

Future GSI analysis facility: Benchmarking with analysis trains

Federica Sozzi



ALICE offline week, CERN
8-10 November 2017

For Run 3 the analysis will be organized following the established analysis train model in **dedicated sites: Analysis Facilities (AF)**

AODs collected in few sites specialized for the analysis, to optimize performance:

4PB of AODs in a 12 hours period

[from TDR: 5000 job slots with peaks of 20000:

20000 jobs - 5MB/s for job – aggregate throughput of 100GB/s]

At GSI a first analysis facility prototype based on the existing ALICE T2 setup is being finalized

Analysis Facility Prototype at GSI: a very flexible setup



ALICE

usual interface to GRID

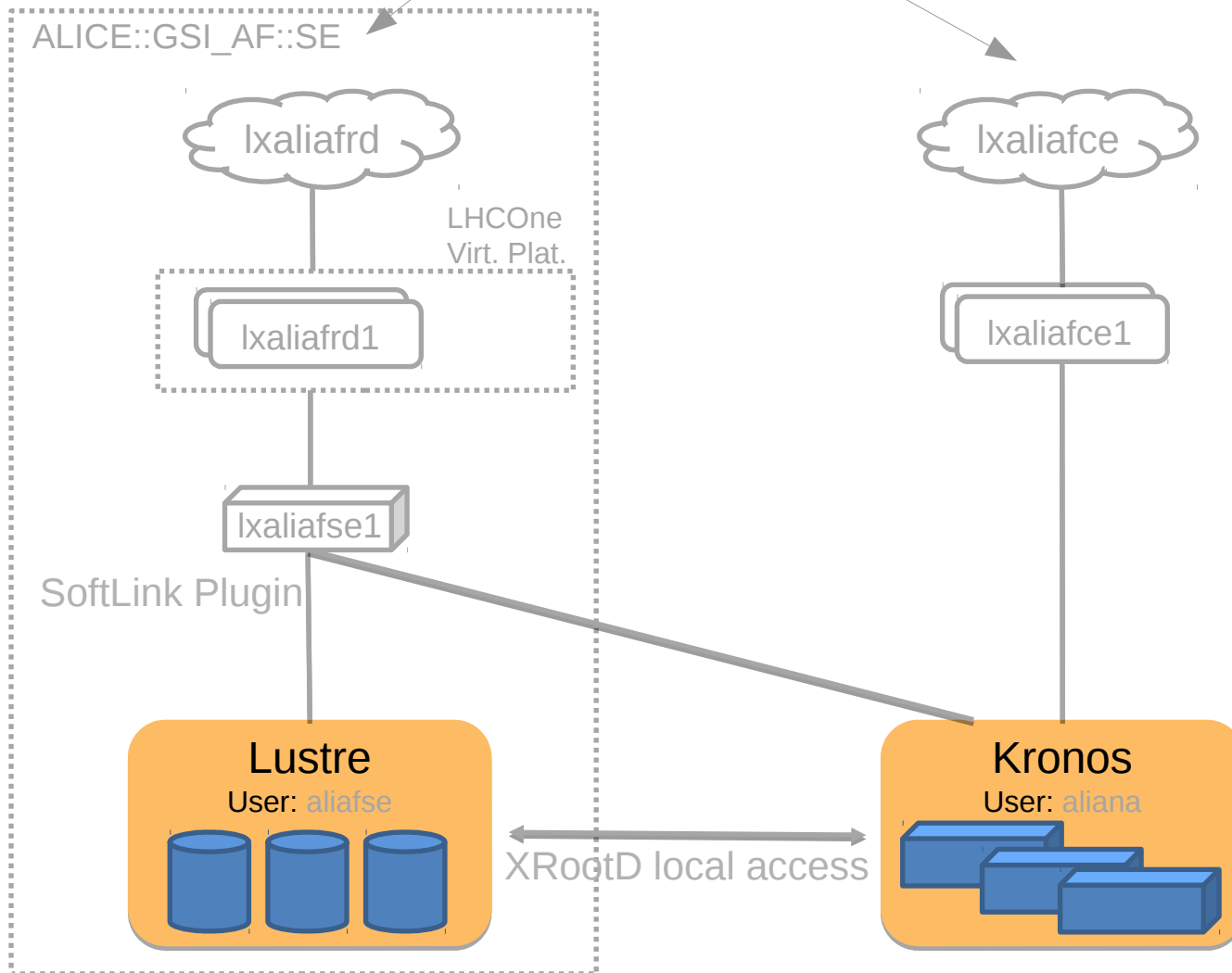
new ALICE Storage Element and Compute Element

Disk storage elements in Monalisa:

16. GRIF_IRFU - DPM	ALICE::GRIF_IRFU::DPM	593 TB
17. GSI - AF_SE	ALICE::GSI::AF_SE	600 TB
18. GSI - SE2	ALICE::GSI::SE2	2.3 PB

Initial Resources

- up to 1000 job slots (taken from Tier 2 allocation)
- Initial dataset AOD194LHC15o



local access

XRootD Client Plug-in -
XrdOpenLocal:
Clients should open a file
directly from Lustre if at GSI

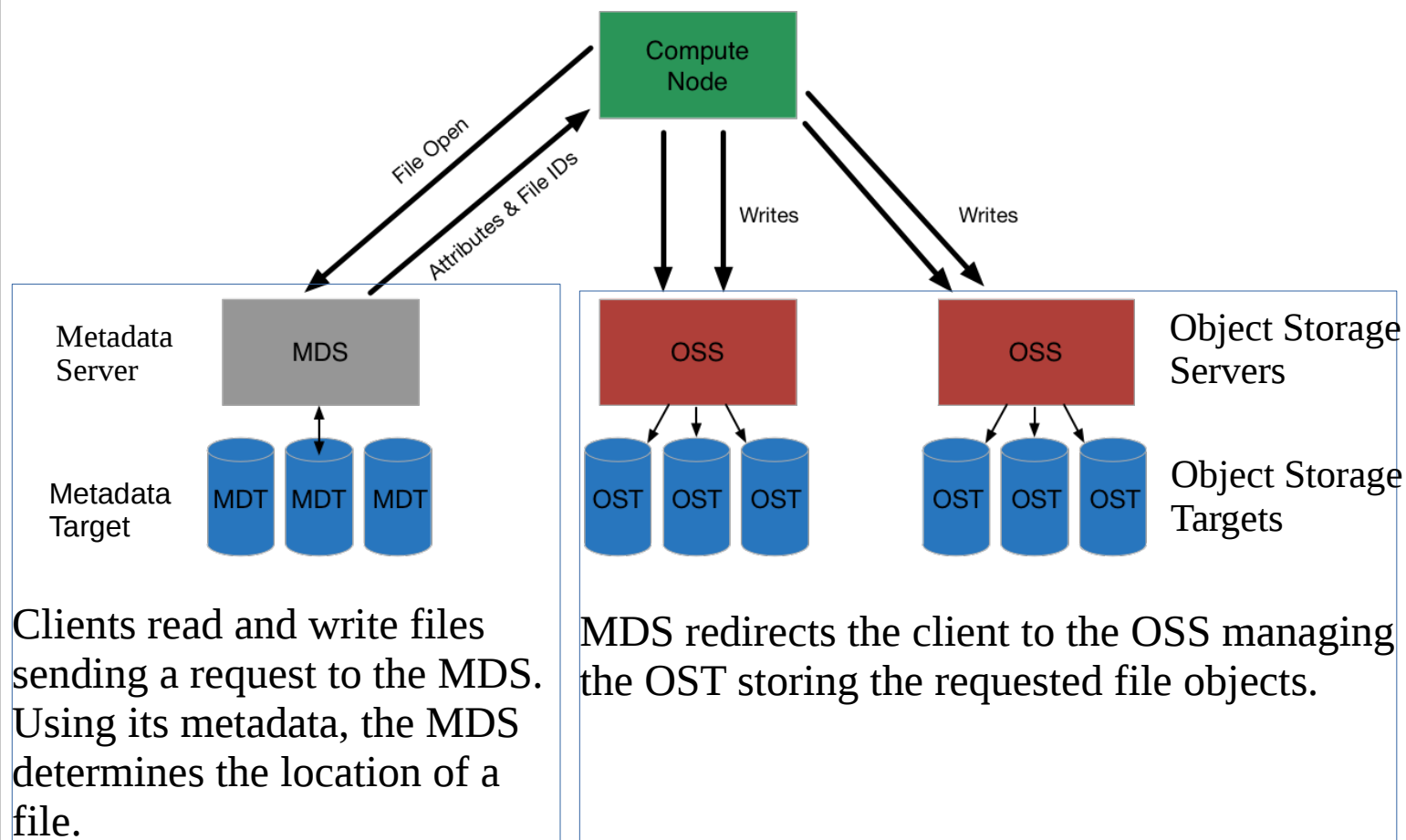
XrdOpenLocal is now also
available as
Server/Redirector Plug-In

