Data Analysis strategies Current work with the Alice CAF

Atlas meeting

30 August 2006

René Brun

CERN

Data Analysis: Qui suis-je?

- 1- Conventional batch analysis generating ntuples -> ROOT used a la PAW, ie LEP model or pre-data-taking LHC model.
- 2- GRID-based batch analysis with an interactive front-end to submit the jobs + a post-processor to merge output files: the “standard model”
- 3-Interactive model a la PROOF on a local cluster (current Alice CAF)
- 4-same as 3 but GRID-based

Data Analysis: Ou ca?

- 1-from laptop connect to an “interactive” farm like lxplus
- 2-from laptop login to a specialized and dedicated farm accepting a mixture of batch sessions and interactive sessions (eg normal ROOT)
- 3-run ROOT (or like) on laptop connected with a PROOF-like daemon on a cluster where the real things happen: only final results processed on laptop. The daemon is single-thread
- 4-same as 3 but with a PROOF-like master distributing work to N workers (event parallelism)

Data Access models

- 1-access LAN remote data from worker node.
  - rfio, rootd, xrootd, dcache,etc
- 2-access WAN remote data from worker node
  - xrootd, gfal?
- 3-move process to data
  - with experiment specific software
  - with general software, eg PROOF

Cost effectiveness

- How to define it and measure it?
- Maximum throughput after one week or month?
- The best response time for short, medium jobs or queries?
- Hardware/software uniqueness or a general solution easy to clone somewhere else?
- A system that can be easily upgraded by steps?
- A system easy to use and robust?

Constraints

- Money
- Manpower
- Time to implement, Time to understand the various scenarios.
- Requirement to run the full experiment framework on the analysis facility? A subset?
- Need to access external data bases (the real killer!)?

Decision Process

- Risk to be biased by the simulation/production experts. Data analysis is substantially different.
- Risk to be biased by the IT experts with a strong expertise in managing large batch clusters.
- At the end people vote with their feet, but it might be too late.
- Discussions are important now. More prototyping is needed. Users should be confronted to different scenarios.

Decision Process 2

- Exploit acquired knowledge with batch and GRID-based systems. Understand complementary approaches and evaluate real costs.
- Rely on the IT experts with a strong expertise in managing large batch clusters to provide the best guidance to buy and manage the hardware.
- Discussions are important now. Users should be confronted to different scenarios and gradually converge towards the most economical hardware infrastructure with the best return value, ie users satisfaction.

Typical analysis tasks

- Navigate in one or more files, selecting events, filling histograms, selection lists, new ntuples.
- Same but on a larger and larger set of files; could be several thousand files.
- The analysis algorithm can range from a simple query (tree.Draw), a simple algorithm in C++ or equivalent, to a complex analysis task with many thousand lines of code.
- When the analysis task grows, more and more selective algorithms are applied to large collections. The selectivity can be high, ie only a few events per file are selected.

Typical analysis tasks 2

- The process is repeated many many times. The user correct bugs, refines his selections.
- Feedback histograms are an important ingredient while the query/job is running, combined with CRTL/C.
- Users connect/disconnect frequently to inquire the status of the running queries.

Important parameters

- Impact of multi-core cpus.
  - Today dual-core machines
  - This fall quad-core
  - 16-core in 2012?
- How much memory per core?
- Necessary network bandwidth
- High bandwidth, high latency networks
  - Must minimize number of messages and increase the message size.
- Disk access granularity:
  - Minimize number of seeks, make longer transactions.

Multi-Core cpus

- Multi-threading or/and parallel processes
- Gain real-time by doing as much as possible tasks in parallel, eg
- Use multi-threading for:
  - Asynchronous file read-ahead
  - Unzipping
- Use parallel processes for event level parallelism
- BUT more and more difficult to keep a good balance between CPU and I/O.

Network considerations

- We will see higher and higher bandwidths for LANs and WANs (likely terabits/s)
- But latency will remain constant
- Current TCP/IP has a default window size of 64K that introduces a serious performance penalty if the application can manage large block sizes (see later). Hoping for some progress in the near future with adaptative window sizes.

Disk considerations

- Reading small blocks might be inefficient.
- Seeking randomly on disk is bad. Better read sequentially if you can (see later)
- Multiple concurrent users reading from the same disk generate a lot of seeks (extremely bad)
- These considerations are less important in a batch environment, but absolutely vital for interactive applications.

Case of ROOT Trees

- Trees will be probably the most frequent type of collections. Trees are divided into branches that have their own buffers. Typical branch buffers are in the range of 1kb to a few kb (after compression).
- This structure is essential for fast data analysis when processing only a subset of the branches and/or a subset of the events.
- But it had a big performance penalty up to ROOT version 5.12 when accessing remote files.

Improvements in ROOT I/O
Impact on data analysis

- In version 5.12, a new cache mechanism has been introduced improving considerably the I/O performance in LAN and WANs.
- rootd and xrootd are already able to take advantage of this improvement.
- This improvement opens new possibilities, in particular efficient access to remote files on fast networks (even high latency WAN networks)

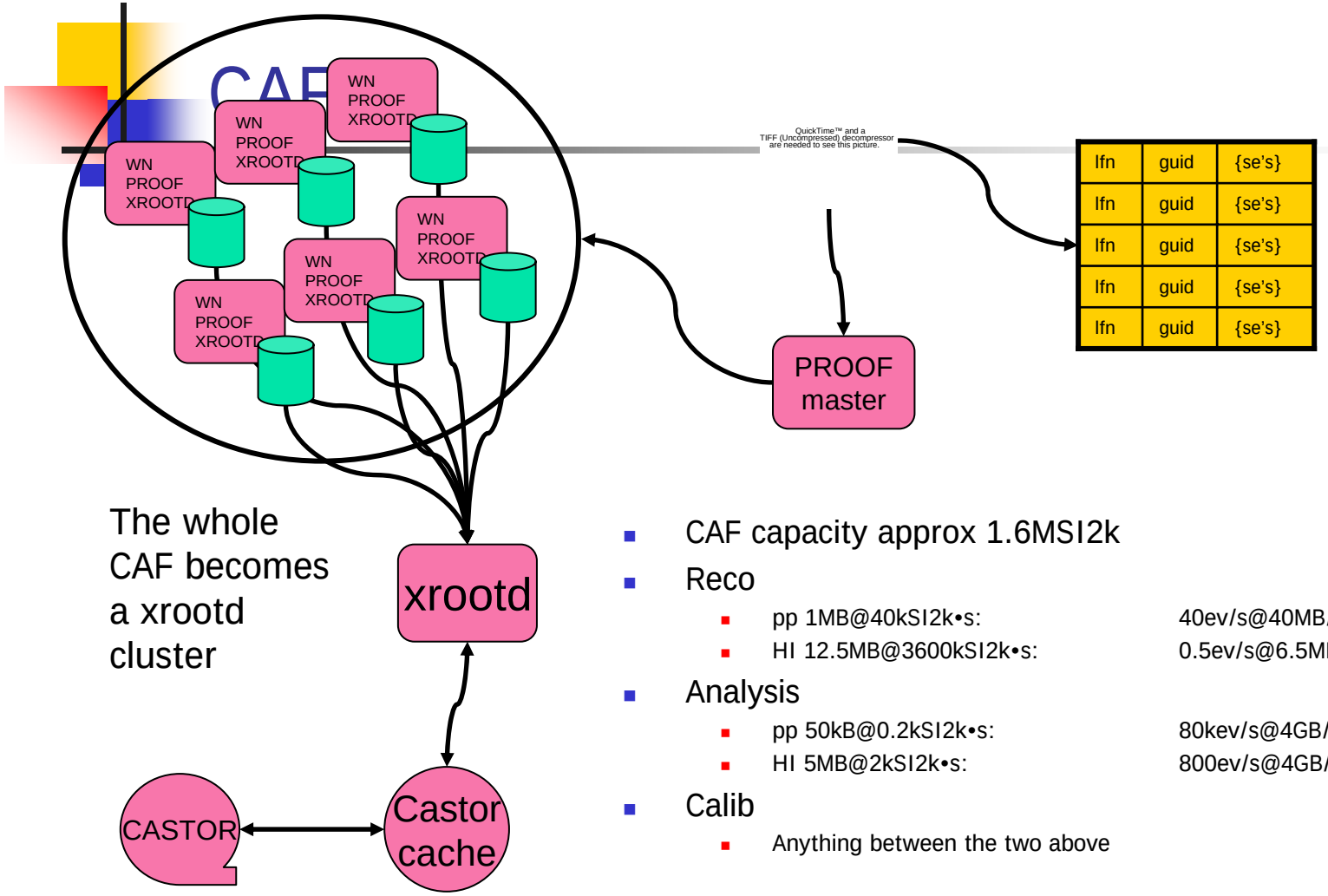
Improvements in ROOT I/O
Impact on data analysis 2

