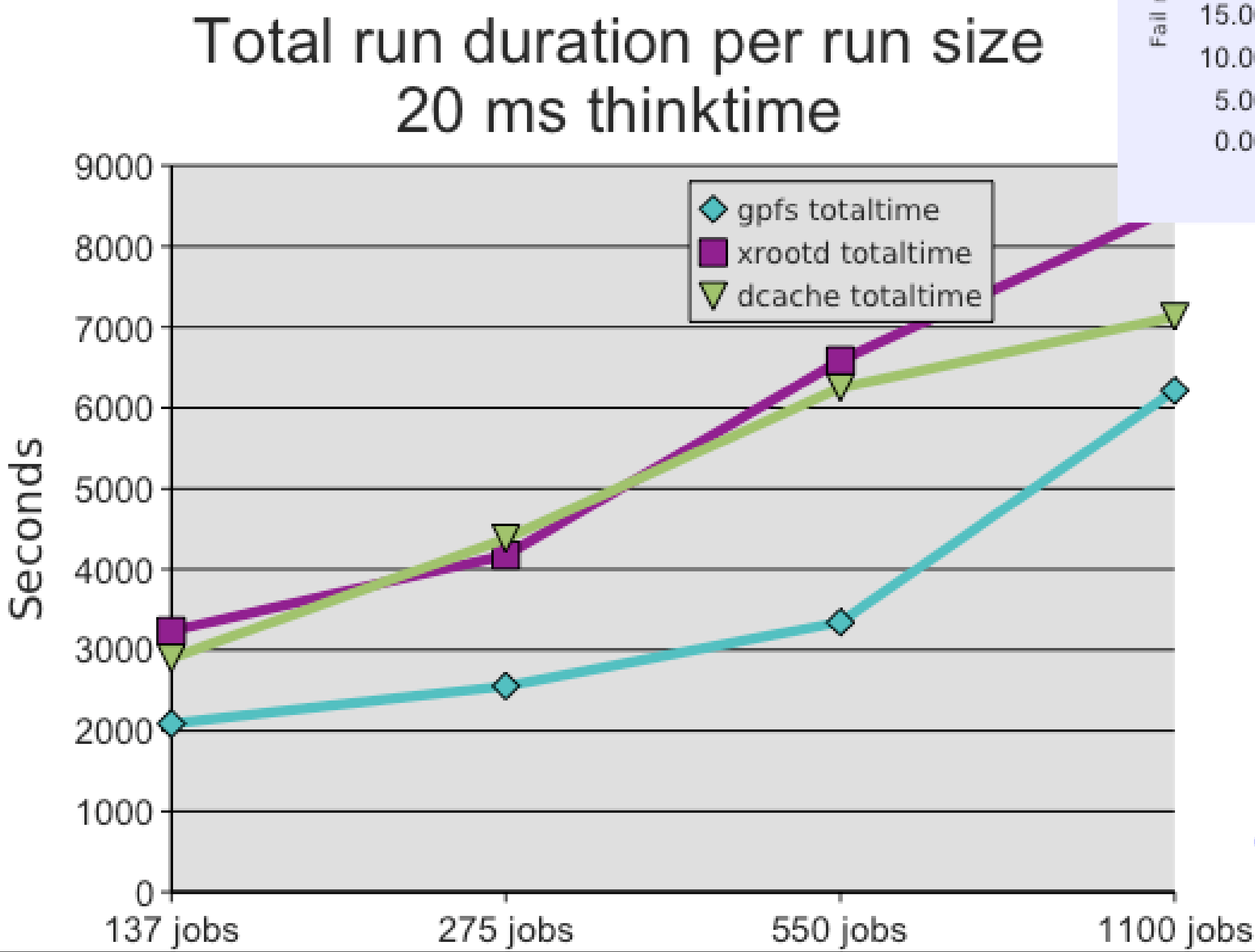


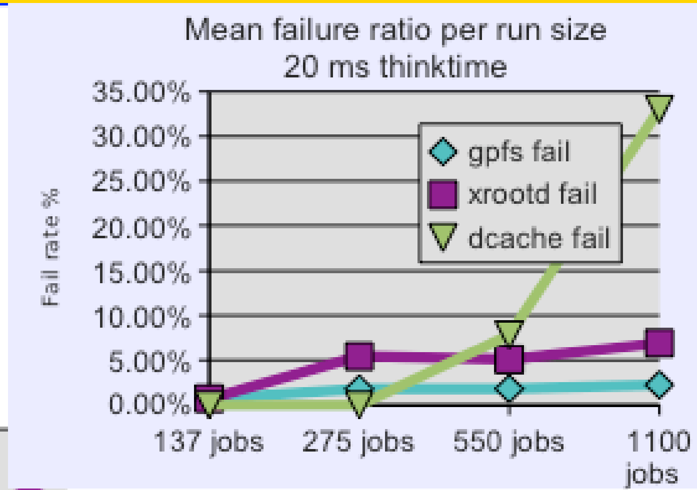
Results: JOBtime vs size vs failures - 20ms



