A Comparison of Data-Access Platforms for The BaBar and ALICE analysis Computing Models at the Italian Tier1

Armando Fella[1], Fabrizio Furano[2], Francesco Noferini[1], Luigi Li Gioi[1], Matthias Steinke[4],

Daniele Andreotti[3], Alessandro Cavalli[1], Andrea Chierici[1], Luca Dell'agnello[1],

Daniele Gregori[1], Alessandro Italiano[1], Eleonora Luppi[3],

Barbara Martelli[1], Pier Paolo Ricci[1], Elisabetta Ronchieri[1], Davide Salomoni[1],

Vladimir Sapunenko[1], Dejan Vitlacil[1]

1) INFN-CNAF 2) CERN 3) INFN-Ferrara 4) Ruhr-Universität

CHEP09 Prague March 21-27

Goals and Outline

- Comparative evaluation of data-access solutions used by the BaBar and ALICE experiments at CNAF
- Enabling ALICE at CNAF: reliability and performance measurements

Two analysis tasks were performed on real physics events using production CNAF systems

- **Pure Scalla/Xrootd**
- **GPFS**
- hybrid **Scalla/Xrootd over GPFS**

- in a real production environment
- Different job set sizes: up to 1000 jobs per data access solution

Real ALICE analysis jobs:

- 700 and 1000 parallel ALICE analysis jobs
- Two-particle correlations analysis on PbPb events (MC & full reconstruction-data like)

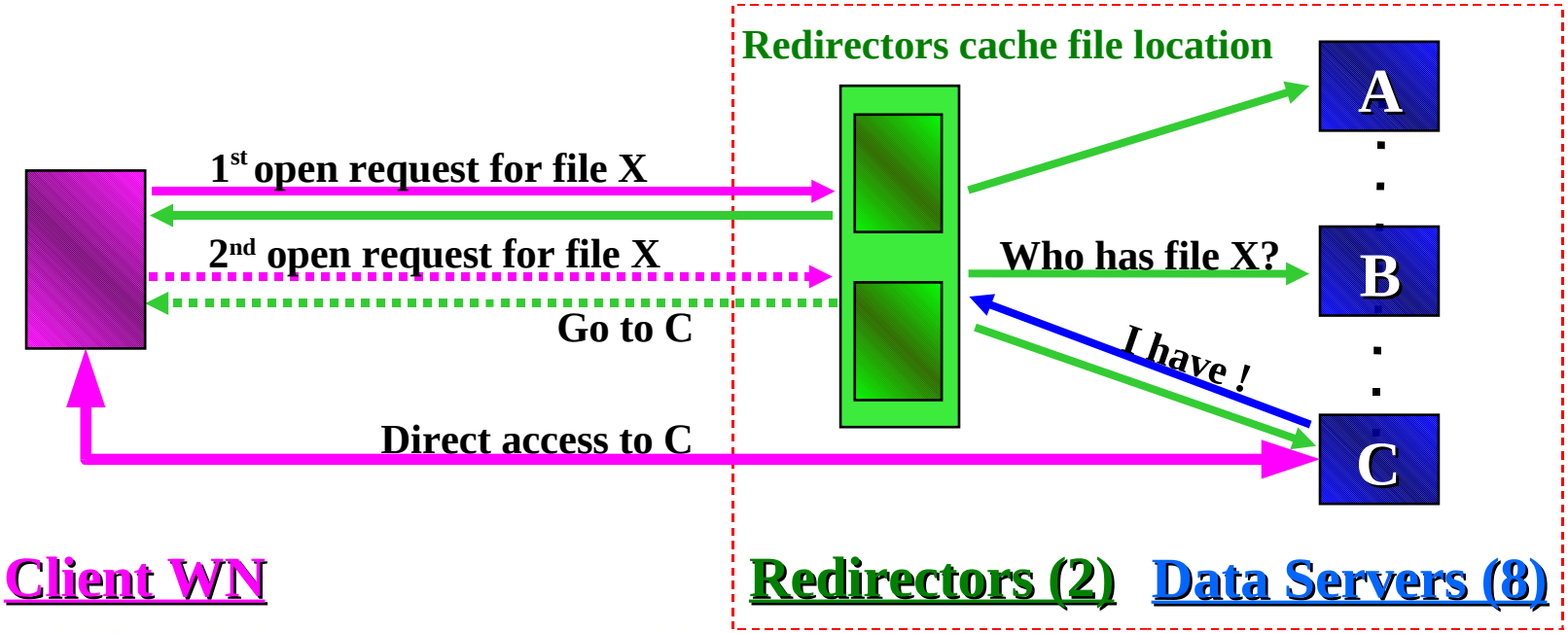
Real BaBar analysis jobs:

- 1000 parallel BaBar analysis jobs
- Selection of $D^{*+} \rightarrow [K_S\, \pi^+ \pi^-]_{D0}\, \pi^+$ on real BaBar data

Scalla/xRootd: Test Installation

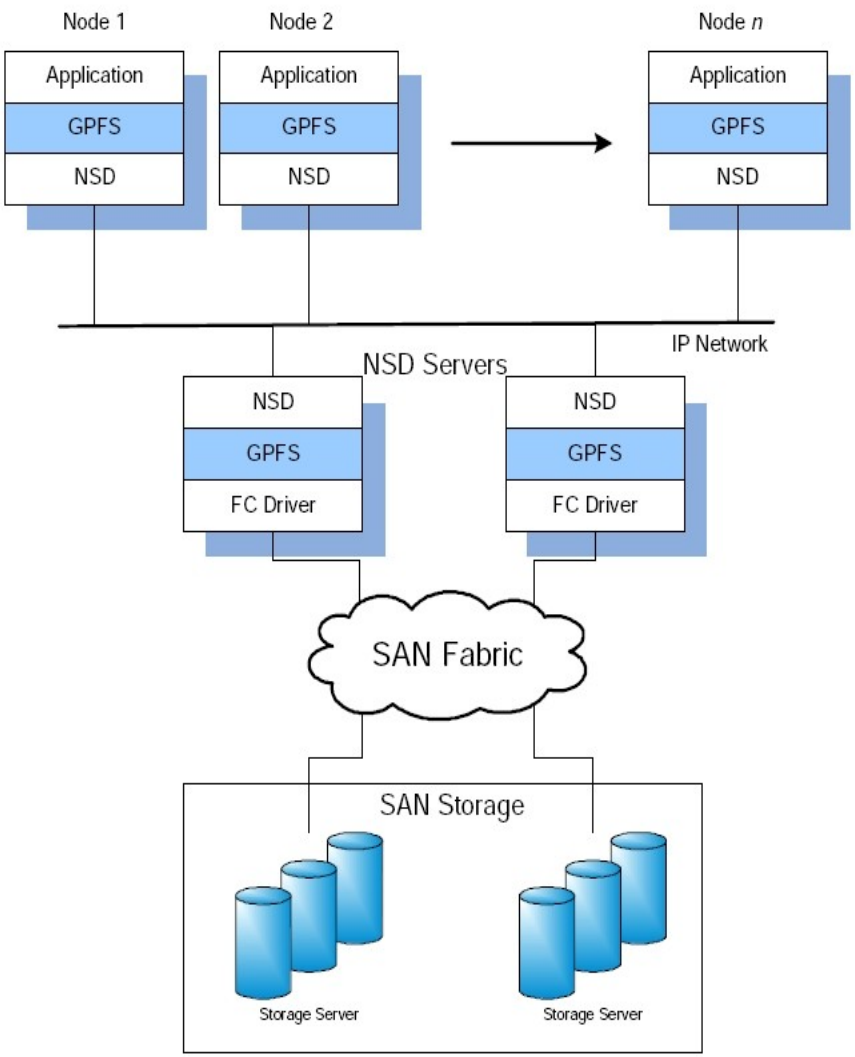
The Scalla/Xrootd platform is a pure disk data handling system developed as a SLAC and INFN collaboration. Scalla/Xrootd is designed to provide fault tolerant location and access to files distributed throughout a cluster, by employing peer-to-peer-like mechanisms.[http://xrootd.slac.stanford.edu]

