



FIO

Fabric Infrastructure
and Operations

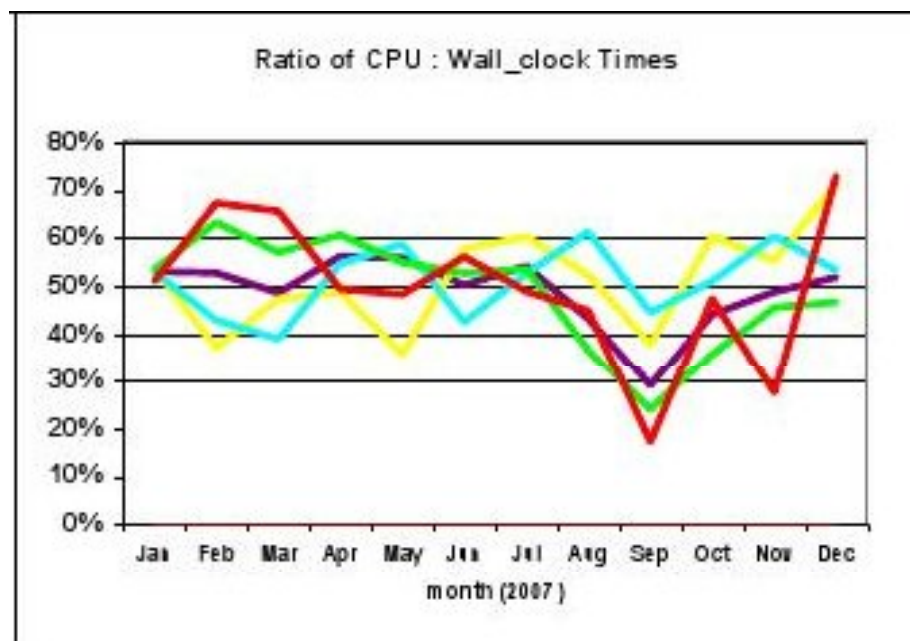
CERN IT
Department


CPU Efficiency

Ulrich Schwickerath
WLCG GDB, 5th March 2008



Job efficiency (CPU/Wall time) for CERN in 2008



ALICE  CMS 
ATLAS  LHCb 

- Error trapping in SEAL
 - Hitting eg. LHCb jobs which got stuck in the system

--> Fixed

- Job Overhead
 - Middleware overhead
 - Measure the overhead for a “hello world” job; calculate minimum cpu time for 85% efficiency
 - Is 30,000 8 minute jobs sensible? This also causes heavy load on the batch system.
 - Sandbox timeouts were 5h15. Being improved at CERN: maximum wait is now just...
... 1h15 minutes!!!

- Job Overhead
 - Data fetch/upload overhead
 - Local SE \leftrightarrow local disk
 - Job recall from tape (shouldn't happen!)
 - Optimum policy for reading from SE or from local disk
 - » Will depend on fraction of data read
 - » Need instrumentation in experiment framework to measure the data copy overhead
 - WN \leftrightarrow remote SE
 - Need instrumentation in experiment framework to understand time spent here.

- Pilot Job Framework overheads
 - “wasted time” between “joblets”?
 - Can be up to 10 minutes in some cases for one framework. Worker node is idle in this time.
 - Need instrumentation of framework to measure
 - Time to start first joblet
 - Time between joblets
 - Added in DIRAC 3.
 - Does this explain difference between site and experiment view of efficiency?
 - Or do experiments only measure efficiency for successful joblets? This would introduce a bias

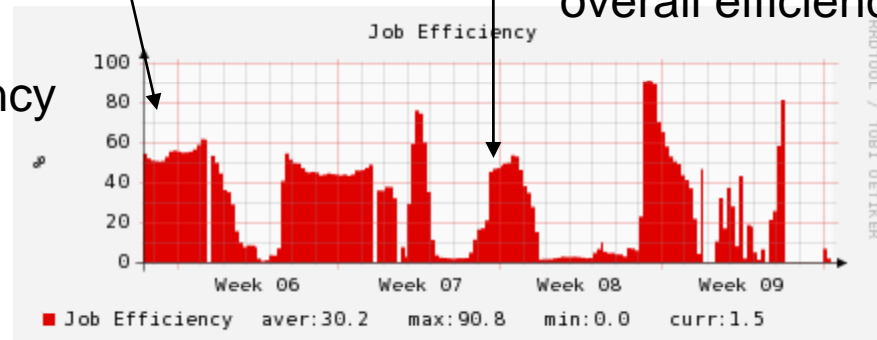
Example: Alice joblet efficiency in the past 4 weeks