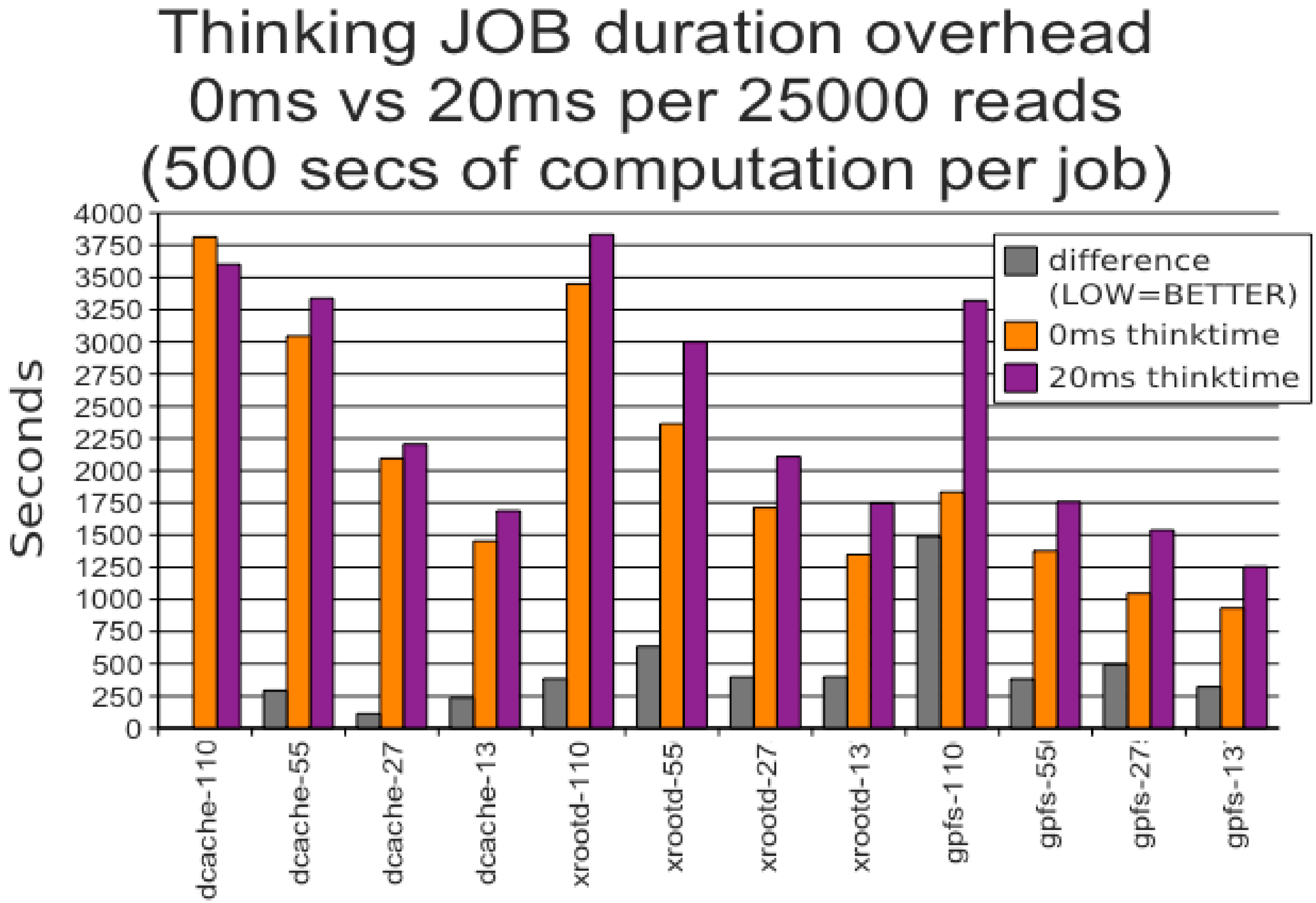


Results: RUNtime vs size vs failures - 20ms

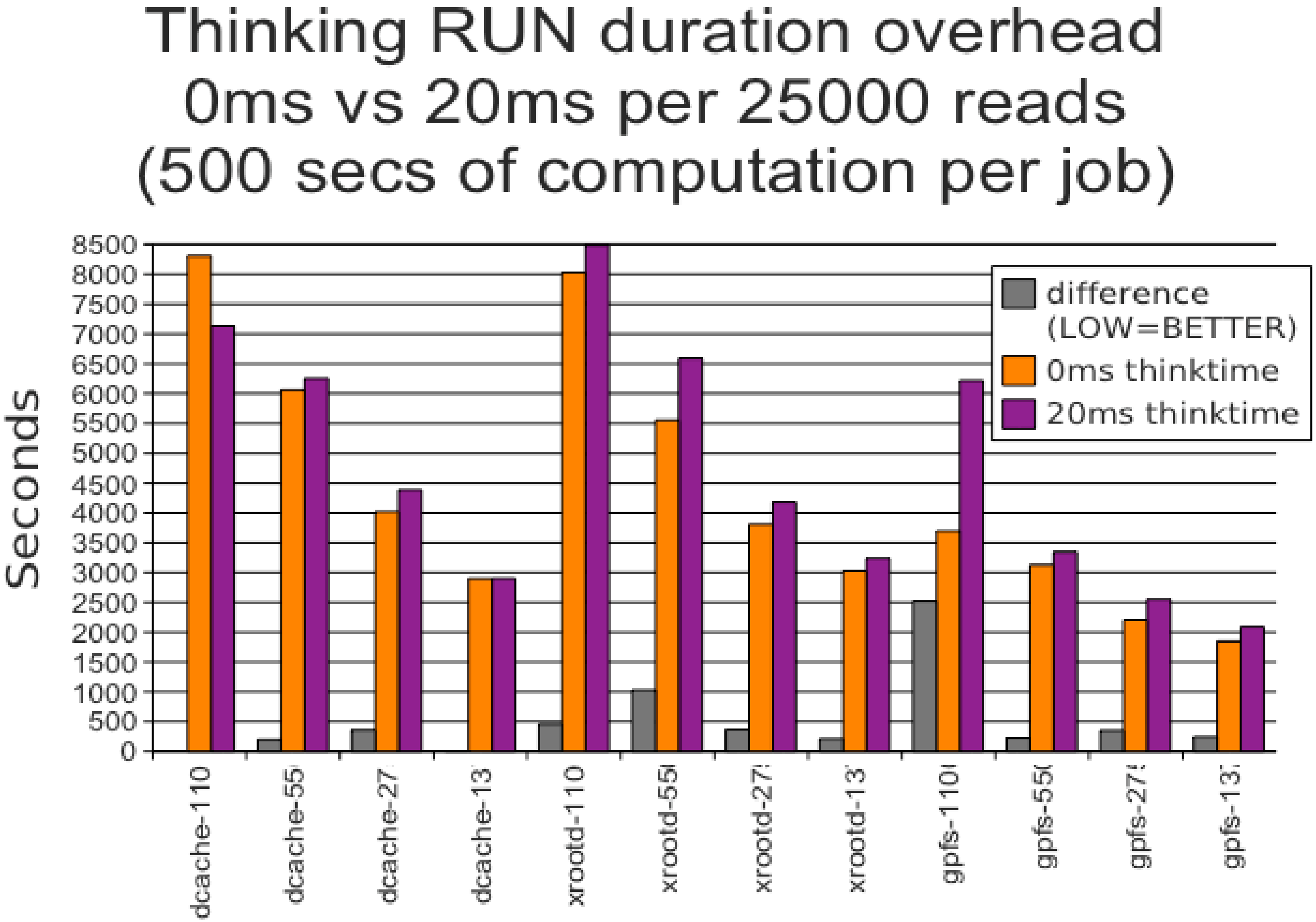