- The file is on a CERN machine connected to the CERN LAN at at 100MB/s.
- The client **A** is on the same machine as the file (local read)
- The client **B** is on a CERN LAN connected at 100 Mbits/s with a network latency of 0.3 milliseconds (P IV 3 Ghz).
- The client **C** is on a CERN Wireless network connected at 10 Mbits/s with a network latency of 2 milliseconds (Mac Intel Coreduo 2Ghz).
- The client **D** is in Orsay (LAN 100 Mbits/s) connected to CERN via a WAN with a bandwith of 1 Gbits/s and a network latency of 11 milliseconds (P IV 3 Ghz).
- The client **E** is in Amsterdam (LAN 100 Mbits/s) connected to CERN via a WAN with a bandwith of 10 Gbits/s and a network latency of 22 milliseconds (AMD64 280).
- The client **F** is connected via ADSL with a bandwith of 8Mbits/s and a latency of 70 milliseconds (Mac Intel Coreduo 2Ghz).
- The times reported in the table are realtime seconds

client	cachesize=0	cachesize=64KB	cachesize=10MB
A	3.4	3.4	3.4
B	22.0	6.0	4.0
C	11.6	5.6	4.9
D	124.7	12.3	9.0
E	230.9	11.7	8.4
F	743.7	48.3	28.0

One query to a 280 MB Tree I/O = 4.6 MB

The Alice CAF prototype

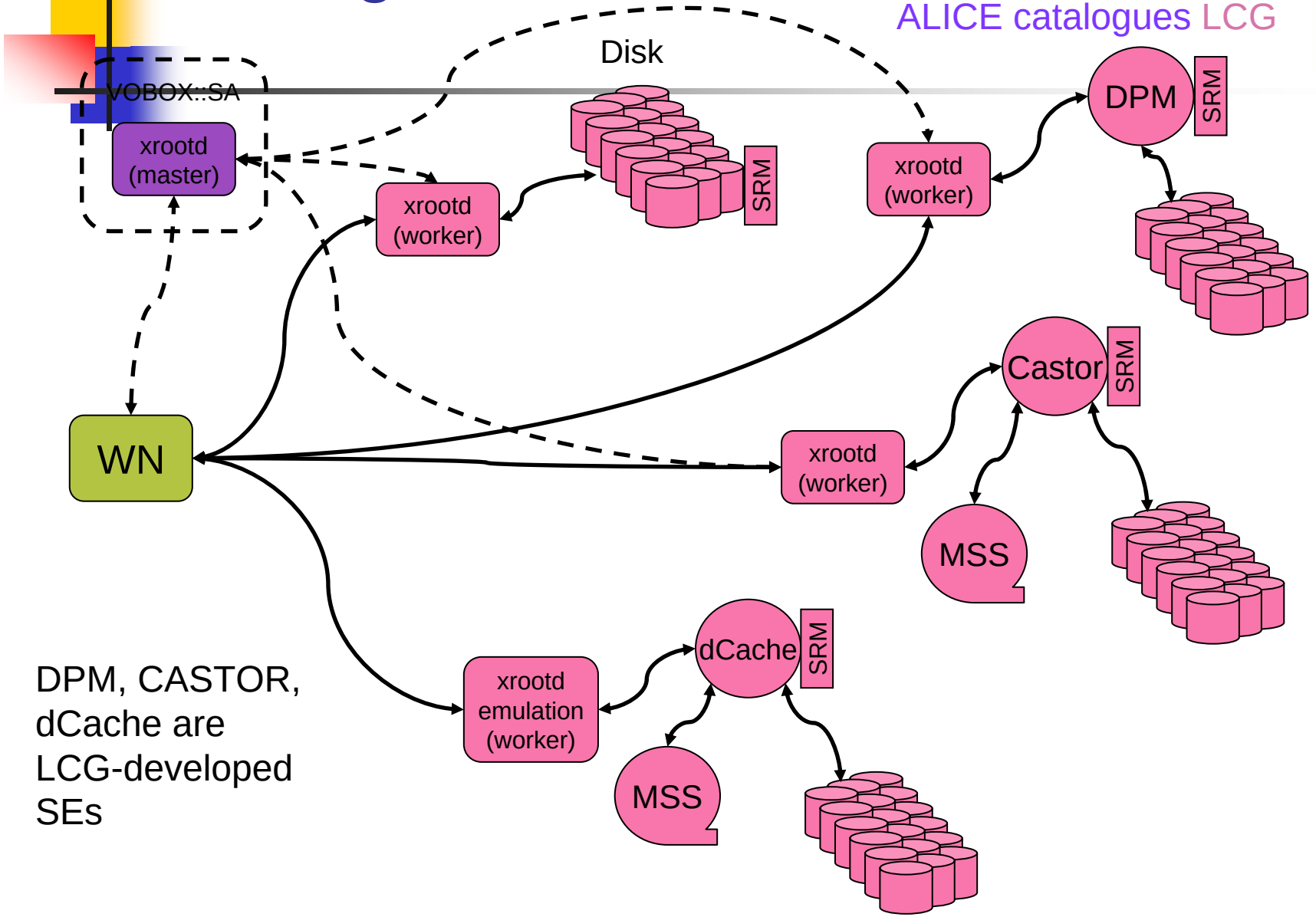


- CAF capacity approx 1.6MSI2k
- Reco
  - pp 1MB@40kSI2k•s: 40ev/s@40MB/s
  - HI 12.5MB@3600kSI2k•s: 0.5ev/s@6.5MB/s
- Analysis
  - pp 50kB@0.2kSI2k•s: 80kev/s@4GB/s
  - HI 5MB@2kSI2k•s: 800ev/s@4GB/s
- Calib
  - Anything between the two above