- Scalla/xRootD latest production release 20090206.1632
- A total of 10 servers
  - **2 Redirectors** (DNS balancing) + **8 Data servers**
  - No data redundancy, No server redundancy
  - Plain default configuration per computing model
- 1 LUN (8TB) mounted per data server, XFS formatted
- Files have been distributed into dataservers disks in a round-robin fashion



General Parallel File System: Installation

GPFS is a general purpose distributed file-system developed by IBM. It provides file-system services to parallel and serial applications. GPFS allows parallel applications to simultaneously access the same files in a concurrent way, ensuring the global coherence, from any node which has the GPFS file-system locally mounted.
http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp



- GPFS production release 3.2.1-4
- A total of 10 servers involved:
  - 8 Data servers, 2 cluster admin
  - WNs and Frontend machine belonging to the farming cluster, data servers belonging the BaBar production cluster.
  - No data redundancy, No server redundancy
  - Plain default configuration
- 8 storage groups, 8 NSD daemon
- 8 TB per storage group = 64TB
- Each Client WN mounts the unique GPFS "partition" and performs I/O operations by posix like system calls

NSD = Network Shared Disk
SAN = Storage Area Network

Data Handling configurations

The experiment's testbeds share**:**

- common GPFS client and server installation and configuration
- the server side Scalla/xRootd and the client side ROOT installation

Default ALICE configuration

Server side

Monalisa monitor **enabled**
ALIEN Authentication mechanism **on**
Global redirector infrastructure **on**
TTreeCache optimizations **on**

Client side

ROOT parameters
Read-ahead: 500KB
Cache size: 10MB

Default BaBar configuration

Server side

Monalisa monitor **disabled**
ALIEN Authentication mechanism **off**
Global redirector infrastructure **off**
TTreeCache optimizations **off**