version 4.8
(Bachelor thesis Paul-Niklas
Kramp)

Kronos & nyx cluster at GSI

Job scheduler: Slurm

Storage system: lustre v 2.6.92 upgrading to v 2.10



At GSI:
546 OST
7 OST for 1 OSS

Aggregated
bandwidth
425GB/s

While the prototype is being finalized, we prepared a benchmark and run it on the current GSI cluster

- to have a baseline: measure the present data analysis speed
- to measure and compare the performances after changes in the setup and software

Data sample used: some AOD Pb-Pb LHC15o pass1 available locally

Measurements:

- Data reading speed: copying files to */dev/null*
- Unzipping speed
- Data analysis speed: analysis in local mode and using the local train framework

Copying files to /dev/null



Copy done directly in the terminal and through batch jobs (no difference)

70 measurements, values varying a lot: [0.11, 12.3] GB/s,
most frequent value around 1.2 GB/S

No systematic measurement over long period of time, measurements done over few hours

Unzipping files

Measure of the unzipping procedure, using “unzip” command in the terminal:

Average: 101 MB/s

Range: [75, 116] MB/s

Analysis speed: local analysis



Very simple task used, filling a p_T spectra (*taken from the ALICE tutorial – Author Redmer Alexander Bertens - with few modifications*)

Local test with the macro *runAnalysis.C*, reading each time 20 AOD files (~18 GB)

25 measurements in batch system:

Average: 24.53 MB/s

Range [9.75, 33.44] MB/s

Analysis speed: local analysis with train framework



Task added on a simplified version of the LEGO train, able to run locally at GSI

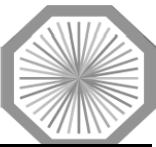
Tests with different number of jobs, each job reading 20 AOD

Merging disabled

Files *syswatch.log* are produced for each job, these are the input for the benchmark (python script)

When removing the task the results are almost unchanged → this is a measure of the framework speed

Benchmark output (1/2)



Outcome of the benchmark:
information displayed
on the terminal ...

Train Statistics

Train started at 2017-10-05 15:02:31

Train ended at 2017-10-05 15:28:25

Number of submitted jobs: 500

Finished: 500 (100.0%)

Finished without errors: 500 (100.0%)

Data processed [GB]: 9014.6

Processing rate [MB/s]: 5800.5

Job statistics

cpu time [s] : 917.8 [654.9,1424.1]

wall time [s] : 1067.6 [765.5,1537.5]

num. events : 37924.0 [29000.0,77000.0]

num. files : 20.0 [20.0,20.0]

virtual mem [MB] : 993.0 [937.8,1673.9]

res memory [MB] : 530.5 [459.5,1194.8]

data processed [GB] : 18.0 [12.5,24.1]

processing rate [MB/s]: 17.0 [13.3,22.0]

Train processing rate (aggregate throughput):

evaluated as
sum of the data processed in
the jobs / total time duration of
the train