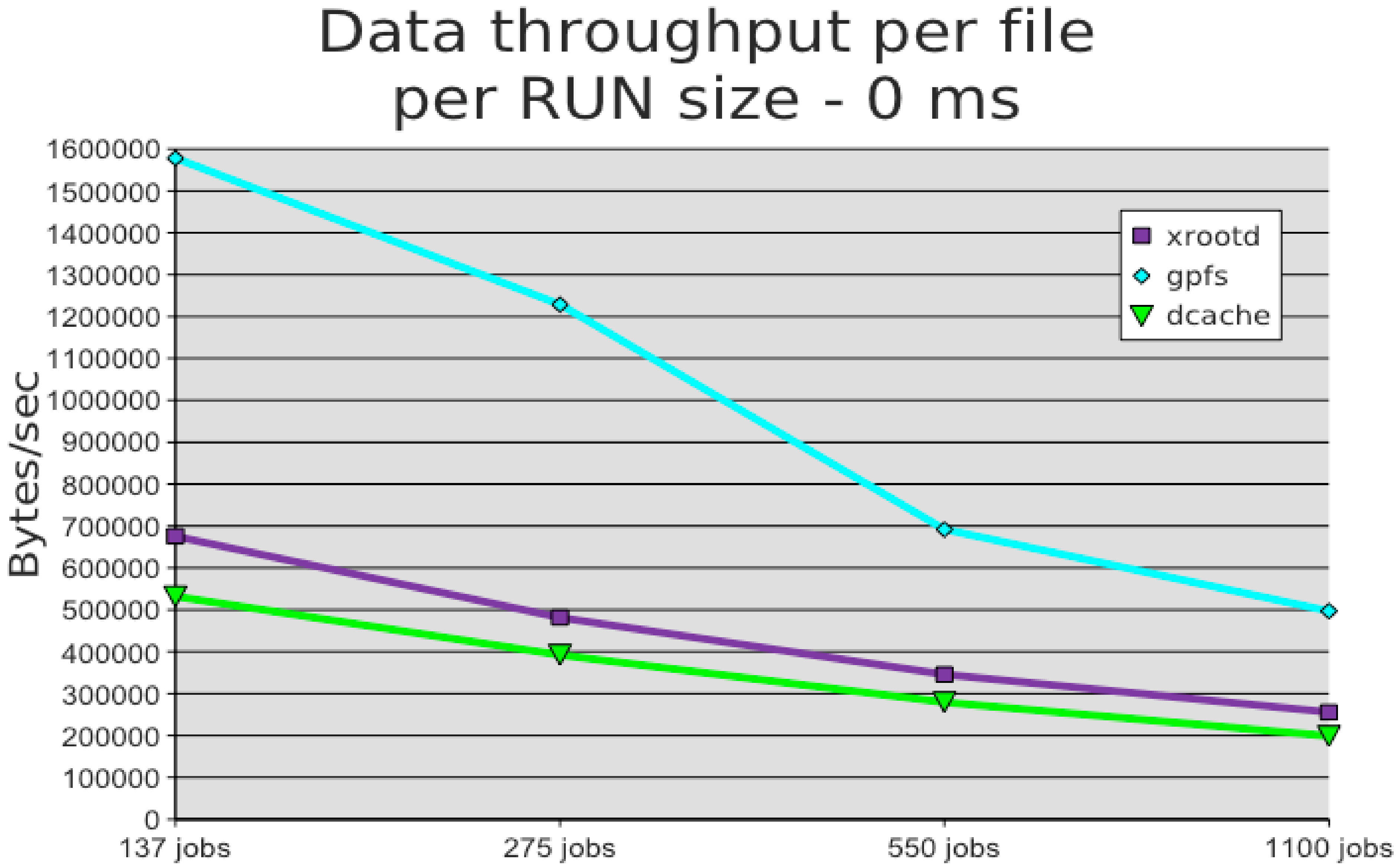
Computation impact on JOBtime



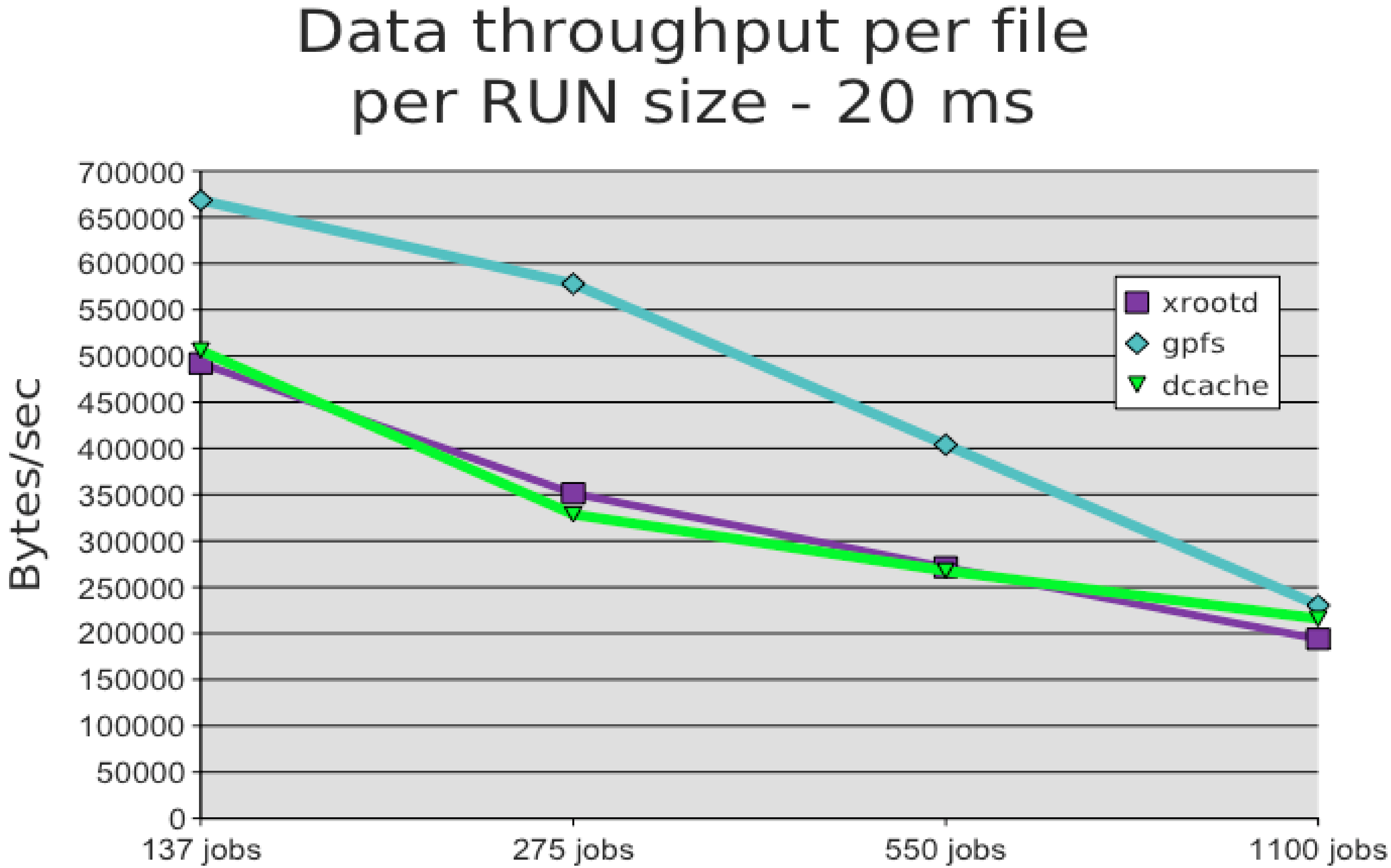
Computation impact on RUNtime