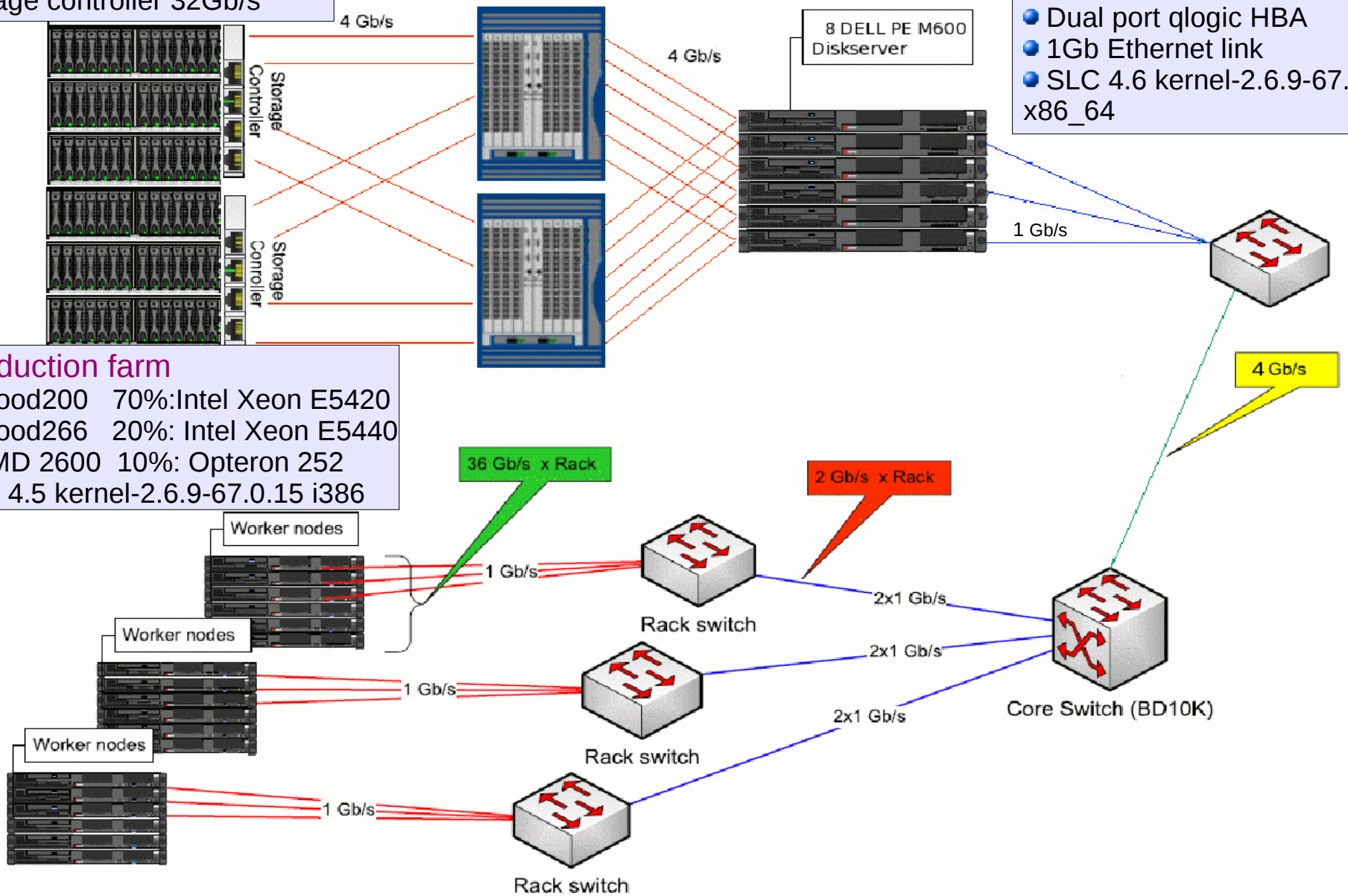
Client side

ROOT parameters
Read-ahead: 0KB
Cache size: 0KB

Test-bed layout

1 EMC CX-380 SAN system:
- 160 TB data disk raw
- 1TB SATA disks
- Storage controller 32Gb/s

8 Disk-server Dell PE M600 enclosure,PE M100B blade:
- 2.33 Gz Intel Xeon dual 4 core
- 16 GB ram
- Dual port qlogic HBA
- 1Gb Ethernet link
- SLC 4.6 kernel-2.6.9-67.0.15 x86_64



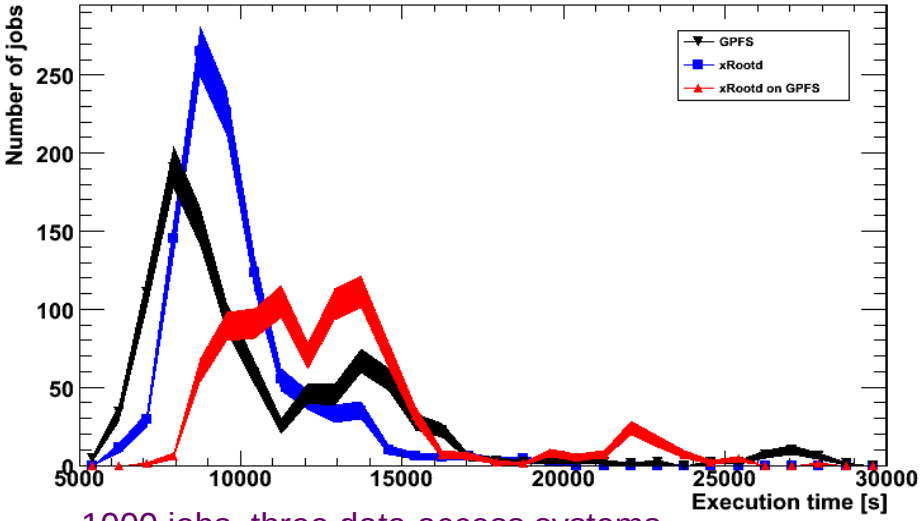
Production farm
- Wood200 70%:Intel Xeon E5420
- Wood266 20%: Intel Xeon E5440
- AMD 2600 10%: Opteron 252

SLC 4.5 kernel-2.6.9-67.0.15 i386

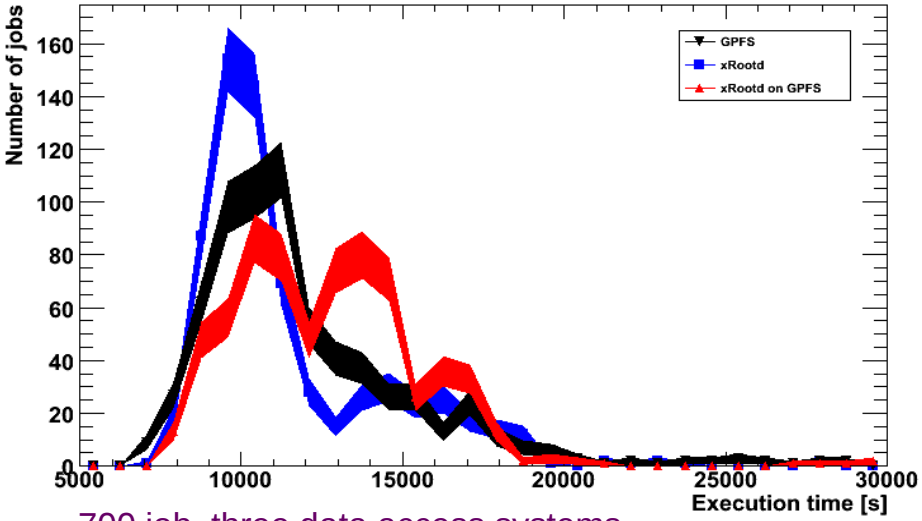
CNAF basic services shared by test bed: DNS, LSF batch system, accounting

ALICE Job time distribution

Execution time distributions per Data Access system with 1000 and 700 jobs run.