Storage element

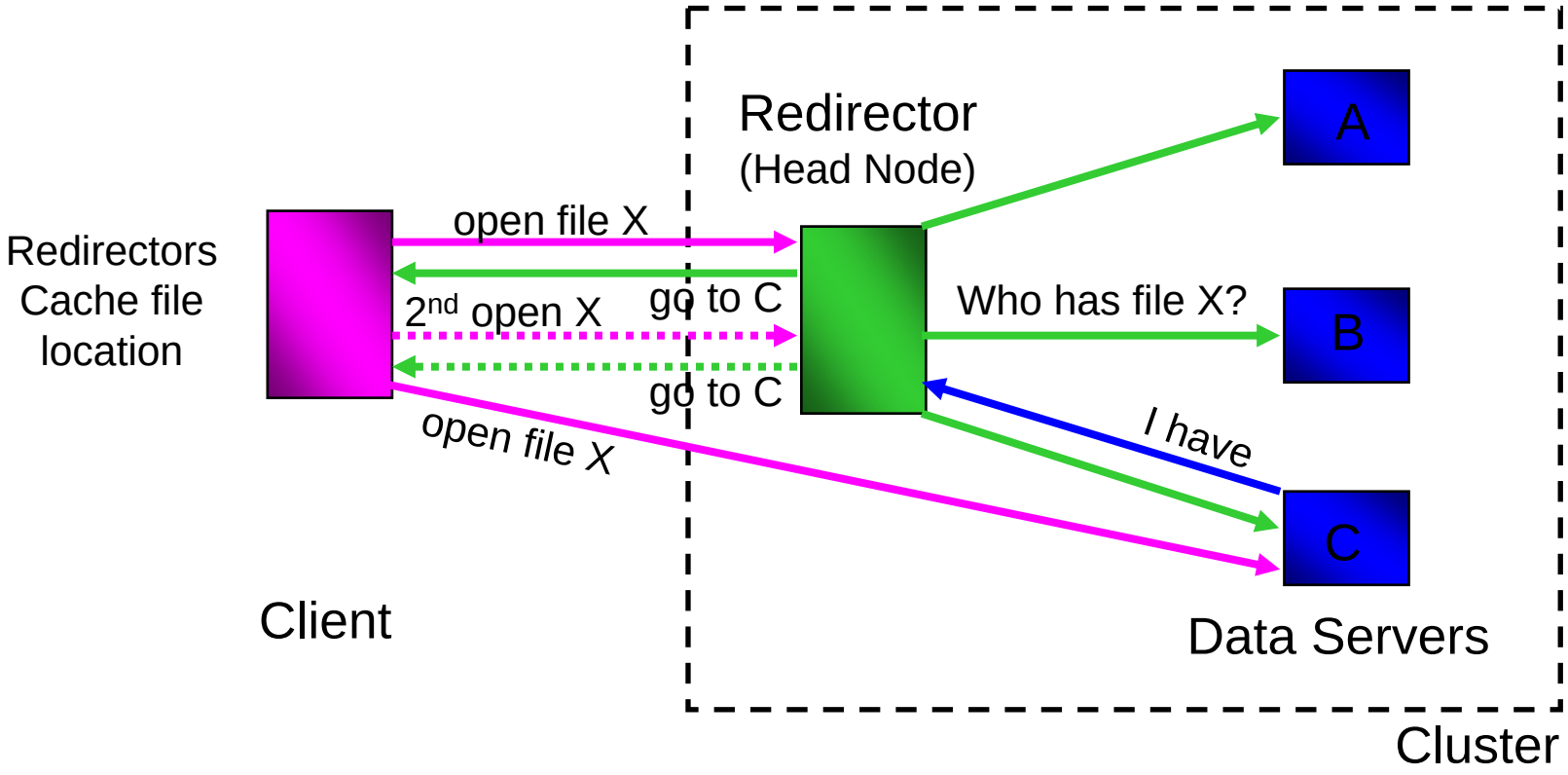
VO-Box User Job

ALICE catalogues LCG



DPM, CASTOR, dCache are LCG-developed SEs

xrootd architecture



Client sees all servers as xrootd data servers

Expected usage by Alice

- A limited number of physicists access the CAF for fast processing of the data
  - 2-3 per detector -> 50-100 users
- Emphasis on response time rather than throughput
  - However we want to check that the system is not underused!
- Files will come from the T0 buffer and will be distributed onto local node disks by xrootd
- Analysis and calib files will be written on the GRID SE's as needed

Goals of the current prototype

- Does the system respond to our requirements?
- What is the advantage over giving each user N_{mach}/N_{user} machines?
- What are
  - The performance features of the setup?
  - The bottlenecks?
  - The hardware requirements?
  - The needed developments?

Answers as of today

- What is the advantage over giving each user N_{mach}/N_{user} machines?
  - Faster response time
  - Higher average system utilisation
  - Less data movement

Do MORE with LESS

- What are
  - The performance features of the setup?
    - The parameter space is very large and we have only scratched the surface, see next plots
  - The bottlenecks?
    - Software stability
    - Data distribution
    - No hardware bottleneck as long as each CPU can be fed 15MB/s

Answers as of today

- What are
  - The hardware requirements?
    - We have seen no clear hardware bottleneck
    - Standard PC's are OK provided there is enough local storage (0.5TB/2CPU's)
  - The needed developments?
    - Improve stability
    - Data access and distribution
    - xrootd interface to Castor stager

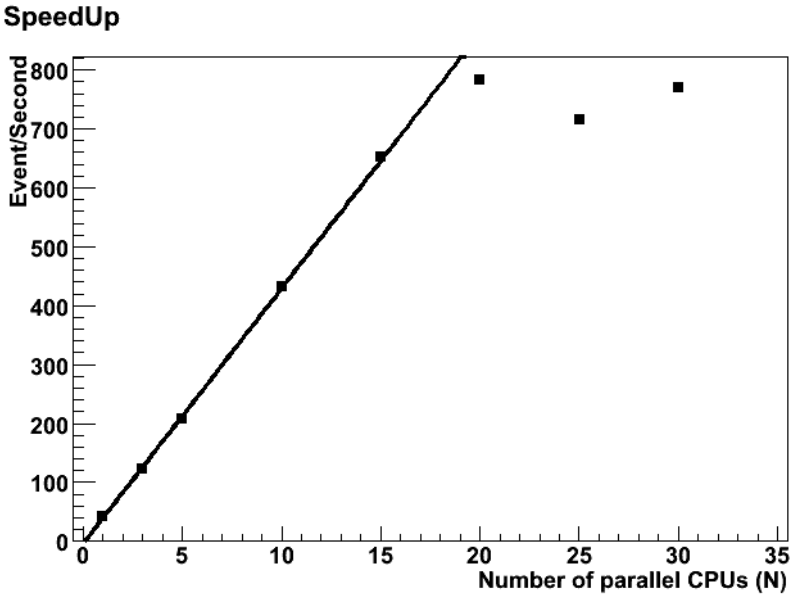
Test setup

- Since May evaluation of CAF test setup
  - 40 machines, 2 CPUs each, 200 GB disk
- Tests performed
  - Usability tests
  - Simple speedup plot
  - Evaluation of different query types
  - Evaluation of the system when running a combination of query types
- Care is taken that the files do not fit in the system cache using long chains of files