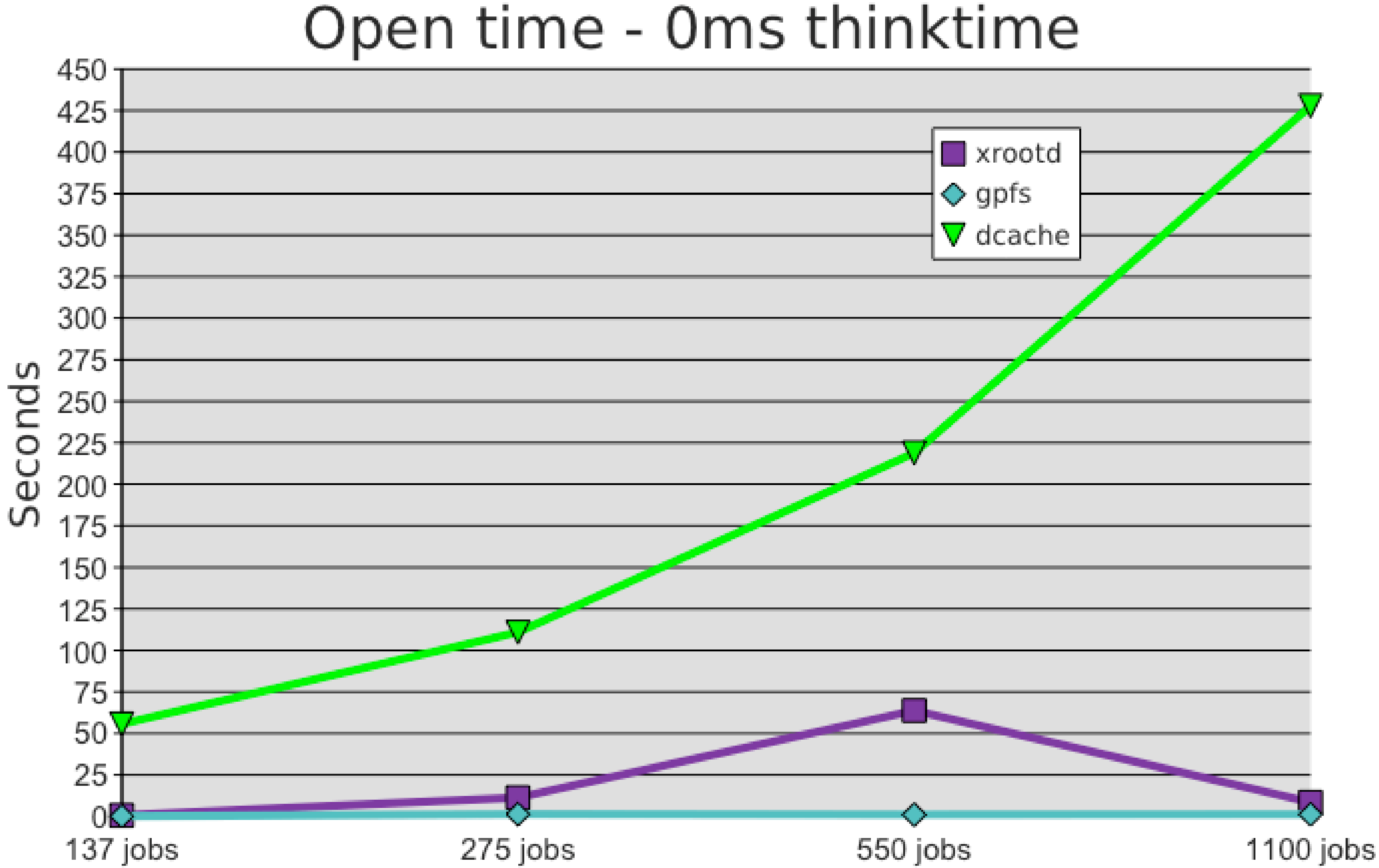
Performance degradation?



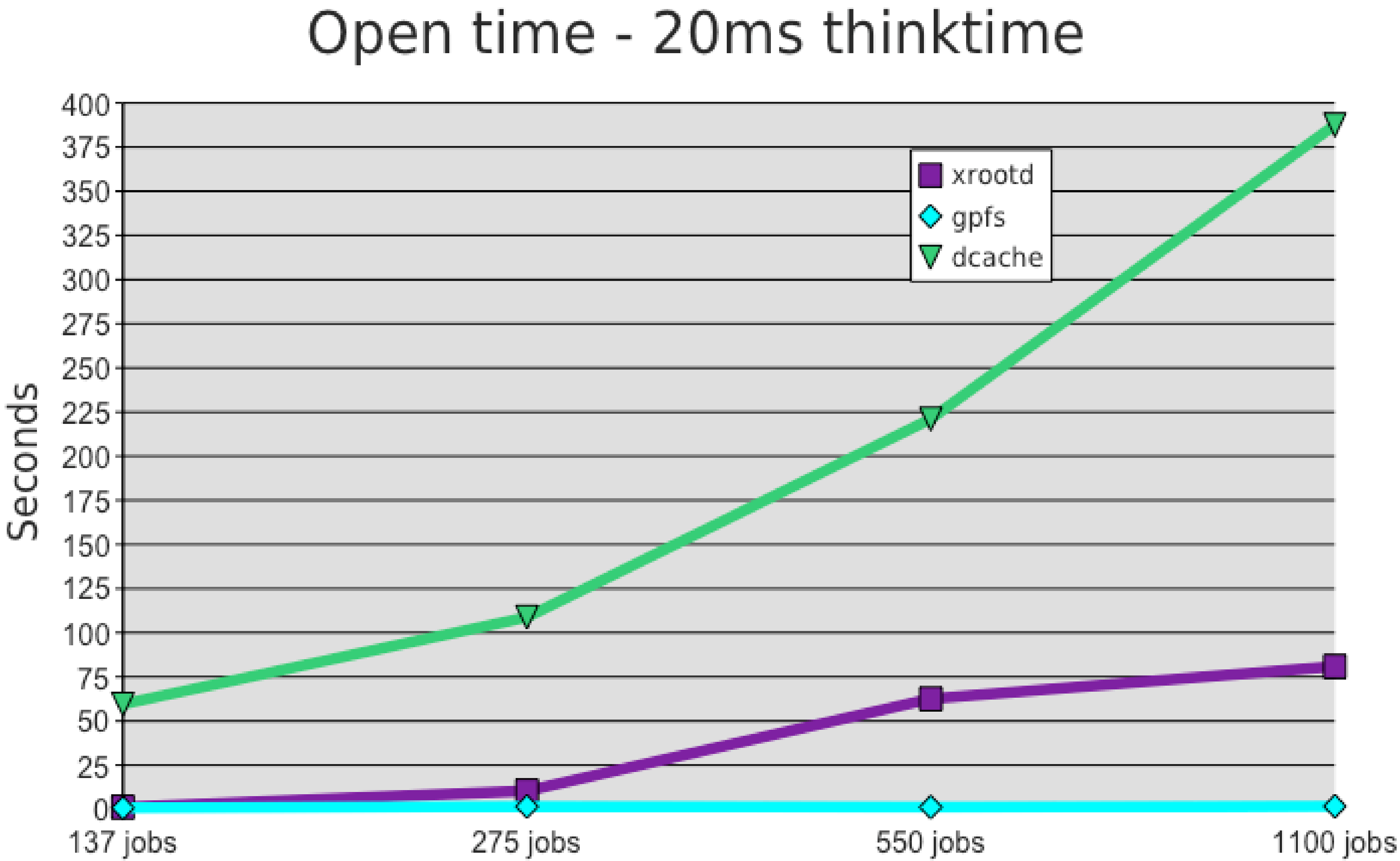
Performance degradation?