1000 jobs, three data access systems

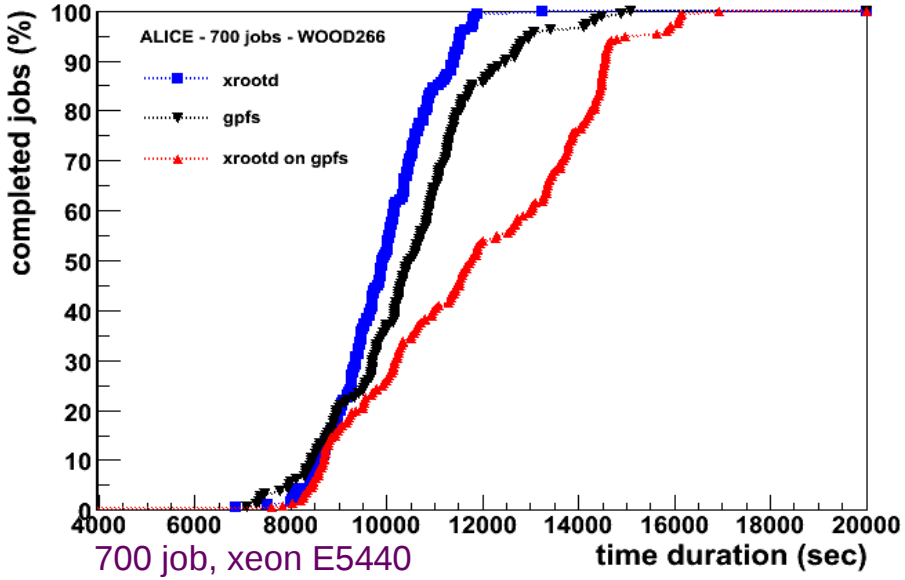


700 job, three data access systems

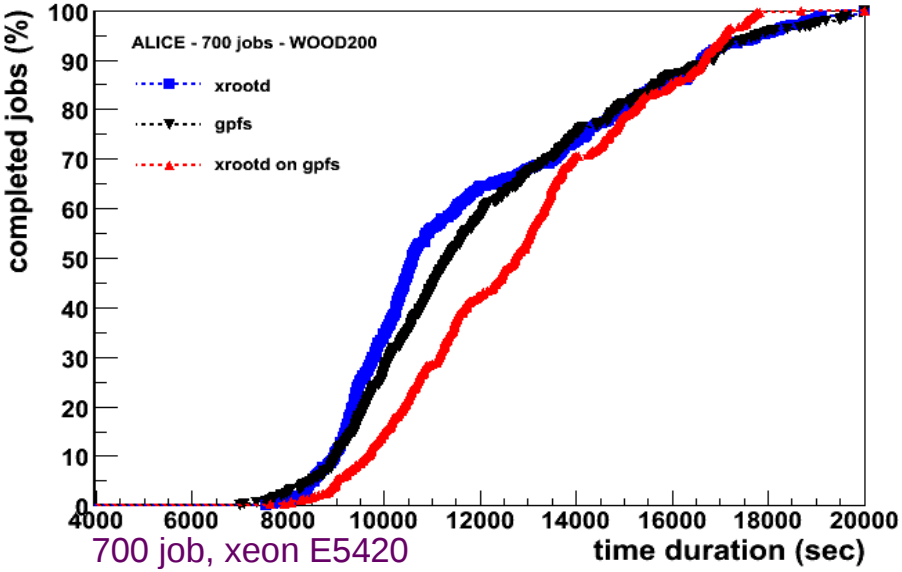
- Run 700 results 20% longer because it ontains in average jobs analysing a larger amount of data.
- In both graphs the xRootd and GPFS distributions are similar, since we are in a real production scenario.
- xRootD over GPFS appears to suffer a bit, the effect is a two-peaked shape

ALICE job time duration per WN arch

Job time duration distributions per Data Access system running on the three involved Worker Nodes HW architectures at CNAF.

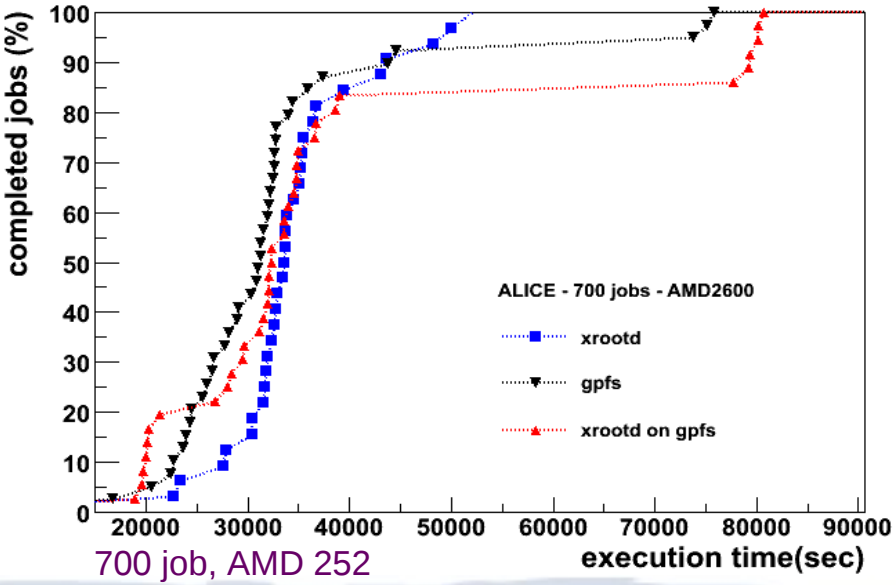


700 job, xeon E5440



700 job, xeon E5420

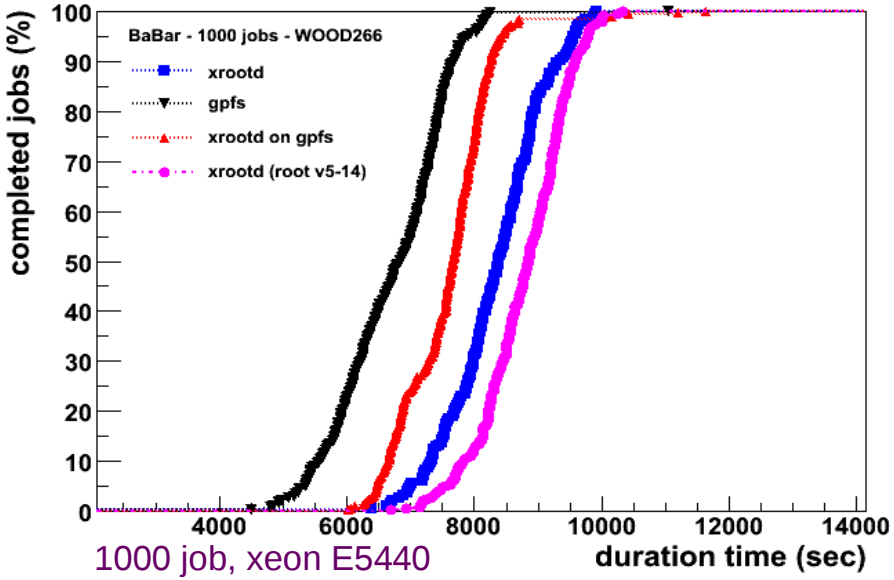
- The three Data Access systems work in a very similar way from the point of view of the job duration for the three worker node architectures.
- ALICE uses all the I/O advanced features in ROOT