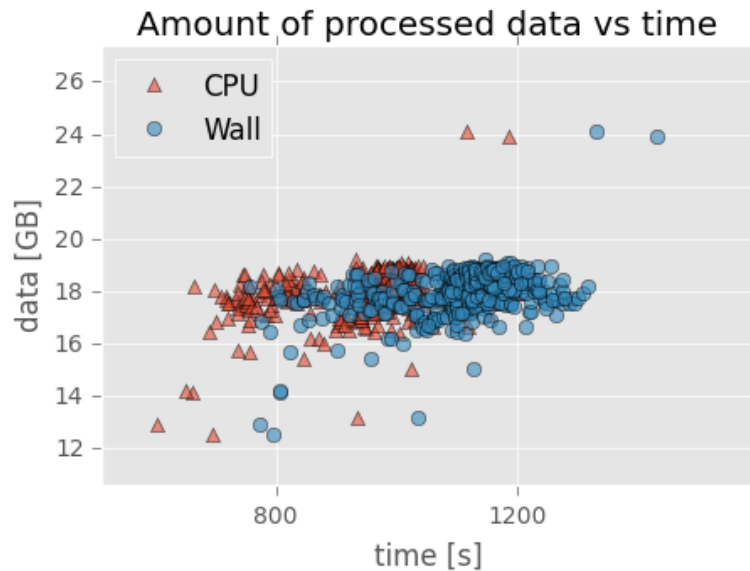
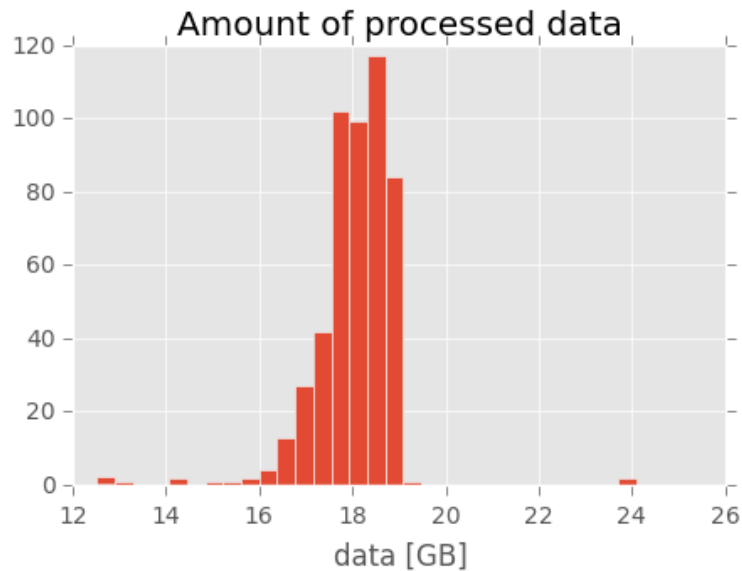
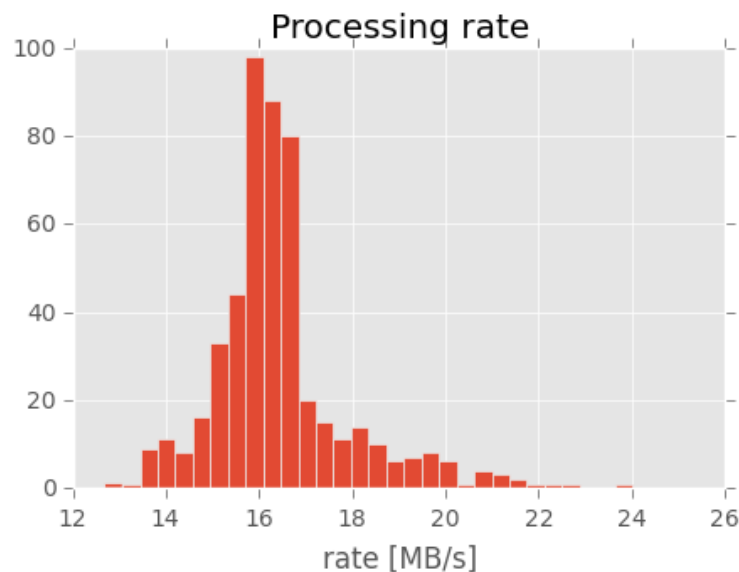
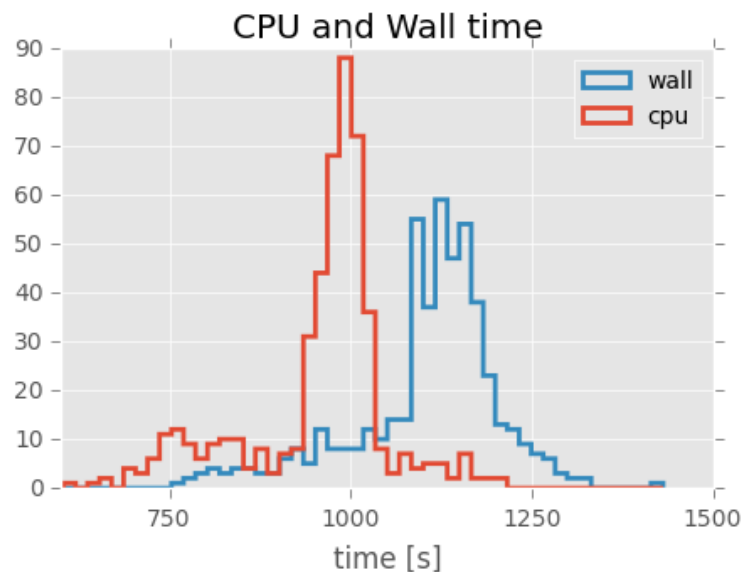
*(This variable will be used again
later)*

Average job processing rate

Same order of magnitude of the
rate measured in the previous test,
a factor ~1.7 smaller (more results
in the next slides)

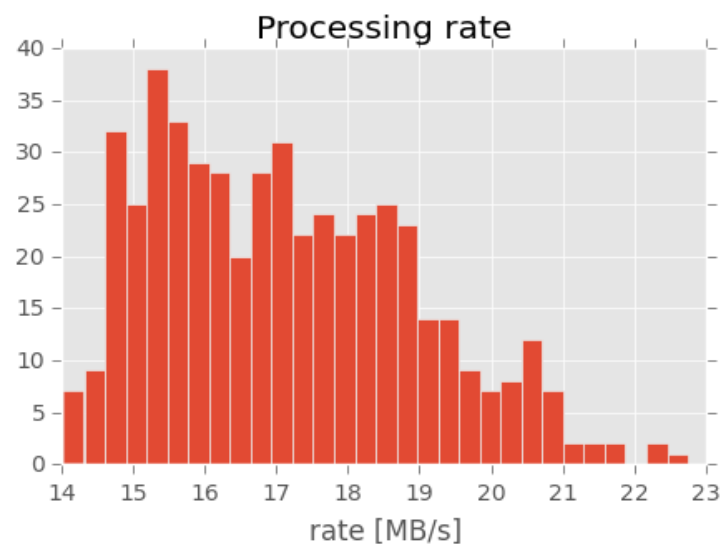
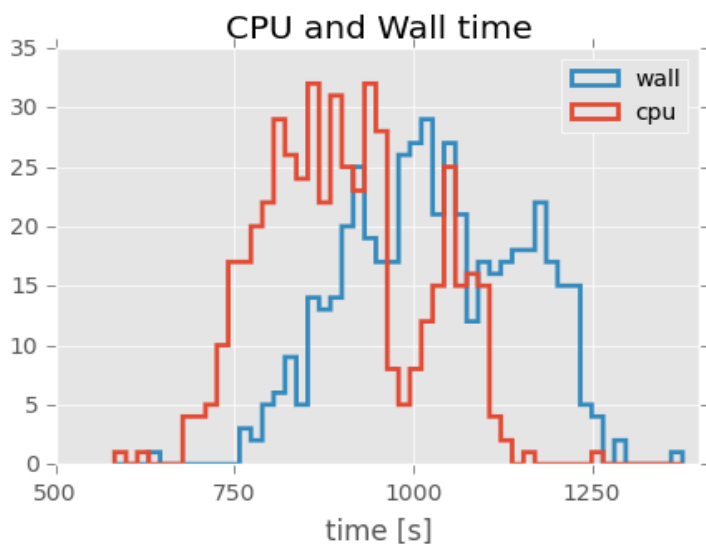
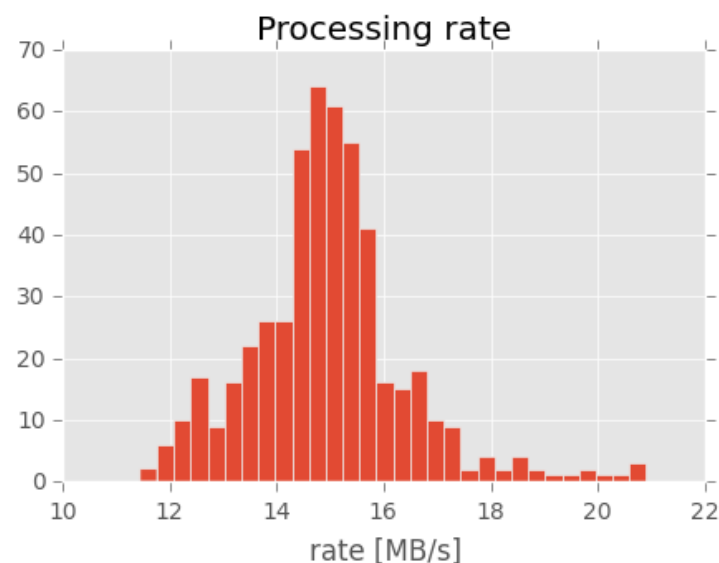
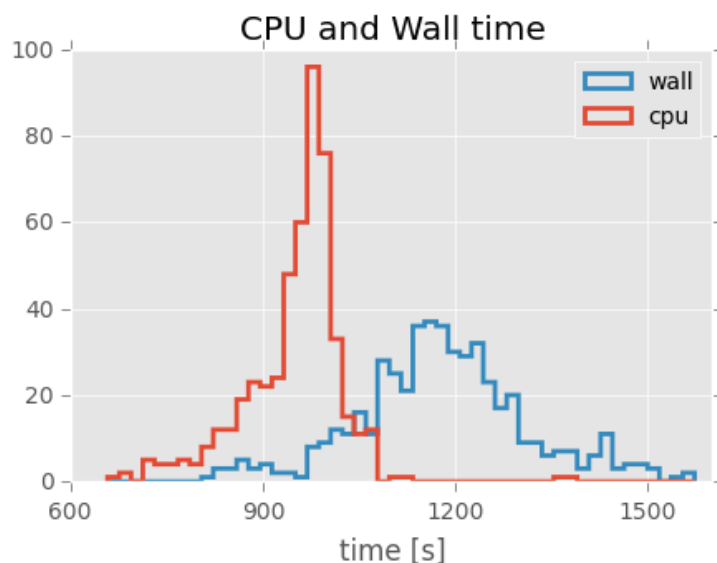
Benchmark output (2/2)