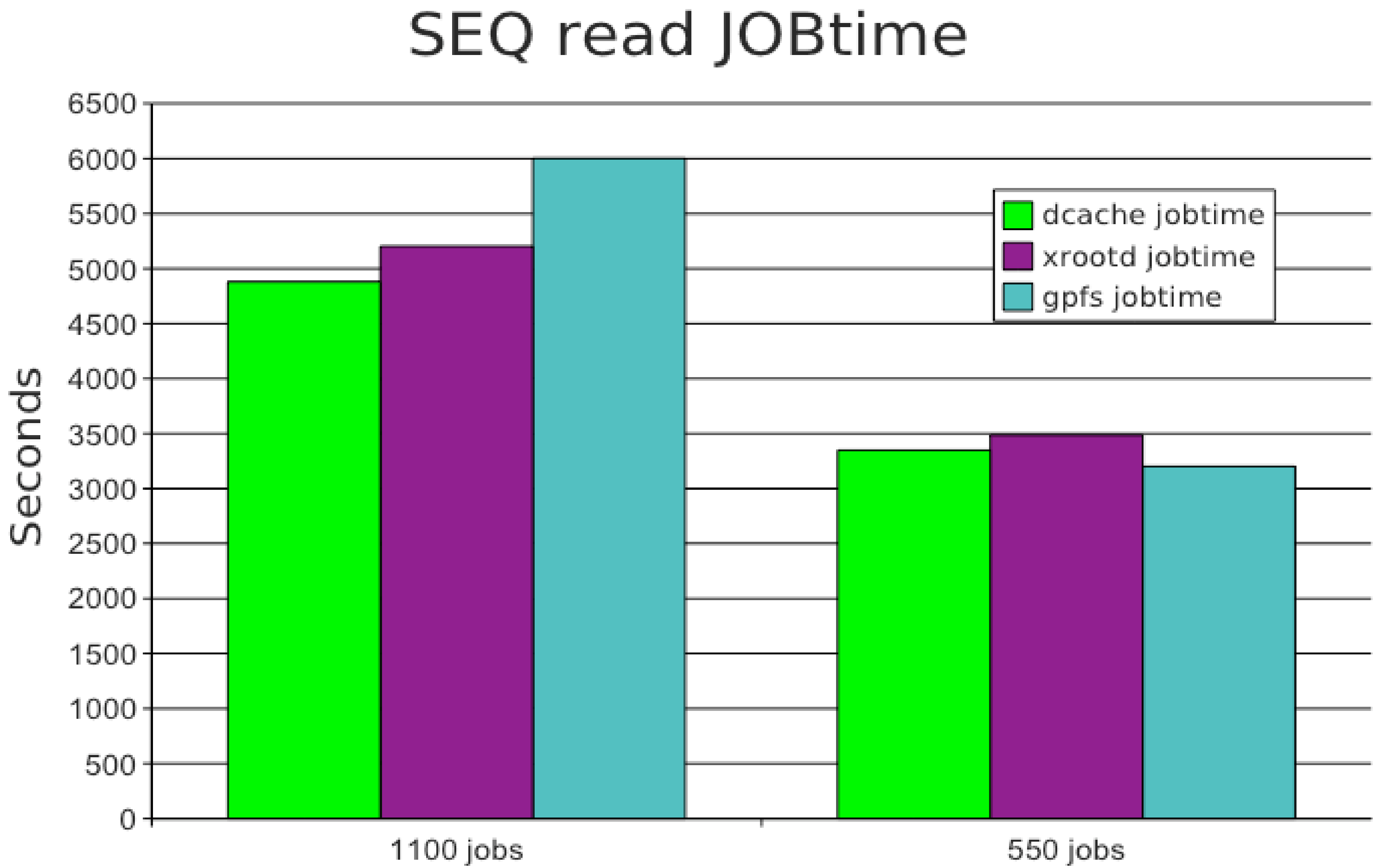
Access to the data



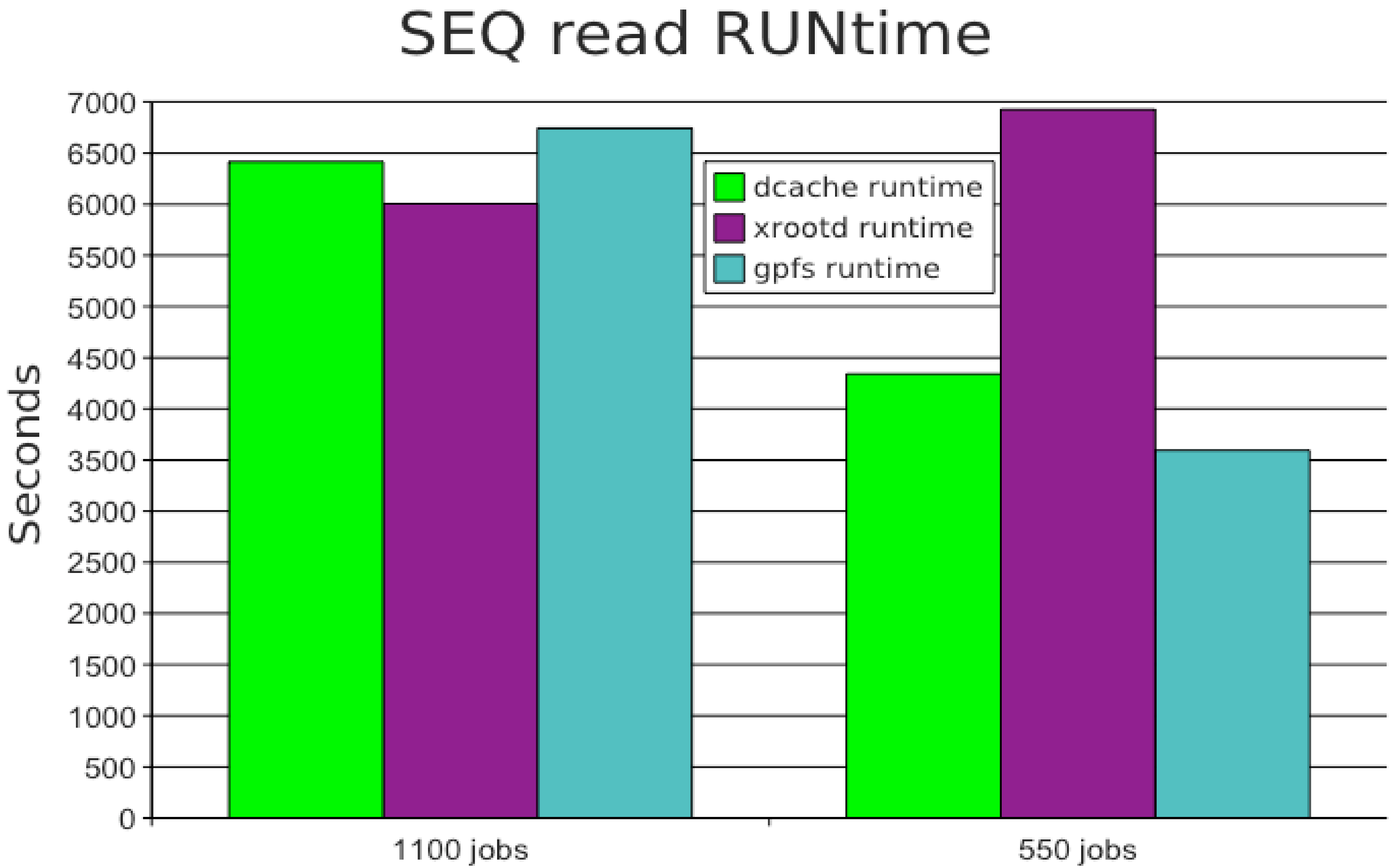
Access to the data



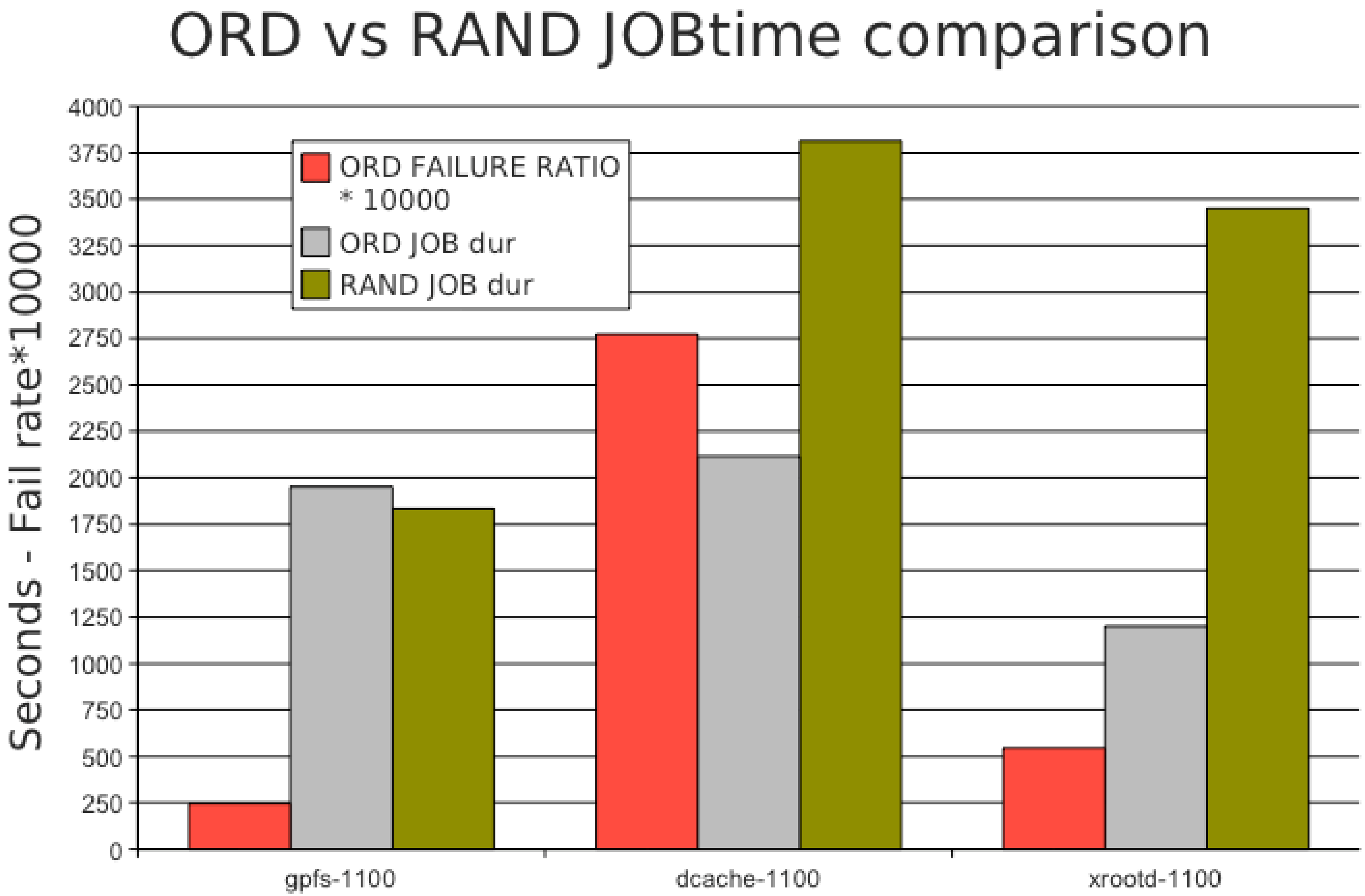
Pure sequential pattern, 2GB per job



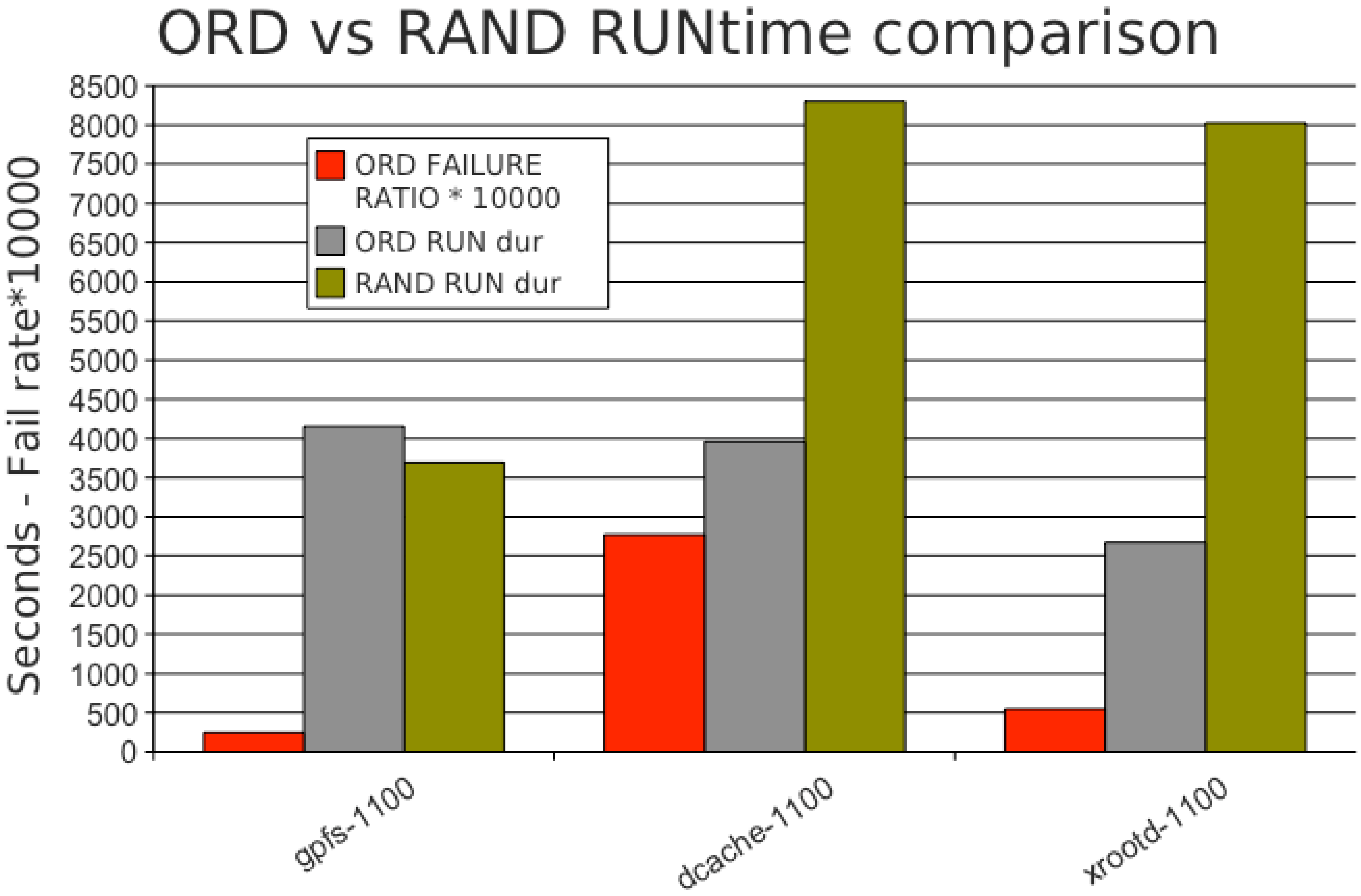
Pure sequential pattern, 2GB per job



Access pattern impact



Access pattern impact



Final thoughts

- These test are focused only on a specific (and very important) feature
  - Each of these complex software has many other aspect to be evaluated
  - The condition in which we tested the system is high loading
    - 5430 Files opened (concurrently)
    - 2 GB/sec of network bandwidth
    - 1.6 GB/sec of disc bandwidth
  - We surely reach the limit of the underline disk sub-system