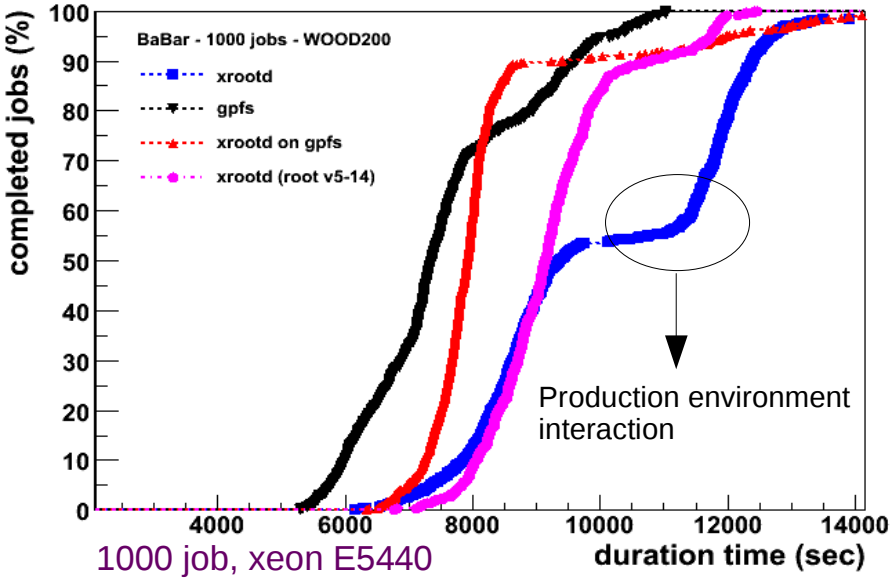


700 job, AMD 252

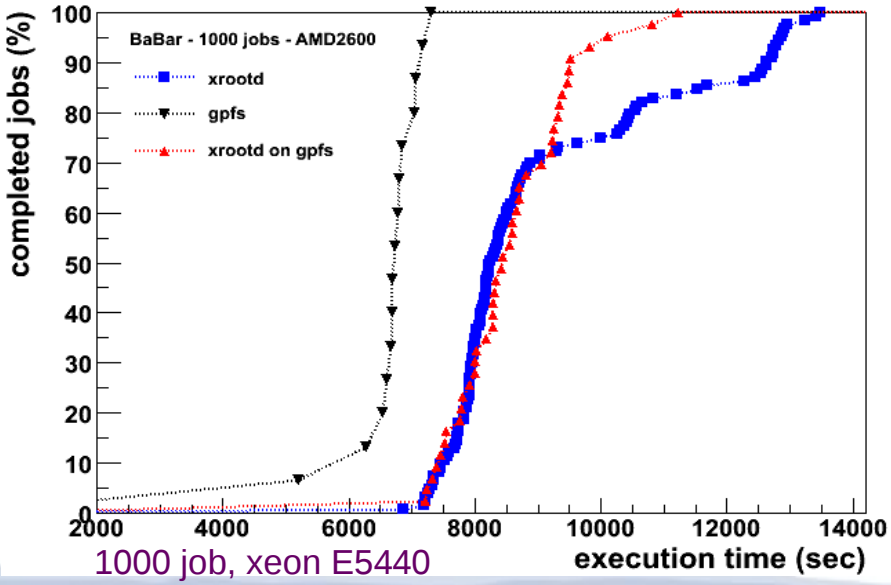
BaBar job time duration per WN arch

Job time duration distributions per Data Access system running on the three involved Worker Nodes HW architectures at CNAF.



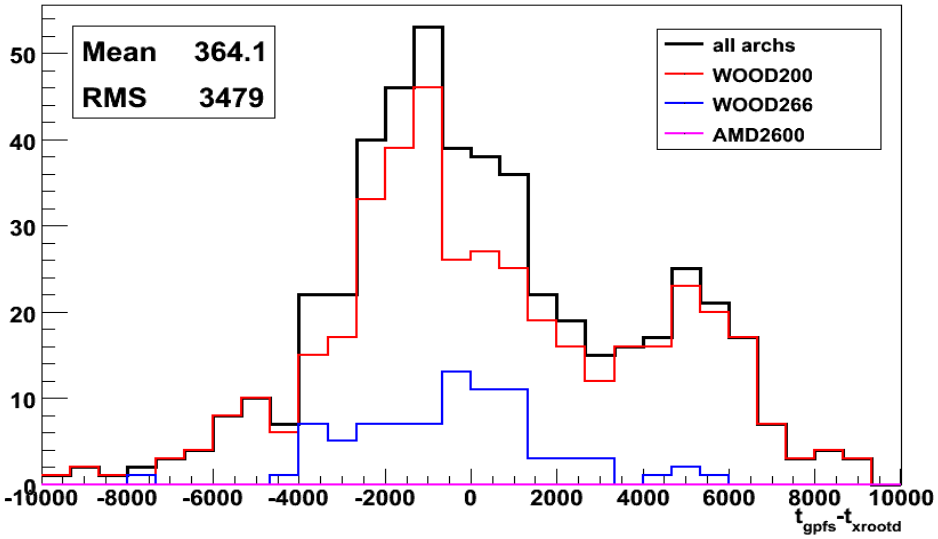


- The (per-computing model) lack of read-ahead and cache for xRootD makes its performance worse.
- GPFS (with its caching) seems faster then xRootD in all the architectures cases.
- In the case of slow worker node architecture the GPFS and xRootD over GPFS curves present similar results.

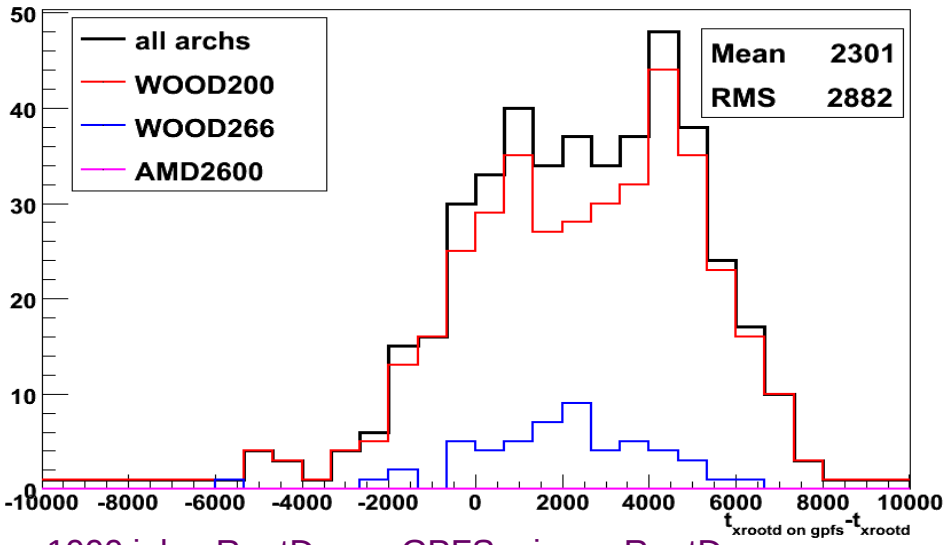


WallClockTime difference ALICE

The job WallClockTime duration difference distribution per Worker Node architecture respectively between GPFS and xRootD runs, and xRootd over GPFS and xRootD runs of 1000 ALICE jobs.