...and some plots (each entry in the plots corresponds to a job in the train)

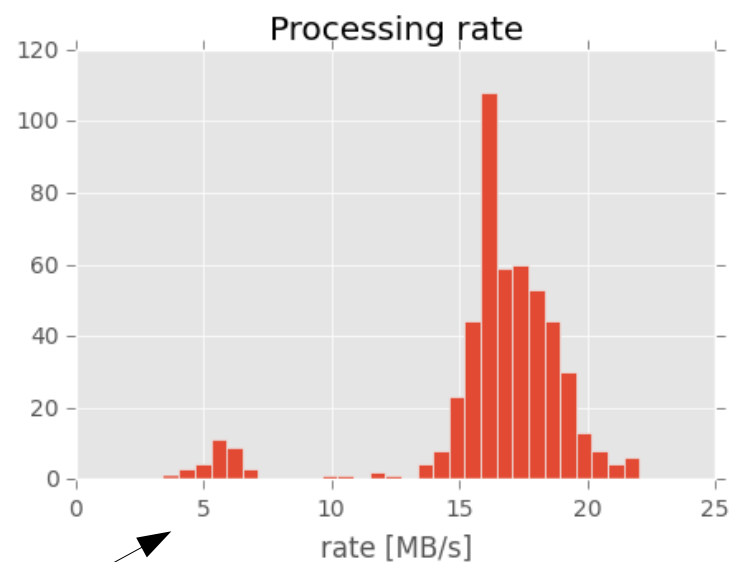
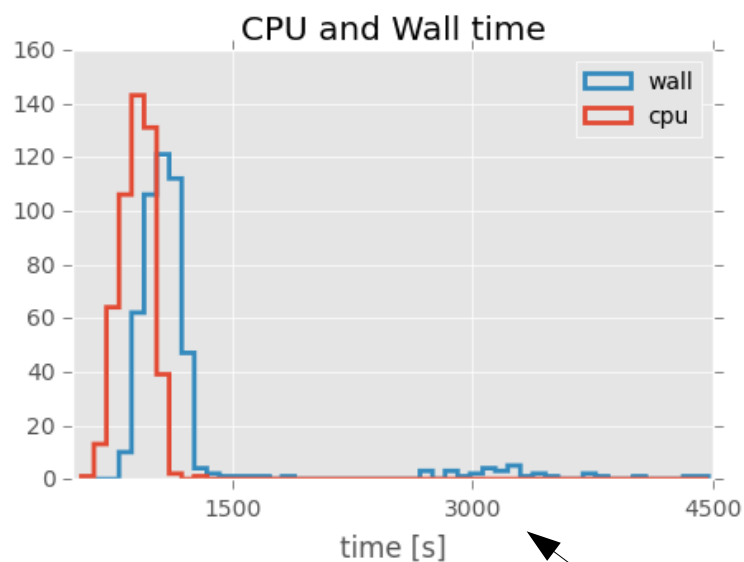
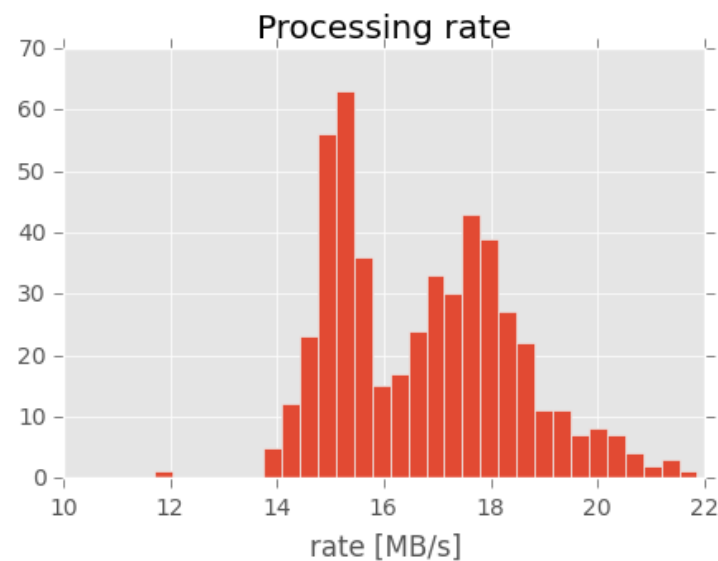
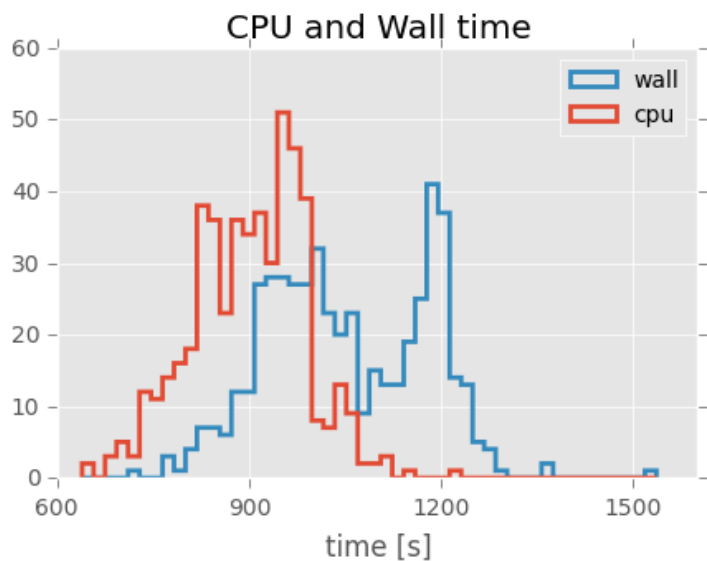


First comparisons (1/2)

In this and the next slide we compare the CPU/Wall time and the corresponding processing rate for jobs in trains with the same conditions submitted at different times (Warning: the scales are always different...)