70%-90% Alice

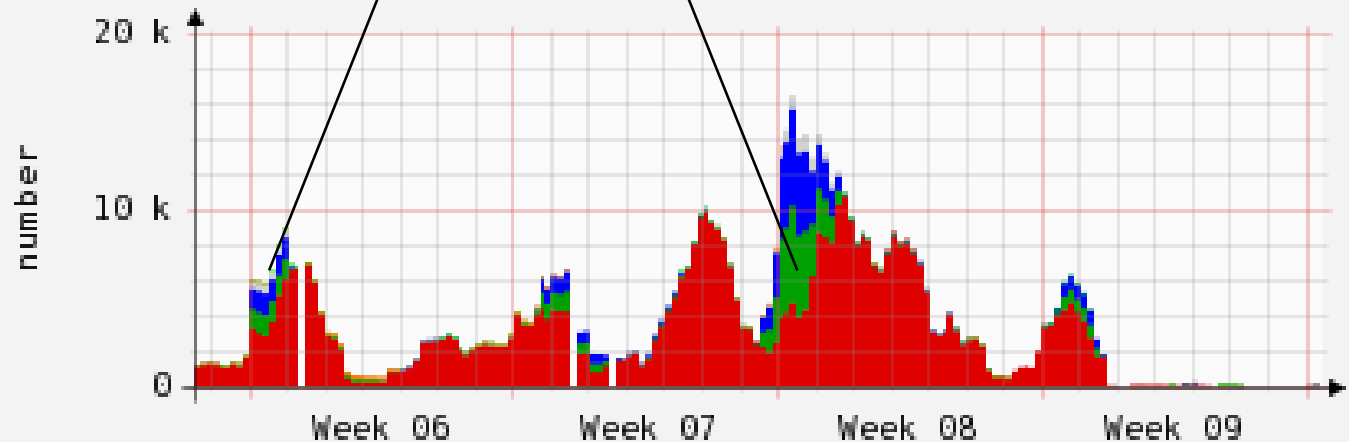
50-60% Alice
overall efficiency slightly less