1000 job, GPFS minus xRootD



1000 job, xRootD over GPFS minus xRootD

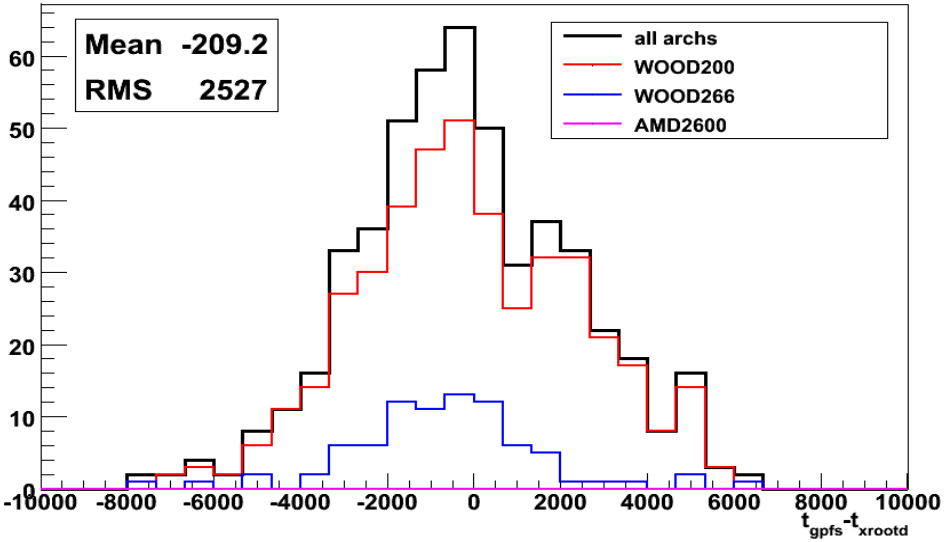
- A statistical analysis on identical jobs running on the three HW architectures. The third architecture is not represented because there are no jobs in the compared runs that read the same collections of data.

- The first graph shows similar performance for GPFS and xRootD for each job run in the three hardware architectures.

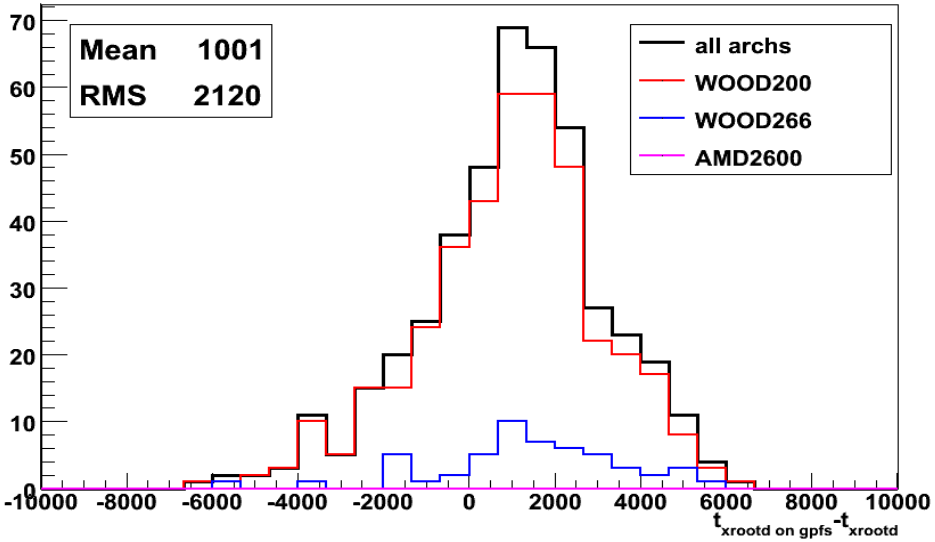
- The graph on the right shows a worse performance of xRootD over GPFS with respect to pure xRootD.

CPUTime difference ALICE

The job CPUTime duration difference distribution per Worker Node architecture respectively between GPFS and xRootD runs, and xRootd over GPFS and xRootD runs of 1000 ALICE jobs.



1000 job, GPFS minus xRootD

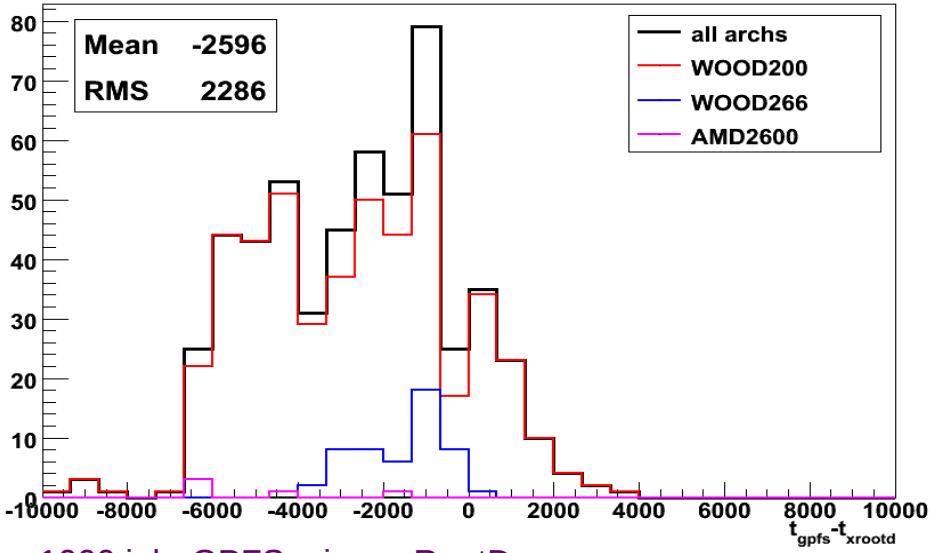


1000 job, xRootD over GPFS minus xRootD

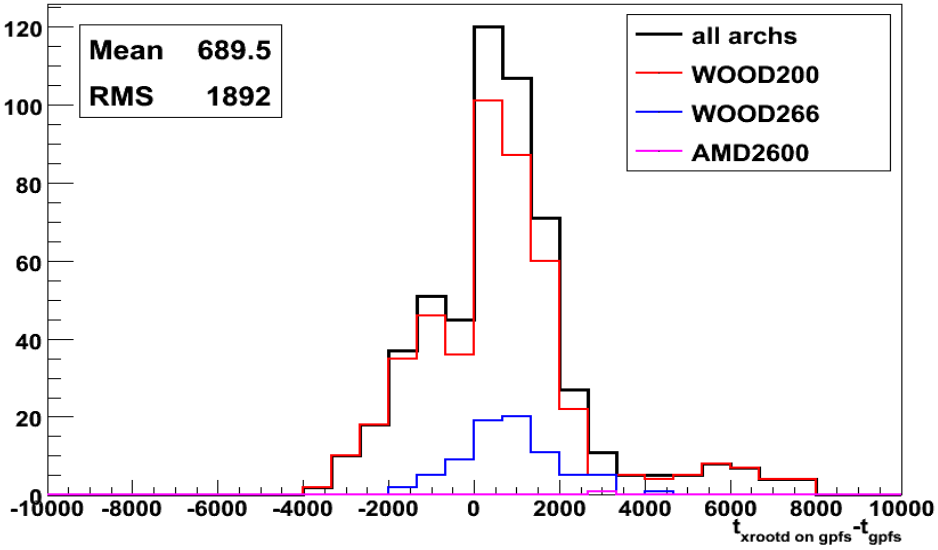
- The case of CPUTime analysis looks in general “cleaner” respect to the WallClockTime one in which the time spent in data transfer impacts to the graph shape smearing it.
- The focusing on CPUTime reveals the xRootD and GPFS client side comparison.