First comparisons (2/2)



A bunch of slow jobs

Stability of trains with same conditions (1/2)



Same train with 500 jobs submitted at different times along 2 days (October 4th and 5th)

Results are not varying so much

Train conditions are the same, the situation of the farm and of lustre can be different

Measured values (see slide 9 for definition) compared to

$$R1 = \langle \text{rate} \rangle_{\text{jobs}} \cdot n.\text{jobs}$$

$$R2 = \min(\text{rate})_{\text{jobs}} \cdot n.\text{jobs}$$

R2 is in fair agreement with the measurement, meaning that the slowest job determines the train rate



Stability of trains with same conditions (2/2)



Test repeated on a longer period of time (3 weeks: October 19th- Nov. 9th)

A train is submitted every 3h.
The average duration of the train is 30'

→ we are monitoring the conditions for 1/6 of the time

Measured values (see slide 9 for definition) compared to

$$R1 = \langle \text{rate} \rangle_{\text{jobs}} \cdot n.\text{jobs}$$

$$R2 = \min(\text{rate})_{\text{jobs}} \cdot n.\text{jobs}$$



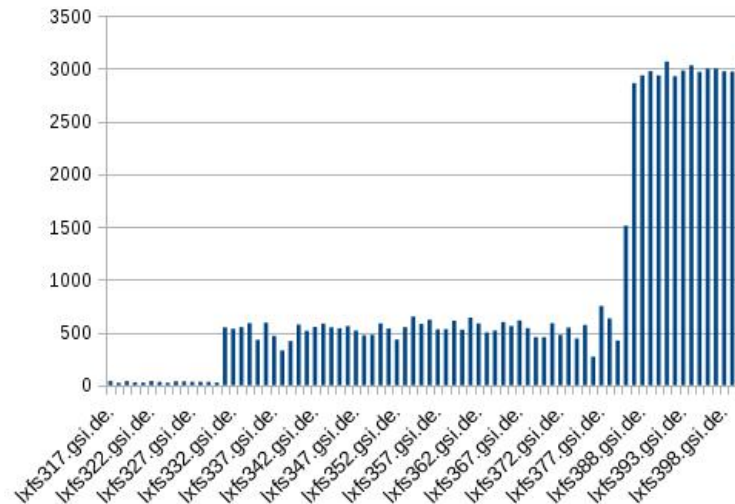
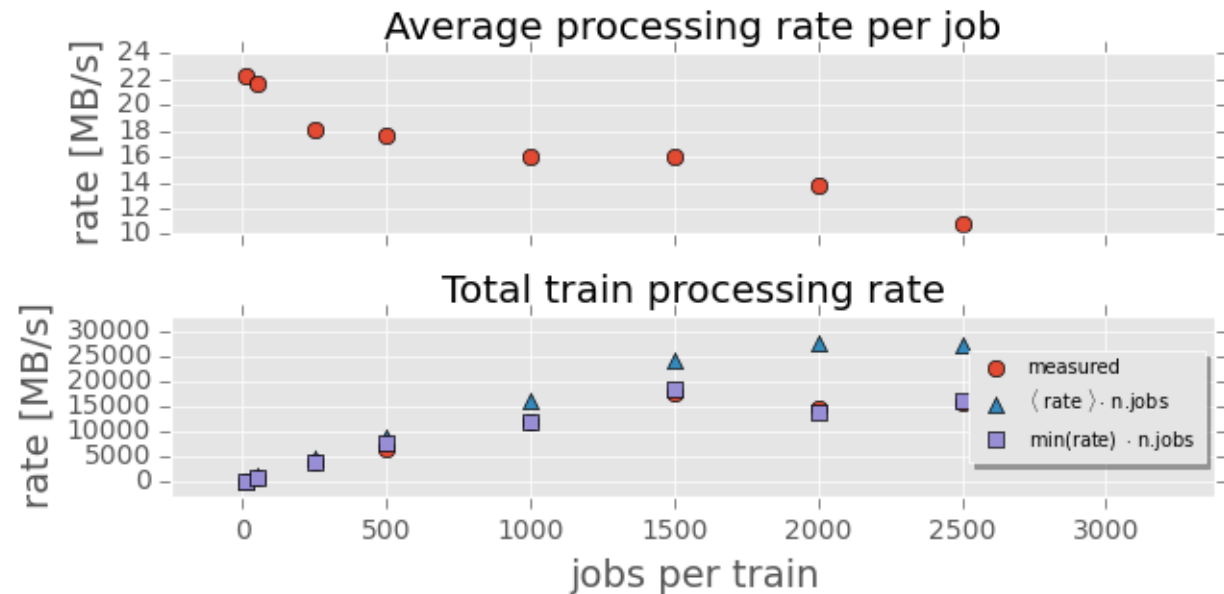
Comparison of train with different number of jobs

Trains with [250-2500] jobs
Higher priority for the jobs, so that
all the jobs start at the same time

For tests with more than 1000 jobs,
results are depending more critically
from the farm usage.
For 1500, 2000, 2500 the best
measurement is shown!

Average rate per job decreases,
aggregate rate is scaling up to
1500 jobs.

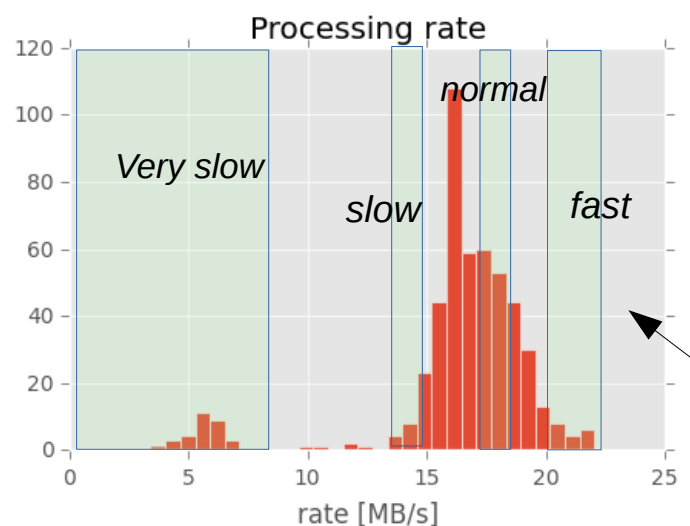
Some of the OSS resulted to be
overloaded during the tests,
due to the fact that the files are
concentrated in few OSS...to be
improved on the AF prototype!