Overall efficiency
50% only



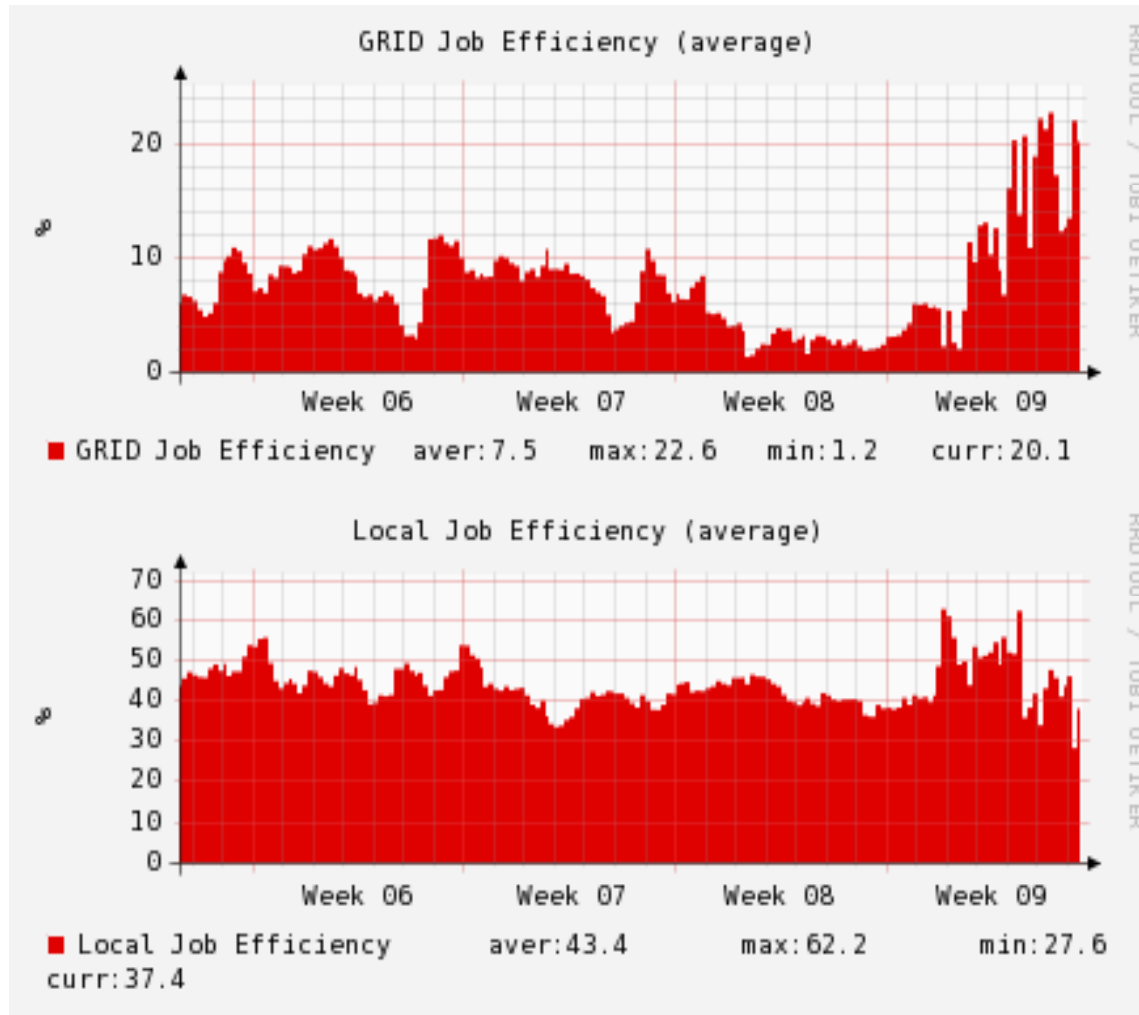
Example: ALICE production jobs in the past 4 weeks

Significant number of killed and exited jobs: - are they counted ?
- what happened to them ?
- need instrumentation of FW



Done State	aver:3.2k	max:10.8k	min:1.3	curr:42.6
Exit State	aver:377.1	max:5.5k	min:0.0	curr:179.3m
Killed Jobs	aver:363.8	max:5.5k	min:0.0	curr:179.3m
Killed Pending	aver:57.6	max:860.4	min:0.0	curr:609.6m
CPU Time exceeded	aver:1.2	max:9.5	min:0.0	curr:0.0
Run Time exceeded	aver:11.6	max:153.8	min:0.0	curr:0.0

- Naturally inefficient (I/O bound) jobs
 - Real world example from LHCb
 - Read 20GB of input data
 - Extract raw data within 288.94s of CPU time
 - Write out 1.6GB of output data (WAN transfer to Rutherford)
 - Estimated theoretical wall time: 31min
 - Maximum of 15% CPU/Wall time possible.
- What fraction of the overall workload do these represent? How can they be scheduled to improve overall efficiency?
 - Tag jobs as I/O bound? How is this passed from WMS to local scheduler?
 - Overall cpu/wall time ratio would not be improved, but sites can schedule I/O intensive jobs to improve box efficiency.
 - Schedule network optimized ?
 - Sites could provide dedicated queues for CPU bound jobs with more powerful machines behind