WallClockTime Difference BABAR

The job WallClockTime duration difference distribution per Worker Node architecture respectively between GPFS and xRootD runs, and xRootd over GPFS and xRootD runs of 1000 BaBar jobs.



1000 job, GPFS minus xRootD

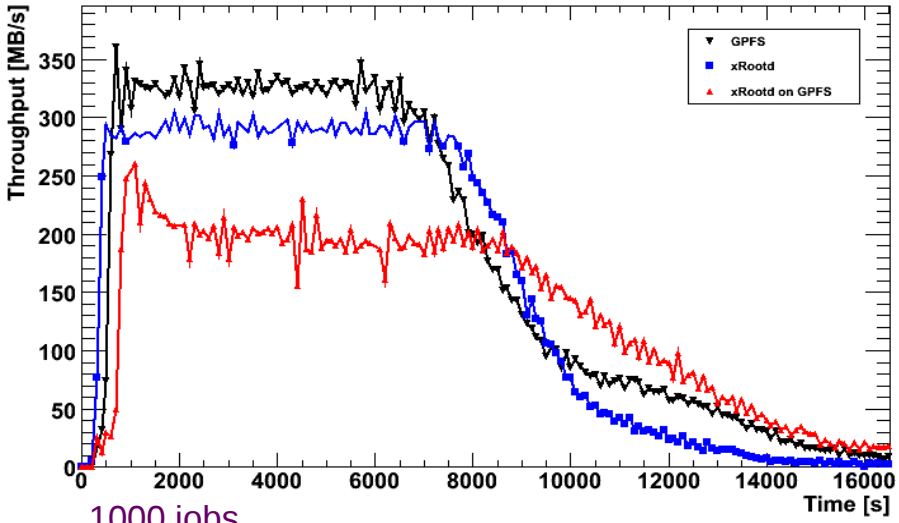


1000 job, xRootD over GPFS minus GPFS

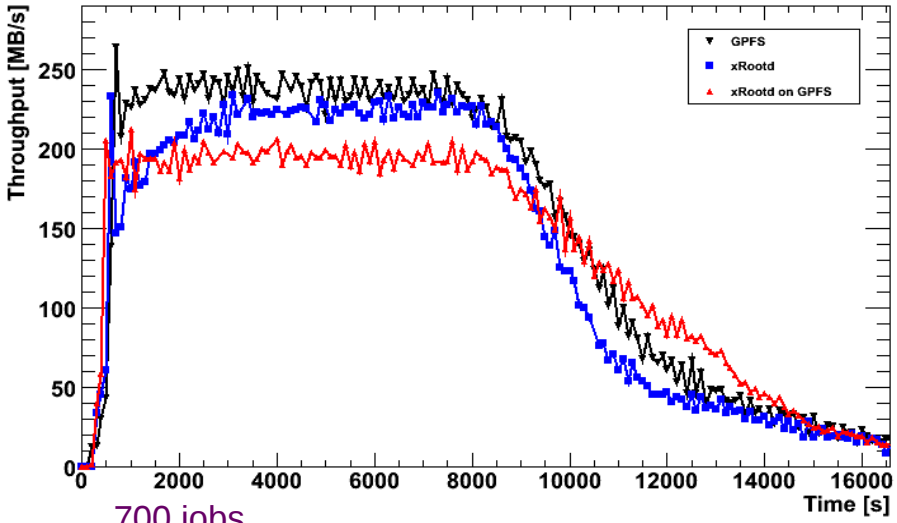
- The GPFS run overcomes the comparison with the pure xRootD one that suffered of a suspect production environment interaction
- In the right graph the case xRootD over GPFS is ~10% slower then GPFS

Throughput ALICE

The throughput distribution per Data Access system respectively for ALICE 1000 and 700 jobs run



1000 jobs

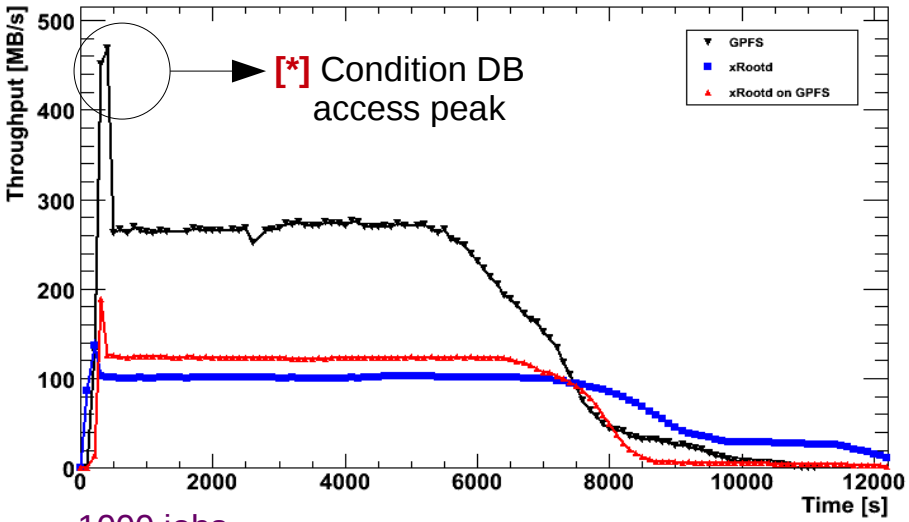


700 jobs

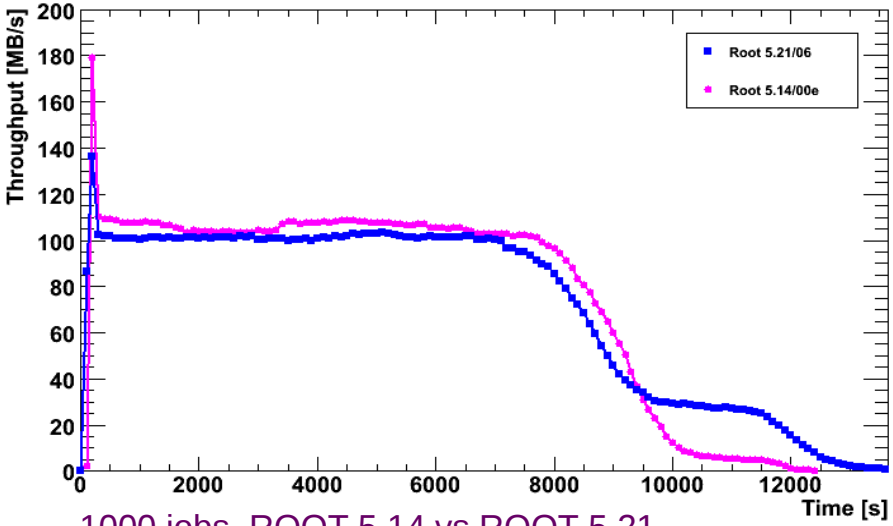
- The curves of 1000 jobs graph have similar integral values (see slide 15).
- The run xRootD over GPFS of 1000 jobs graph had an unlucky architecture job distribution
  - Run time higher then other runs (~20%)
  - affected by 6% of fail rate.
- The three curves represented in 700 jobs graph are substantially equivalent.