File distribution during the test

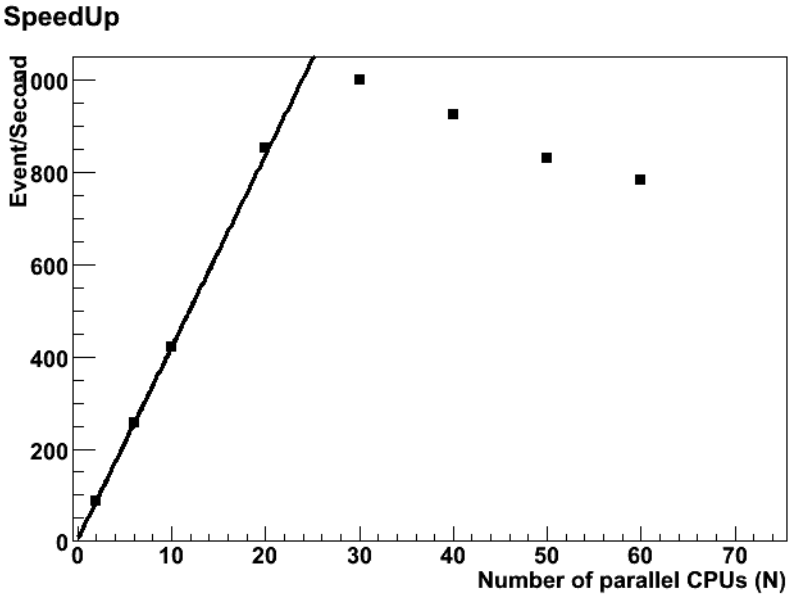
- The files have been distributed before the test using xrootd functionality
  - All were copied to the redirector machine that distributed them over the cluster
  - xrootd tries to distribute the files evenly, but some nodes host more files than others (difference up to 50%)
  - We did not correct because this is a realistic scenario for analysis
  - For each query we selected files at random between those available
- PROOF will try to process local files if any, otherwise it will process the non-local ones
- 22000 zip archive files 50MB/file, 1.1 TByte
  - 100 pp events/file
- 1000 zip archive files 500 MB/file 0.5 TByte
  - 2000 pp events/file

Simple speedup



- One query to an empty CAF
- Data is evenly distributed
- Each query processes at least 10 minutes to make influence of overhead small
- Different data files are given to each query to prevent caching

- Probably data contention kicks in at some point and reduces parallelism, but this needs more study
- More CPU bound queries would give a better speedup – TBD for AA



Query Types

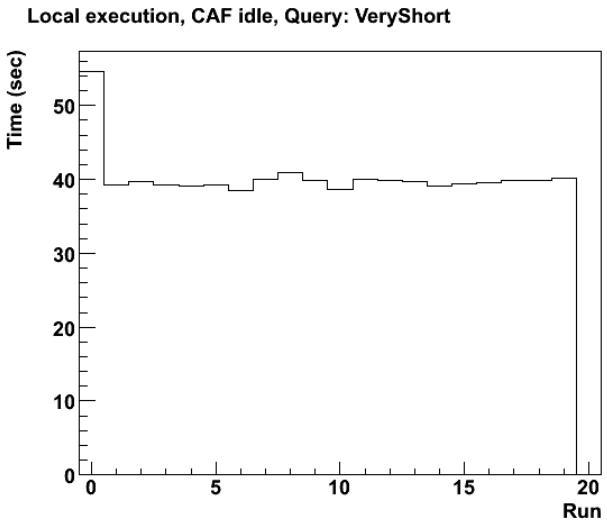
Name	# files	# evts	processed data	avg. time*	I/O rate* (MB/s)	Submission Interval
VeryShort	20	2K	0.4 GB	9 ± 1 s	44.4	30 ± 15 s
Short	20	40K	8 GB	150 ± 10 s	53.3	120 ± 30 s
Medium	150	300K	60 GB	1,380 ± 60 s	43.5	300 ± 120 s
Long	500	1M	200 GB	4,500 ± 200 s	44.4	600 ± 120 s

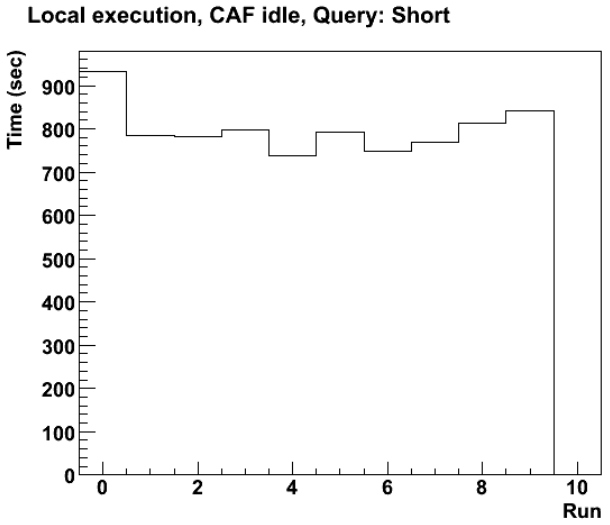
*run in PROOF, 10 users, 10 PROOFservs each

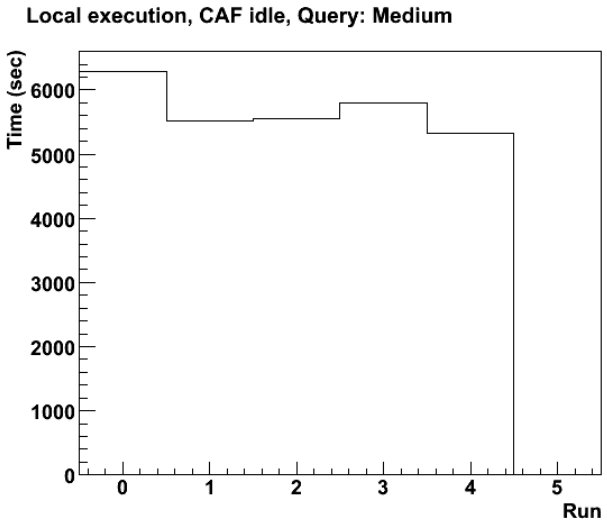
Query type cocktail

- 4 different query types
  - 20% very short queries
  - 40% short queries
  - 20% medium queries
  - 20% long queries
- User mix
  - 33 nodes
  - 10 users, 10 or 30 processes/u per user
  - 5 users 20 processes/u
  - 15 users 7 processes/u
  - Max average speedup = 6.6