Investigating slow jobs



The benchmark provides also some debugging and investigation capabilities with simple visualizations



Investigate different categories of jobs:

"fast jobs" : processing rate > 20 MB/s

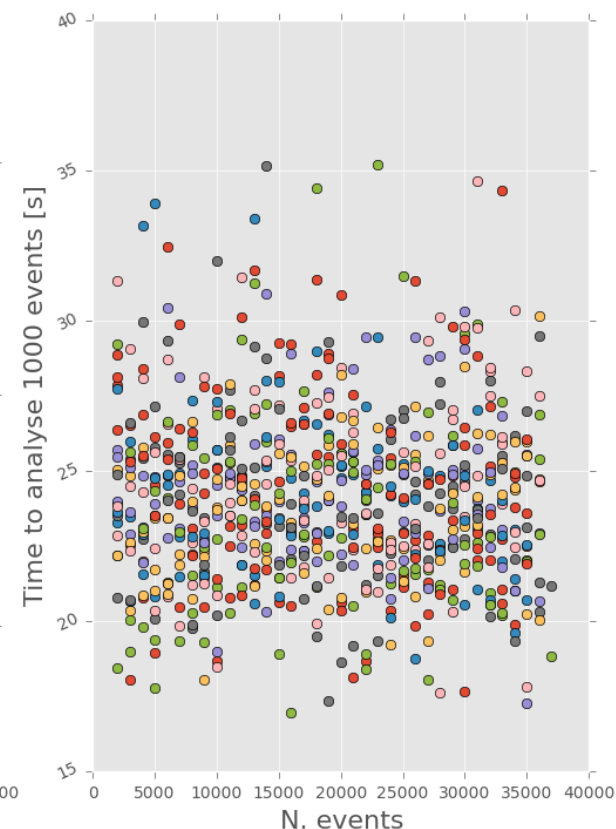
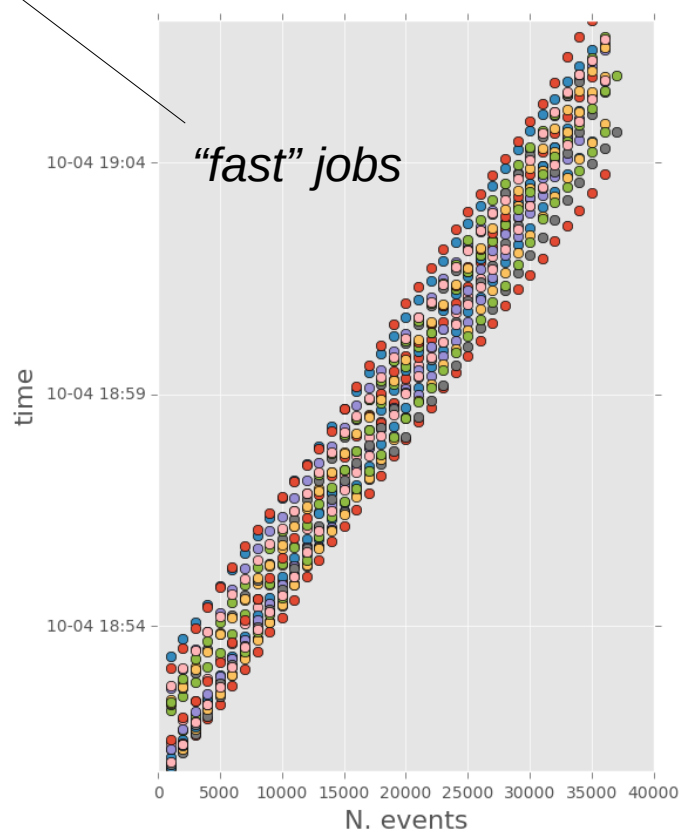
"normal jobs" : rate between 16-17 MB/s

"slow jobs" : rate between 11-15 MB/s

"very slow jobs" rate < 8 MB/s

Each line/color in the plots correspond to a different job

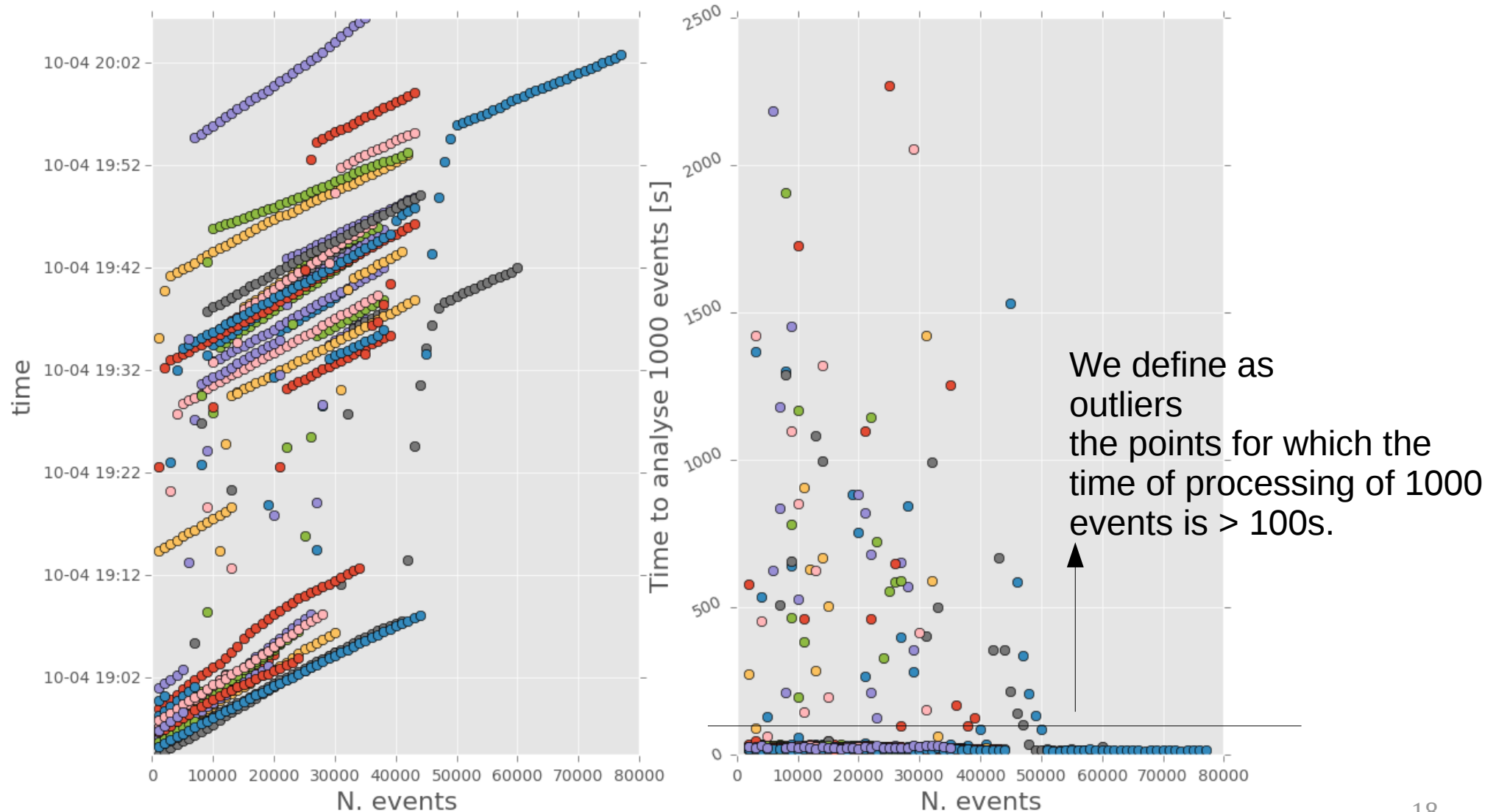
Similar trends for normal and slow jobs



“Very slow” jobs

For the very slow jobs the pattern is more interesting ...