Throughput BaBar

The throughput distributions for 1000 jobs running against xRootD, GPFS and xRootD over GPFS, and the throughput distributions for 1000 jobs running against xRootD, using at client side two different ROOT versions, v5.14 and v5.21



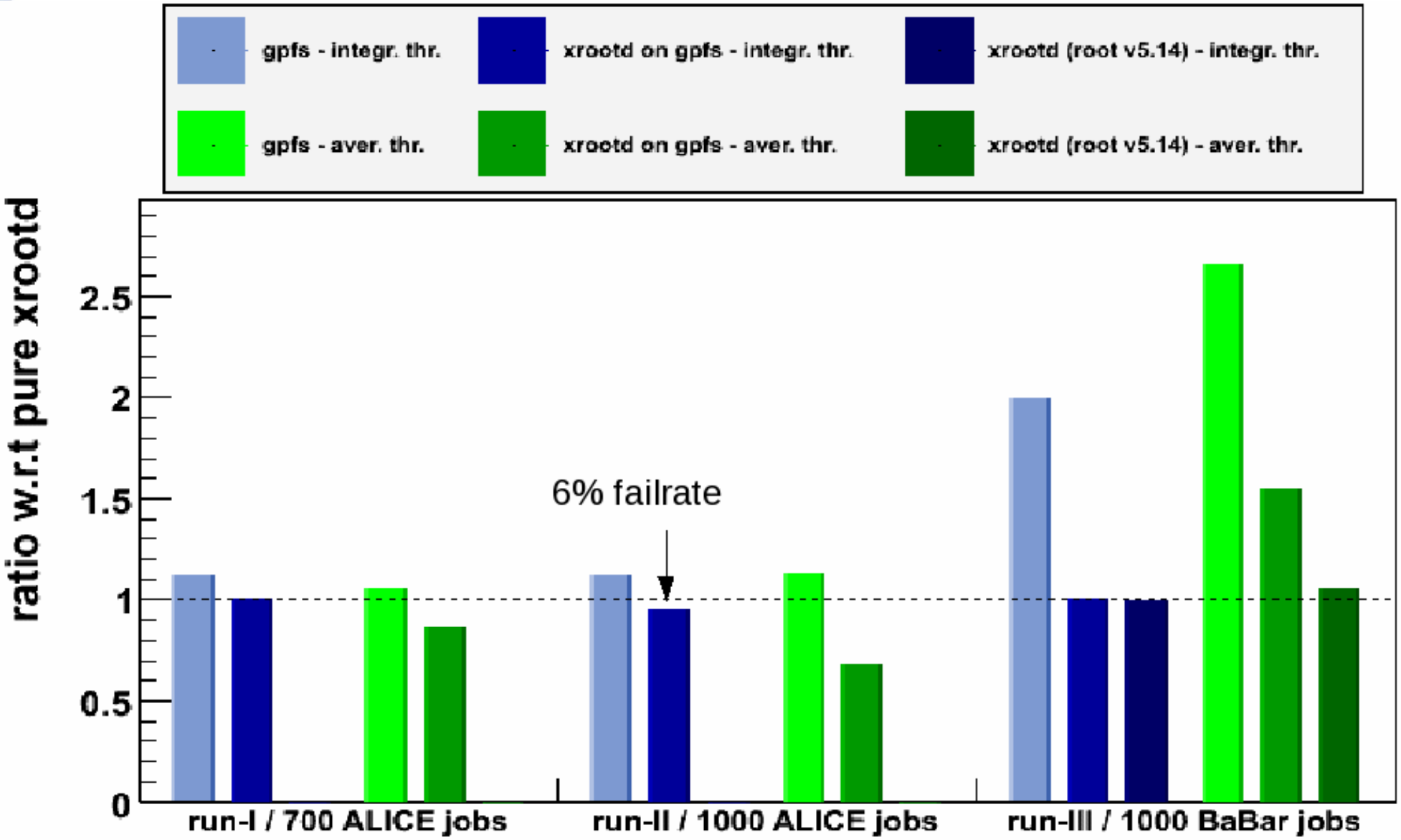
1000 jobs



1000 jobs, ROOT 5.14 vs ROOT 5.21

- The xRootd throughput curve integral and plateau values are less than 50% than the GPFS ones running the same analysis jobs.
  - This behavior is due to the GPFS "Prefetch" algorithms, shown to be not efficient in the case of not-sequential reads, like a real data analysis process.
- The BaBar production release includes at present time ROOT 5.14. It was of interest the comparison of ROOT releases 5.14 and 5.21: the graph shows that the ROOT clients throughput distributions are almost equivalent.
- **[*]** Initial access to the condition database caused the first large "read-ahead" reading.

Network load summary



Summary

- The test was performed in a “pure” production environment, with the usual difficulties : heterogeneous worker node architectures, production and test jobs mixed on the same client machine and shared base services (network, authentication), but permitted to collect information about a working scale representative for a Tier1/2.

- This work shows that there are not huge differences in the scenarios we have exercised, especially looking at the results obtained with newer worker nodes.

- The different xRootd setting of the prefetch algorithms between ALICE and BaBar results in different bandwidth usage respect to GPFS: setting to zero the read-ahead value, xRootd uses about one half of the bandwidth respect to GPFS in the BaBar case. The bw usage in the ALICE case instead is quite similar in all cases.

- The ALICE and BaBar Data Access models are in close relation, but the more modern ALICE usage ends to be more efficient (ROOT TTreeCache + XrootD asynchronous data reads).

- The XrootD client tends to use more cpu resources then the GPFS one.

- The IOWait values in the XrootD server in two occasion went up to ~20%

- For BaBar no performance difference between the ROOT versions 5.14 or 5.21