Time evolution – scalar queries

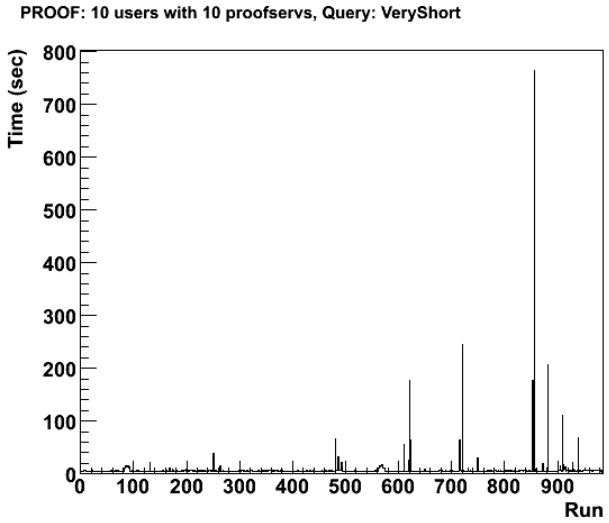


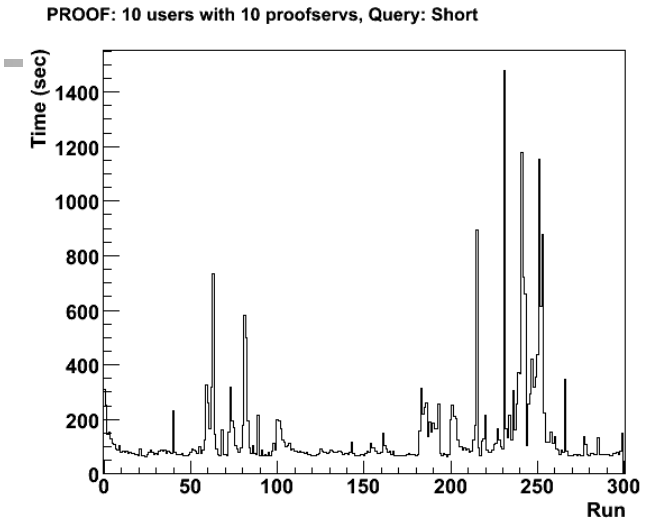




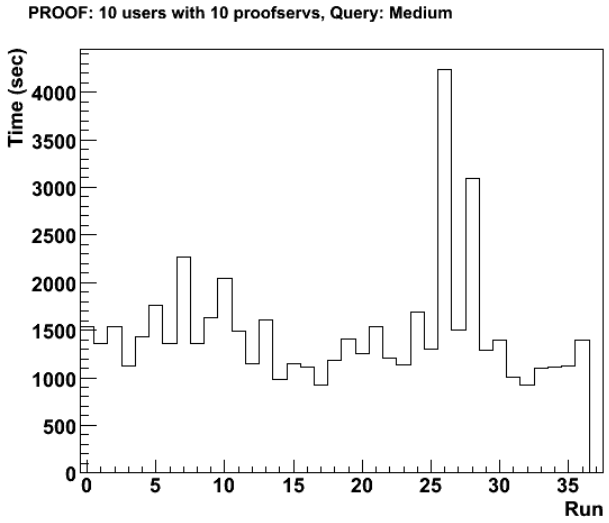
- Even distribution of files => 1/33 are local
- Second query faster because the files are cached in the memory of the machines serving the files