The slope of the first graph changes abruptly for one or few data points and then goes back to normal, indicating that the slowness comes from problems reading specific files



Identifying the reason for the very slow jobs

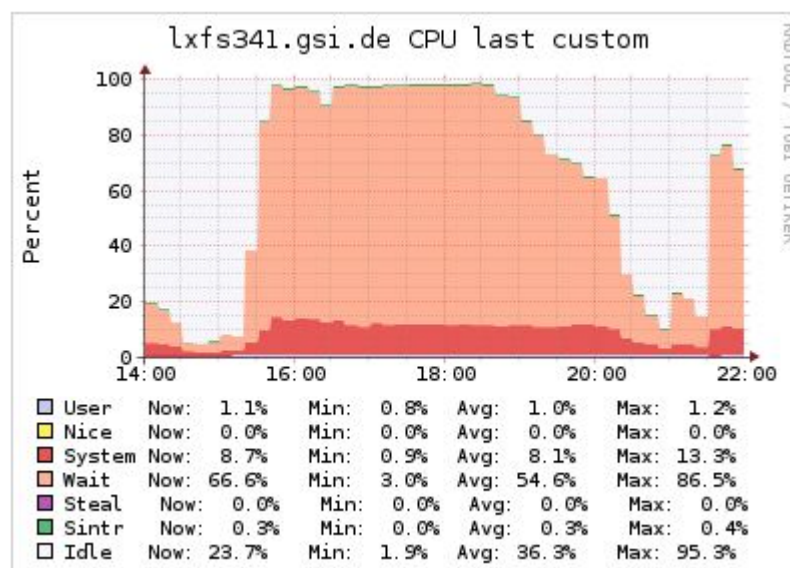
For the **outliers**: we found the **opened AOD files** and retrieved the corresponding **OST** and **OSS**

At the end a **list of suspicious OST and OSS for time windows** is produced: it allows a deeper investigation with the help of the HPC team

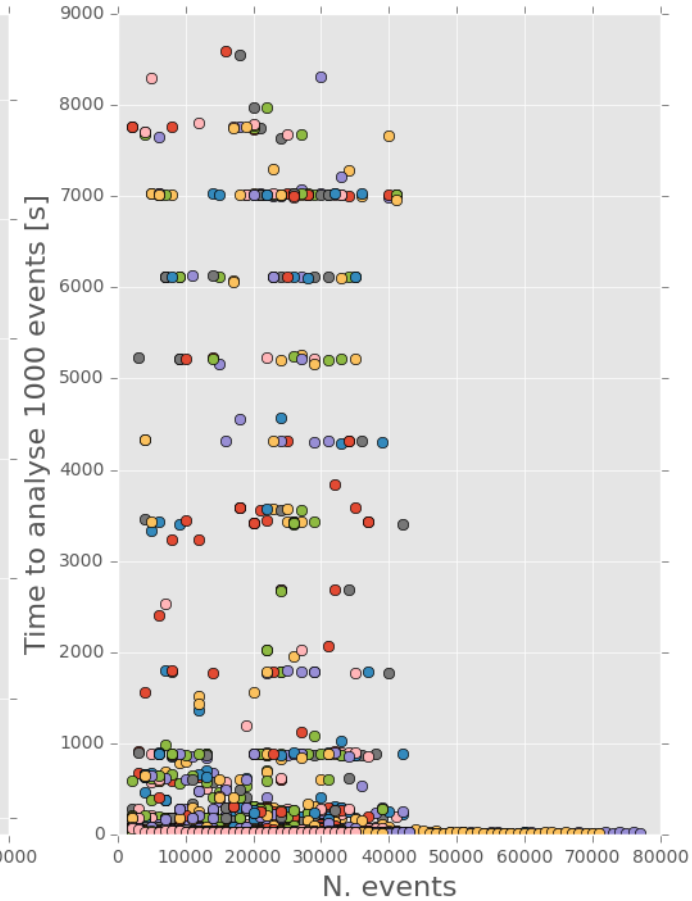
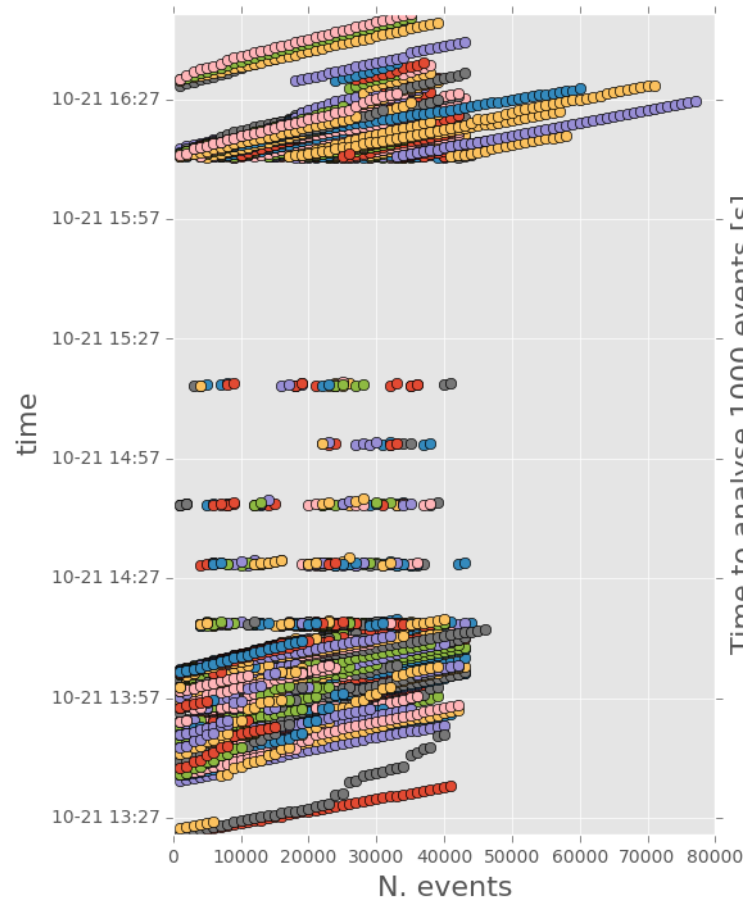
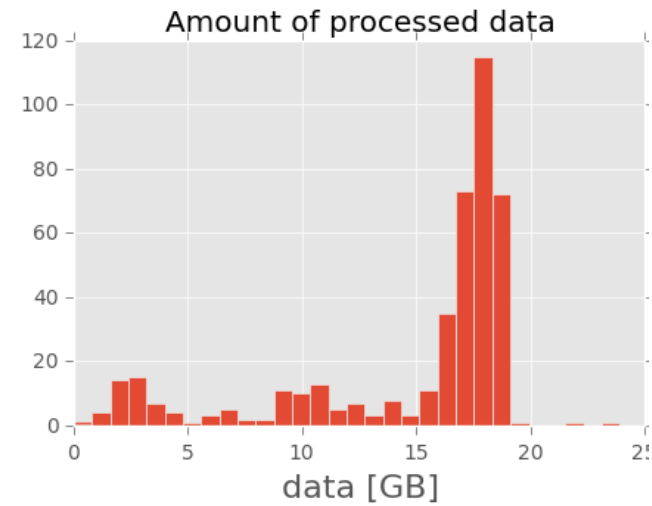
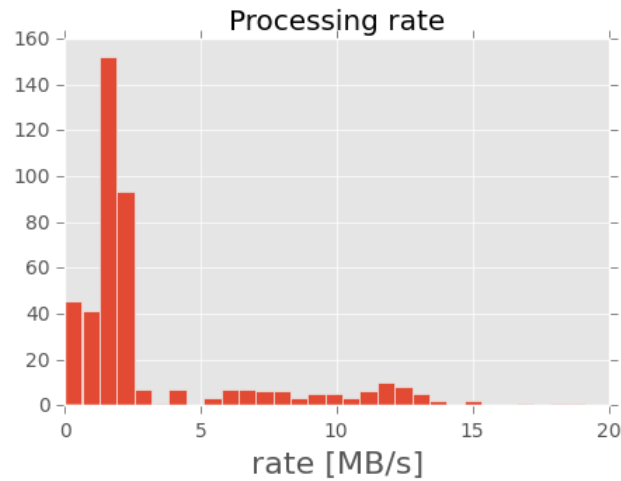
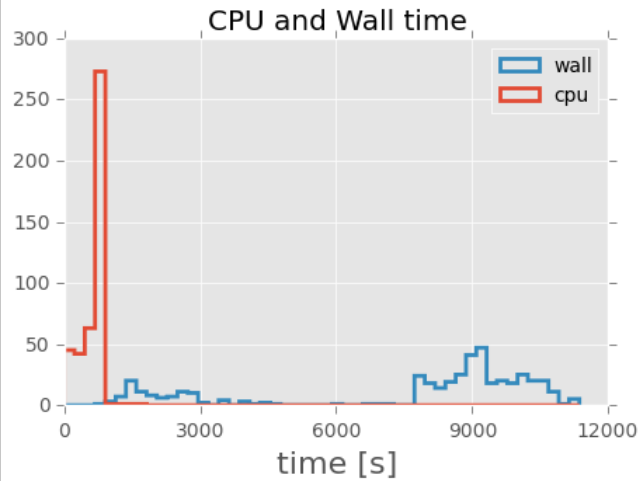
The reason for very slow jobs was due to a overloading of the corresponding OSS (too many requests to the OSS by different users)