Proposed Conclusions

- GPFS:
  - ✓ Excellent performance in all tests
  - ✓ Low failure rate in all tests
  - ✓ SRM available from INFN (StoRM: not usable in production at this moment because only SRMv2.2 is available)
  - ✗ Reduced performance when WN load is high (should be solved increasing the priority of GPFS daemon on WN)
  - ✗ Failure rate constant also with few concurrent jobs
- XRootD:
  - ✓ Good performance
  - ✓ Low failure rate in all tests
  - ✓ Installation and management quite easy
  - ✓ INFN is involved in developing
  - ✗ Failure rate constant also with few concurrent jobs (the problem should be fixed in next client release)
  - ✗ Better performance if client is well tuned
  - ✗ SRM and gridftp not available yet.

Proposed Conclusions

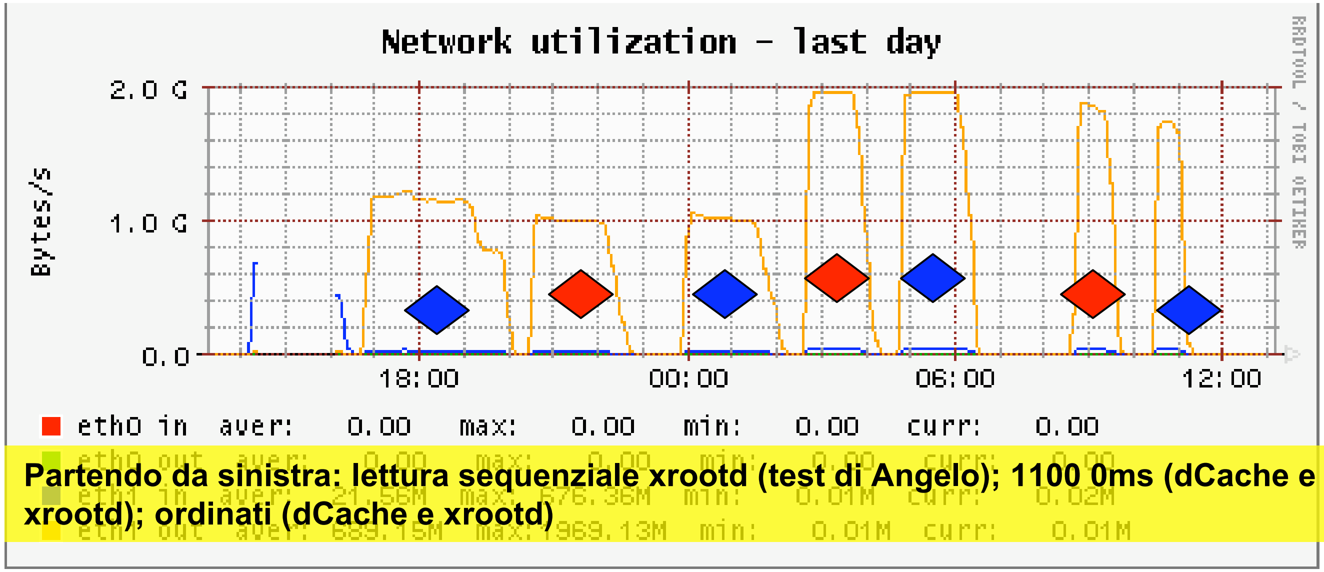
- dCache:
  - ✓ Good stability with few concurrent jobs (0 failure)
  - ✓ Complete and widely distributed solution for LHC T1/2 (with SRMv1/2, gsiftp, xrootd, tape management, etc)
  - ✓ The system is highly configurable to fit with the site needes
  - ✗ Large Failure rate when the load is too high (It is needed a large number of dcap doors to scale at this level of parallelism)
  - ✗ The default set-up is not always a good choice (can be changed at site-level)
  - ✗ Open time greater than the other system (intrinsic limit: “chimera” should solve this in next release)
- CASTOR:
  - ✗ The release in production is not able to pass this test: the next release will solve this problems