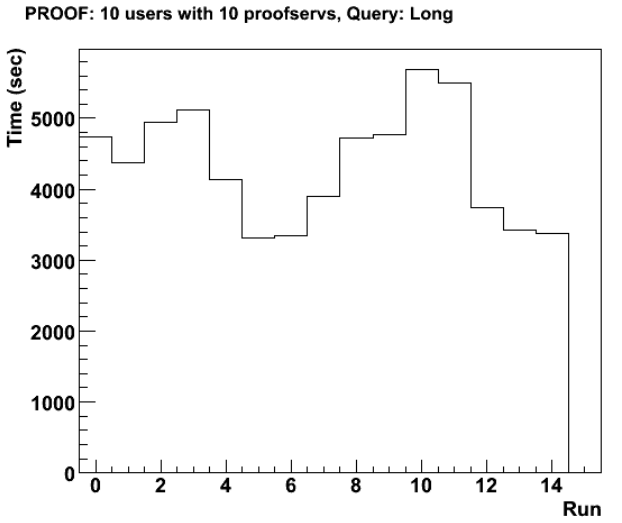
Time evolution – PROOF queries





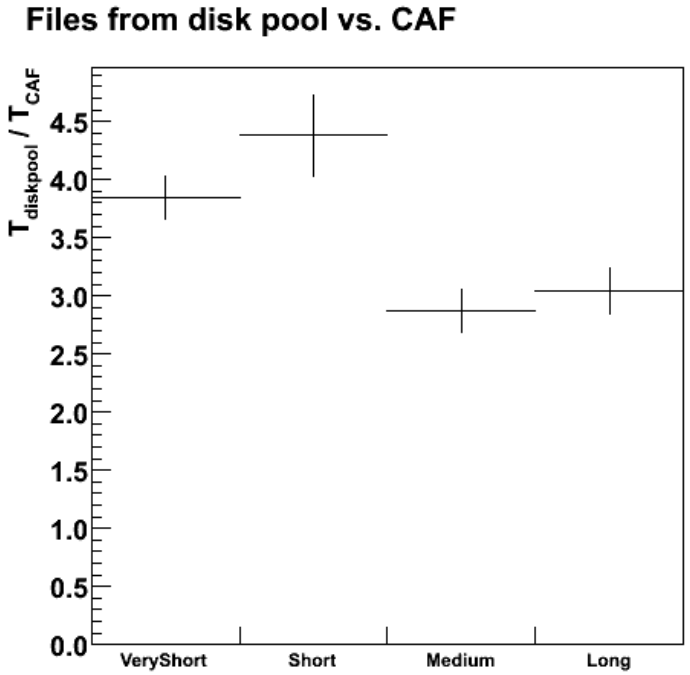
Outliers to be analysed



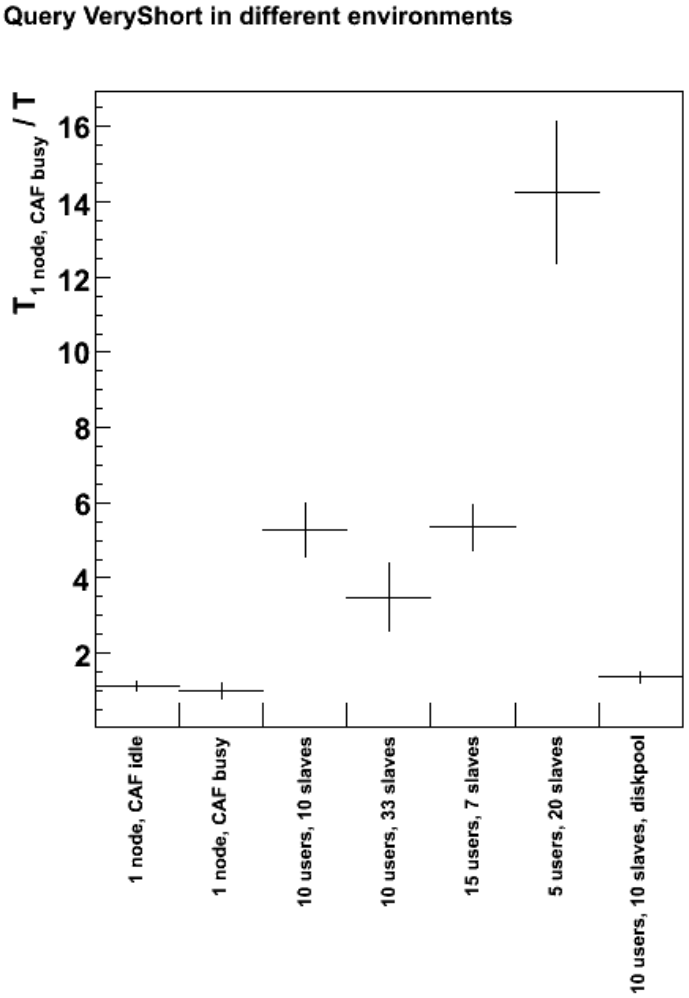


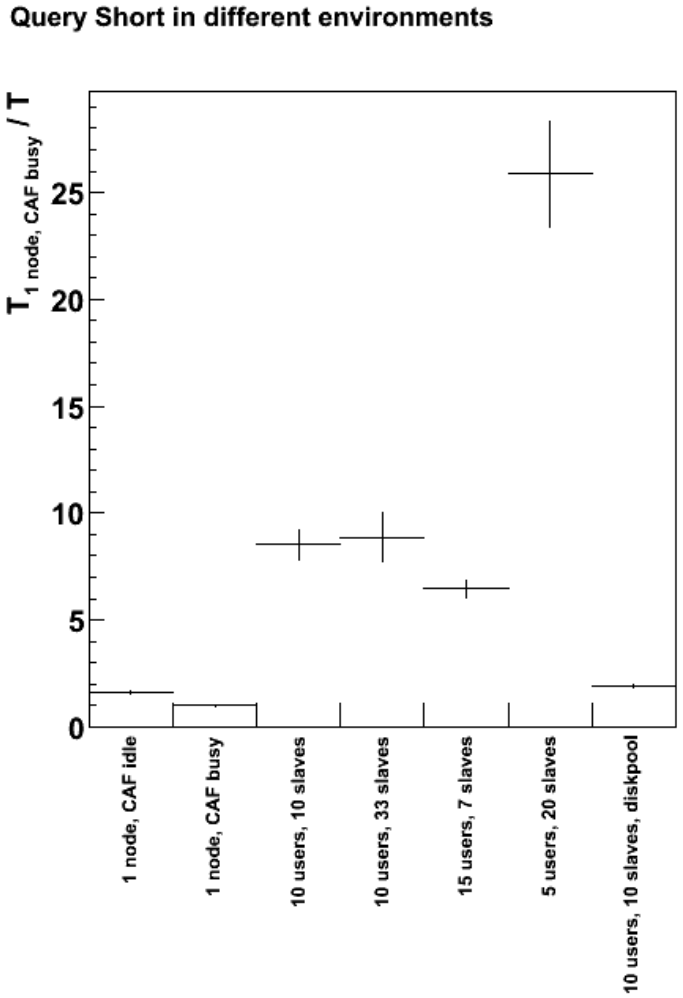
Disk pool vs distributed files

- The theoretical bandwidth probably the 1GB (125MB/s) switch between CAF farm and the disk pool machines
- All other conditions are the same, just the files not read from the CAF itself.