The problem can also be identified looking the ganglia monitor :

large I/O rate



Another example of slow jobs



Jobs stop at the same times
-> problems at the MDS

Summary

An analysis facility prototype is being finalized at GSI.

A first baseline for the analysis facility was measured:

Simple copy: 1.2 GB/s

Unzipping: 100 MB/s

Analysis: ~20 MB/s

Max aggregate train throughput using a simple analysis task measured as

18 GB/s for 1500 jobs

[from TDR: 5MB/s for job – 20000 job slots – aggregate throughput of 100GB/s]

Scalability for higher number of jobs to be measured again with more homogeneous spread of the AOD among the OSS.

Benchmark ready to monitor, investigate and compare the performance of the analysis trains in different conditions

Tests to be repeated using dedicated LEGO train on the analysis facility

Backup



ALICE

Backup – GSI cluster



SCHEDULER Slurm 14.03.9

OF NODES 552

OF CPU CORES 13856

CPU MODEL Intel® Xeon® E5-2660 v3, Intel® Xeon® E5-2680 v4

MEMORY 25 TiB DDR4-2133 reg ECC, 44 TiB DDR4-2400 reg ECC

NETWORK Mellanox® FDR 56 Gb/s Infiniband

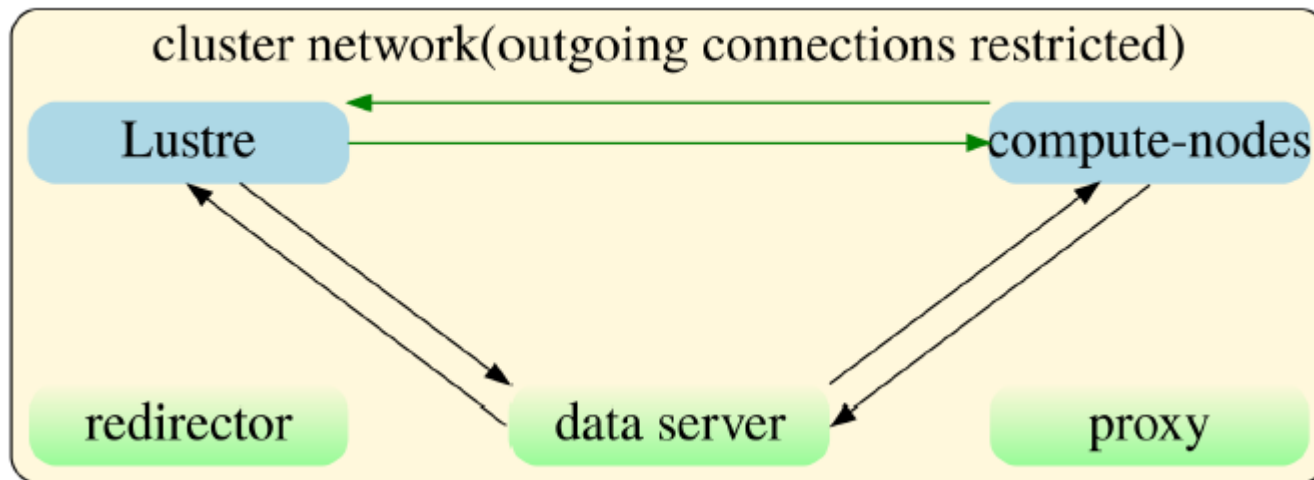
PLATFORM GNU/Linux Debian 8 Jessie

SHARED STORAGE Lustre Nyx (Capacity 15 PB)

546 OST , 7 OST for OSS

XRootD Client Plug-in - XrdOpenLocal:

Clients should open a file directly
from Lustre if at GSI



Available as Client and **Server (Redirector) Plugin.**

Clients will still need a new XRootD Client, though
Needed Client code in XRootD base starting with
version 4.8

(see Bachelor thesis Paul-Niklas Kramp)