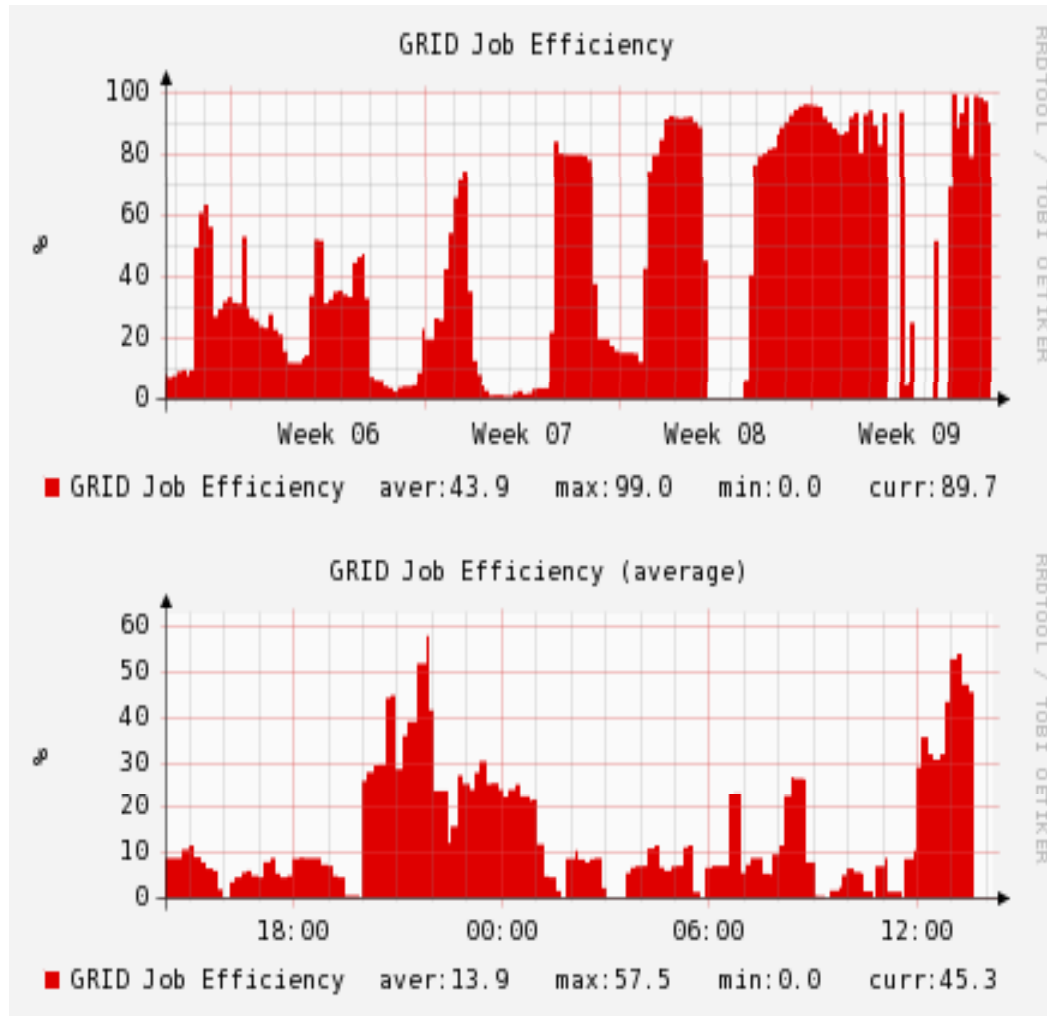


Local jobs appear to be more efficient for CMS and ATLAS.

For LHCb and Alice Grid jobs are more efficient than local jobs

Why ?

Need to instrument the experiment frameworks



LHCb production jobs clearly improved over the last 4 weeks.

Good! But why ?

Other LHCb grid jobs less efficient. Why ?

Key Message:

- need instrumentation so we understand where these differences come from.

- Instrumentation in experiment frameworks to
 - Time job is waiting for data
 - Time between end of one joblet and start of the next
 - Other?
 - Agree common logging format by when?
 - Implement by when? April?
- Remove biases in experiment measurements of efficiency. All jobs should be considered.
- Review in May and June GDBs before and after CCRC

- There is not one single reason for bad CPU/Wall time ratios. There are improvements, but will they be permanent ? We need to find out where exactly the time is spent.
- This can only be done together with the experiments.

Many thanks to everybody who helped to prepare this presentation, and in particular to Latchezar Betev and Philippe Charpentier who kindly provided inputs and material.