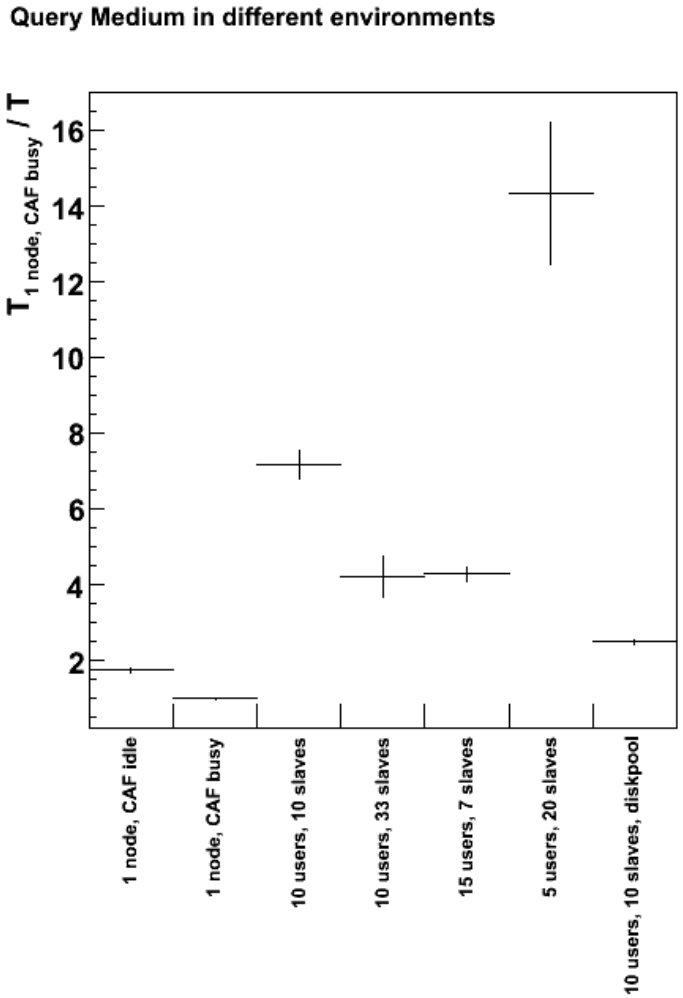


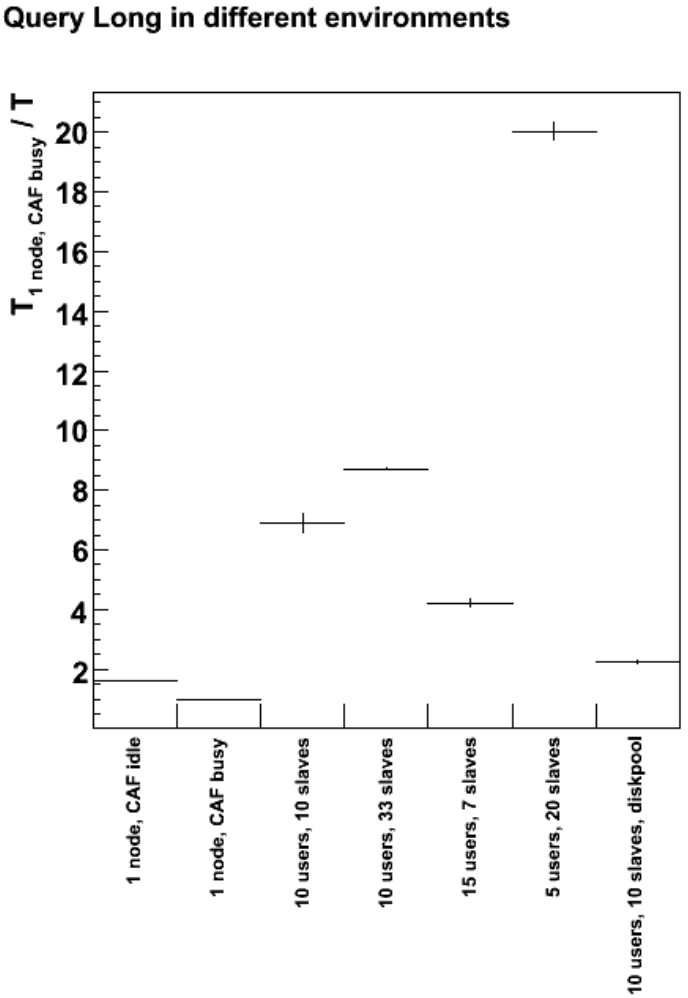
Relative speedup





Relative speedup



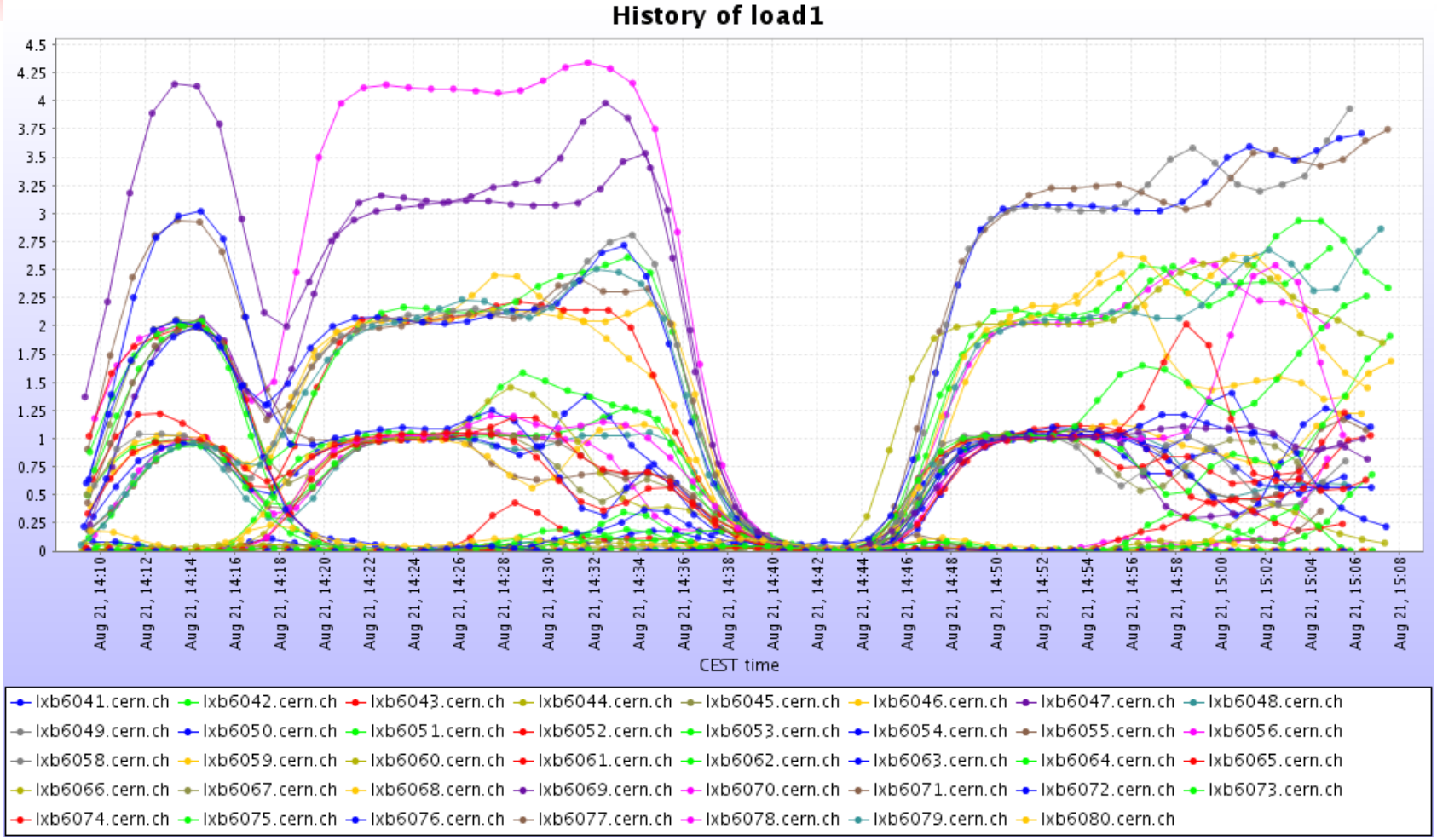


Monitoring

Evaluation of the system behavior

- MonALISA monitoring
  - each host reports to MonALISA
  - each PROOFserv reports to MonALISA
- http://pcalimonitor.cern.ch:8889/
  - Click on CAF monitoring

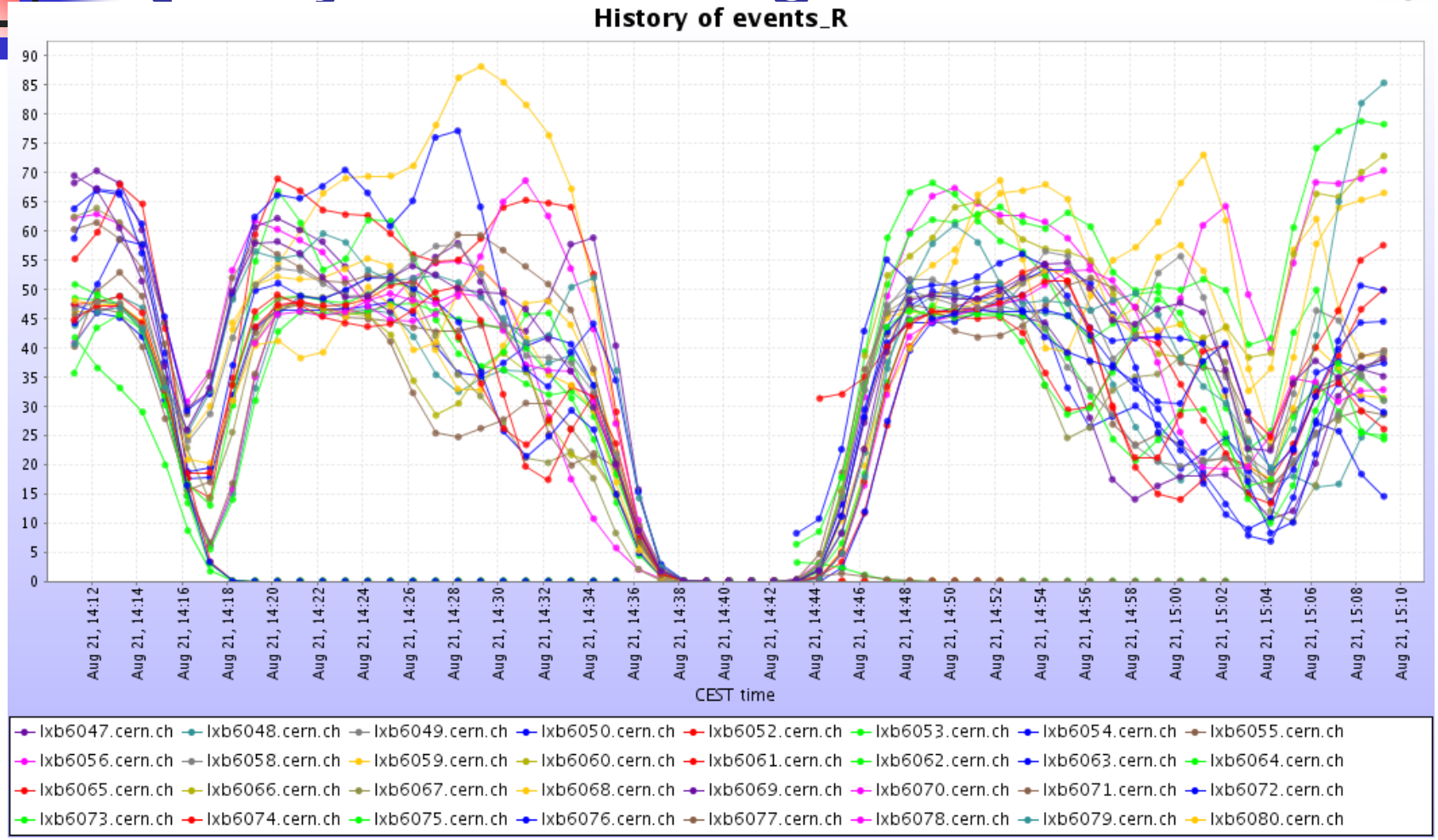
Host monitoring



the same for CPU, memory, swap, network, ...