Credits

- **Network setup:** M. Bencivenni, D. Degirolamo, R. Veraldi, S. Zani
- **Farm setup:** A. Italiano, D. Salomoni
- **Monitoring setup:** F. Rosso, D. Vitlacil
- **Storage hw setup:** A. D'apice, PP. Ricci, V. Sapunenko
- **Storage systems setup:** G. Donvito, A. Fella, F. Furano, G. Lore, V. Sapunenko ,D. Vitlacil
- **Storage systems tests:** A. Carbone, L. dell'Agnello, G. Donvito, A. Fella, F. Furano, G. Lore, V. Sapunenko, V. Vagnoni

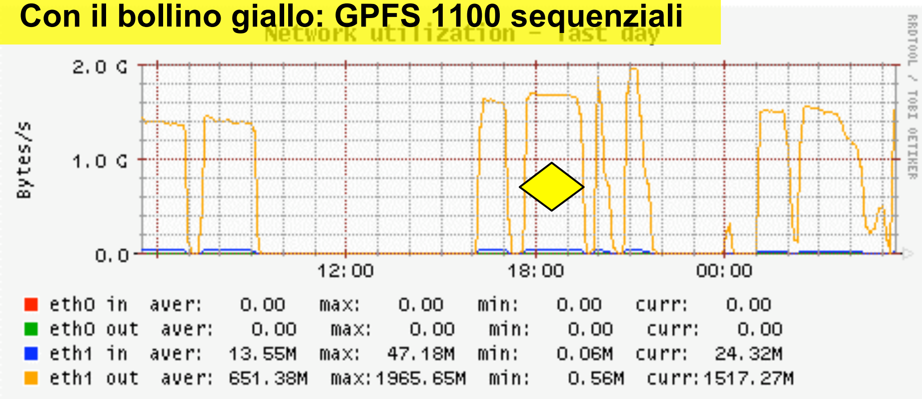
- **Storm development team:** A. Forti, L. Magnoni, R. Zappi
- **Storm tests:** E. Lanciotti, R. Santinelli, V. Sapunenko

Back-up Slides



Partendo da sinistra: lettura sequenziale xrootd (test di Angelo); 1100 0ms (dCache e xrootd); sequenziali (dCache e xrootd); ordinati (dCache e xrootd)

Con il bollino giallo: GPFS 1100 sequenziali



GPFS, nei sequenziali va peggio degli altri per un effetto dovuto alla cache del Sistema operativo che gli altri due sistemi possono usare a differenza di GPFS.

(quindi il risultato va preso un po' con le molle)

	Seq	Prot	Owner	Proc	Pnfsld	Pool	Host
in	1	dcap-3	cms032	23269	000100000000000000873D28	gridse03_3	gridfirb7.ba.infn.it
in	13	dcap-3	cms032	23269	000100000000000000874000	gridse03_3	gridfirb7.ba.infn.it
in	1	dcap-3	cms032	24912	00010000000000000086FF98	gridse03_3	gridfirb7.ba.infn.it
in	12	dcap-3	cms032	24912	000100000000000000870858	gridse03_1	gridfirb7.ba.infn.it
in	2	dcap-3	cmsprd	15937	0001000000000000009AF598	gridse03_3	pccms12.cmsfarm1.ba.infn.it
in	2	dcap-3	cms022	20913	0001000000000000008B86F0	gridse02_2	alicegrid09.ba.infn.it
in	8	dcap-3	cms022	20913	0001000000000000008B8808	gridba6_2	alicegrid09.ba.infn.it
in	2	dcap-3	cms022	15709	0001000000000000008D6E70	gridse03_3	alicegrid10.ba.infn.it
in	8	dcap-3	cms022	15709	0001000000000000008D6B80	gridba6_2	alicegrid10.ba.infn.it
in	2	dcap-3	cms022	15708	0001000000000000008C11D8	gridba6_2	alicegrid10.ba.infn.it
in	8	dcap-3	cms022	15708	0001000000000000008C1220	gridse01_2	alicegrid10.ba.infn.it
in	2	dcap-3	cms022	14096	0001000000000000008F4D80	gridse03_1	pccms32.ba.infn.it
in	2	dcap-3	cms022	15192	0001000000000000008F4D80	gridse03_1	pccms32.ba.infn.it
in	20	dcap-3	cms022	15192	0001000000000000008F4D60	gridse03_3	pccms32.ba.infn.it
in	2	dcap-3	cms022	24870	0001000000000000008F4D80	gridse03_1	alicegrid09.ba.infn.it
in	20	dcap-3	cms022	24870	0001000000000000008F4D60	gridse03_3	alicegrid09.ba.infn.it
in	2	dcap-3	cms022	16231	000100000000000000904720	gridse01_2	pccms32.ba.infn.it
in	2	dcap-3	cms022	17257	000100000000000000904720	gridse01_2	pccms32.ba.infn.it
in	10	dcap-3	cms022	17257	000100000000000000907AD8	gridse01_3	pccms32.ba.infn.it
in	2	dcap-3	cms022	13664	000100000000000000904720	gridse01_2	alicegrid08.ba.infn.it
in	8	dcap-3	cms022	13664	0001000000000000009006E0	gridse02_2	alicegrid08.ba.infn.it