selectable per query type and per host

Query monitoring



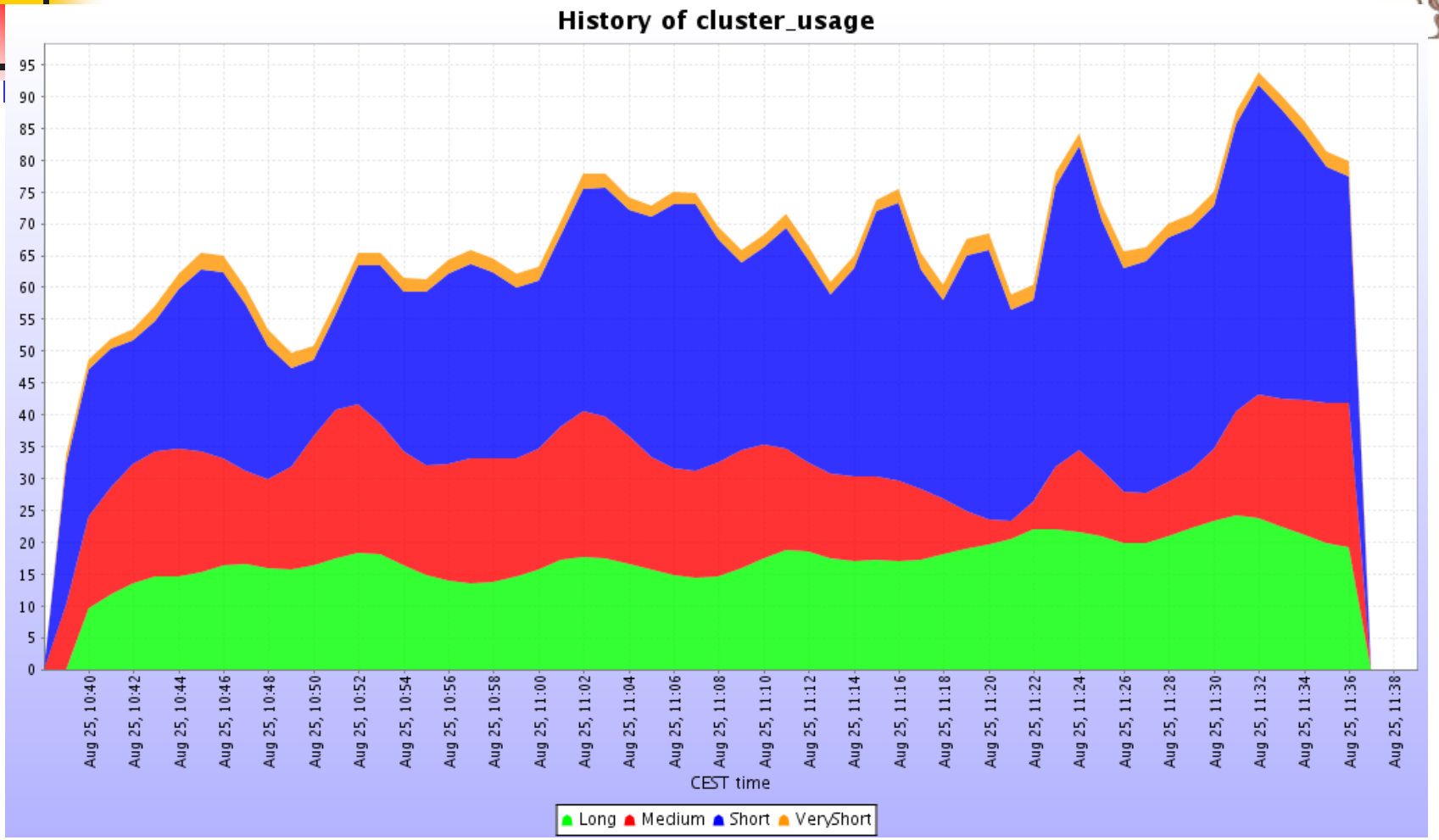
the same for: CPU usage, cluster usage,
Memory, Event rate, Local/External MB/s / files/s

Network traffic

Traffic between the cluster machines (MB/sec) (last 0.5h average)																							
Machine	**6047**	**6048**	**6049**	**6050**	**6052**	**6053**	**6054**	**6055**	**6056**	**6057**	**6058**	**6059**	**6060**	**6061**	**6062**	**6063**	**6064**	**6065**	**6066**	**6067**	**6068**	**6069**	**607**
1. 6047	0	-	-	-	-	-	2.927	2.018	-	-	1.094	-	-	-	1.908	4.112	-	-	0.974	0.614	0	0	
2. 6048	-	9.406	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
3. 6049	-	-	8.678	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
4. 6050	-	-	-	6.692	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
5. 6052	-	-	-	-	3.913	-	1.454	-	-	-	-	3.084	-	0.317	0	0	-	0	-	-	0.985	4.447	
6. 6053	-	-	-	-	-	6.603	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
7. 6054	0	-	-	1.363	-	-	6.195	-	-	-	0	-	-	-	0	-	-	-	-	0	-	-	1.5
8. 6055	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
9. 6056	-	-	-	-	-	-	-	-	4.962	-	2.442	0.525	-	-	-	-	-	-	-	-	-	-	
10. 6057	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
11. 6058	1.164	-	-	-	0	-	-	-	2.531	-	0	0	-	-	-	-	1.103	-	0	-	-	-	
12. 6059	3.755	-	0.622	-	-	-	-	-	-	-	-	11.76	1.955	0	0.677	1.848	0	-	-	-	2.812	-	0.7
13. 6060	-	-	-	-	-	-	-	-	2.068	-	-	-	11.59	-	-	1.06	-	-	-	-	-	-	2.0
14. 6061	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
15. 6062	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
16. 6063	-	-	-	-	1.655	0.27	2.416	-	-	-	-	-	-	0	-	6.38	-	0	-	0	-	-	
17. 6064	-	-	-	-	-	1.123	-	2.822	-	-	-	-	1.621	-	0	-	3.117	-	0	0	-	-	0.5
18. 6065	0	-	-	-	3.52	3.165	-	0	-	-	0	-	-	-	-	-	3.034	0	1.579	-	0	-	
19. 6066	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	

diagonal elements: local traffic (TFile), other elements network traffic: (TXNetFile)

Distribution between queries



Status

- We are now running many more tests to understand existing bottlenecks.
- We intend to improve our query cocktail to be as close as possible of a real situation.
- More and more users will be put in the system to improve the feedback.
- The software used in AliceCAF is not specific to Alice (ROOT + PROOF)