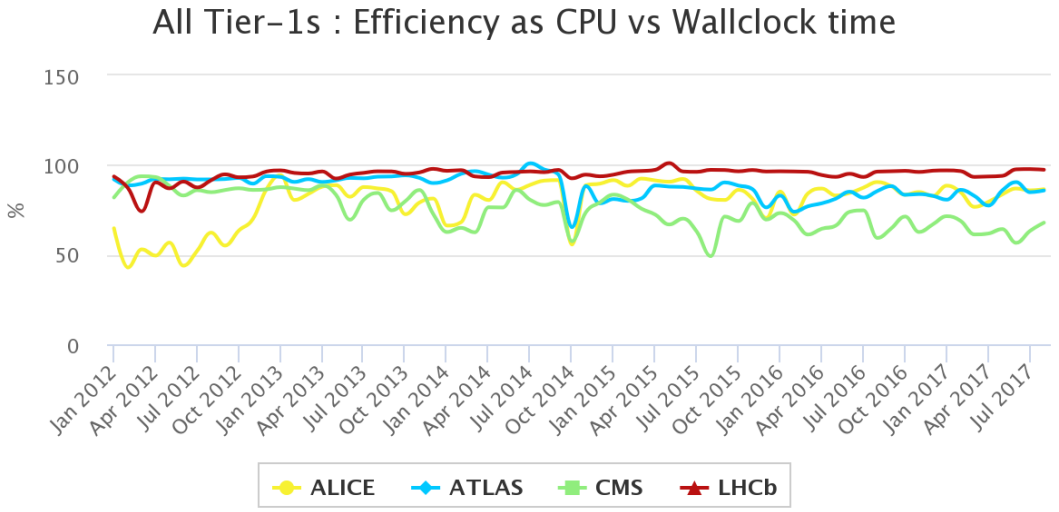
CMS CPU efficiency task force activity

David Colling
(with input from lots of people -Thank You!)

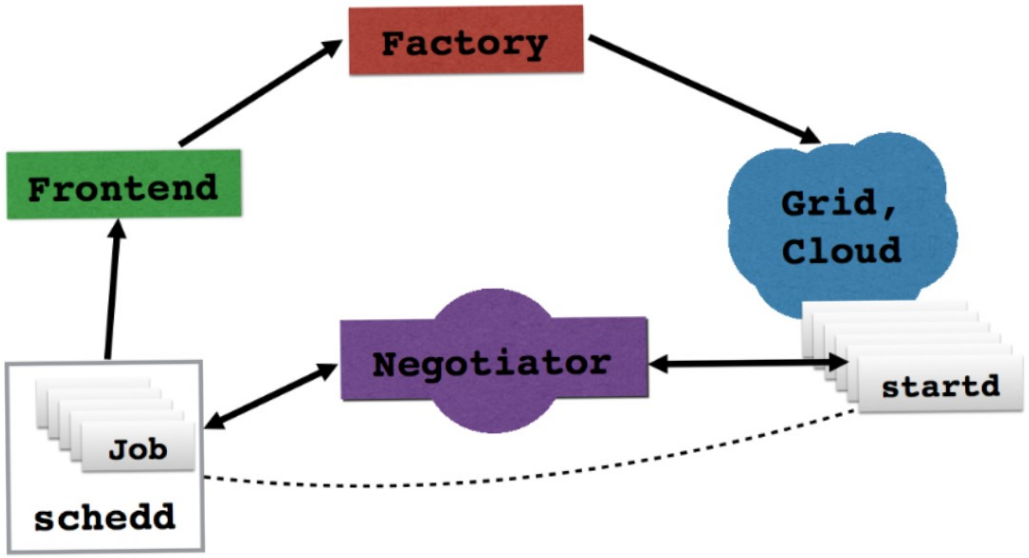
The task force

- Formed May/June last year.
- Contains a wide variety of CMS computing people
- Chaired by the person in CMS computing management who had been complaining most loudly about this issue (me)
- Why formed?

May be not fair to compare with LHCb but ATLAS it certainly is



CMS submission infrastructure – the Global Pool



The policies of the Negotiator are configurable by CMS in the glideinWMS frontend. Currently we match jobs to resources based on a list of DESIRED_Sites and requirements for the number of CPUs (RequestCpus), memory (RequestMemory), and disk (RequestDisk). In the near future, we plan also to provision and request I/O capacity.

The CMS software

CMS is the only LHC experiment to have truly multithreaded software. This has several advantages:

- Reduces memory/core
- Has flexibility to scale workflows from single core to multicore even at run time to match the available cores.
- Can use many core machines (such as KNL). This will bring in an inefficiency but is worth it

However, this will bring some inefficiency (Amdahl's Law), but in controlled conditions our most time demanding applications still have better than 95% efficiency. The Grid is clearly different.

Sources of inefficiency

Broadly speaking, one can factorize the CPU inefficiency problem into two separate sources.

- Inefficiencies associated with the submission infrastructure $= (1 - \epsilon_{\mathrm{In}})$

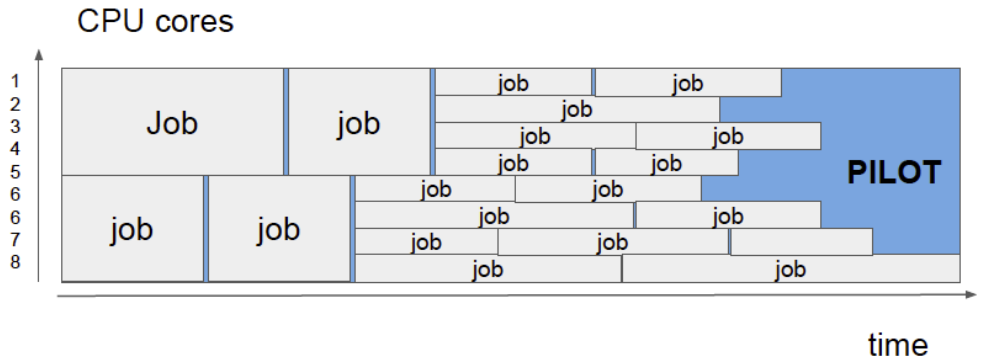
  These accounts for CPUs not being utilized as the pilot infrastructure is not able to direct work to the pilots. This results in unoccupied and inefficient cores.

- Inefficiencies associated with the job itself $= (1 - \epsilon_{\mathrm{Job}})$

  This accounts for CMS software not being able to utilize CPU efficiently, due to factors (for example) such as data access or limitations due to Amdahl's law in multi-threaded applications.

Note total efficiency is the product of the efficiencies i.e. $\epsilon_{\mathrm{In}} . \epsilon_{\mathrm{Job}}$ so inefficiency in both can easily lead to a very low overall efficiency

Submission Infrastructure – multicore scheduling



Pilots typically live for 48 hours

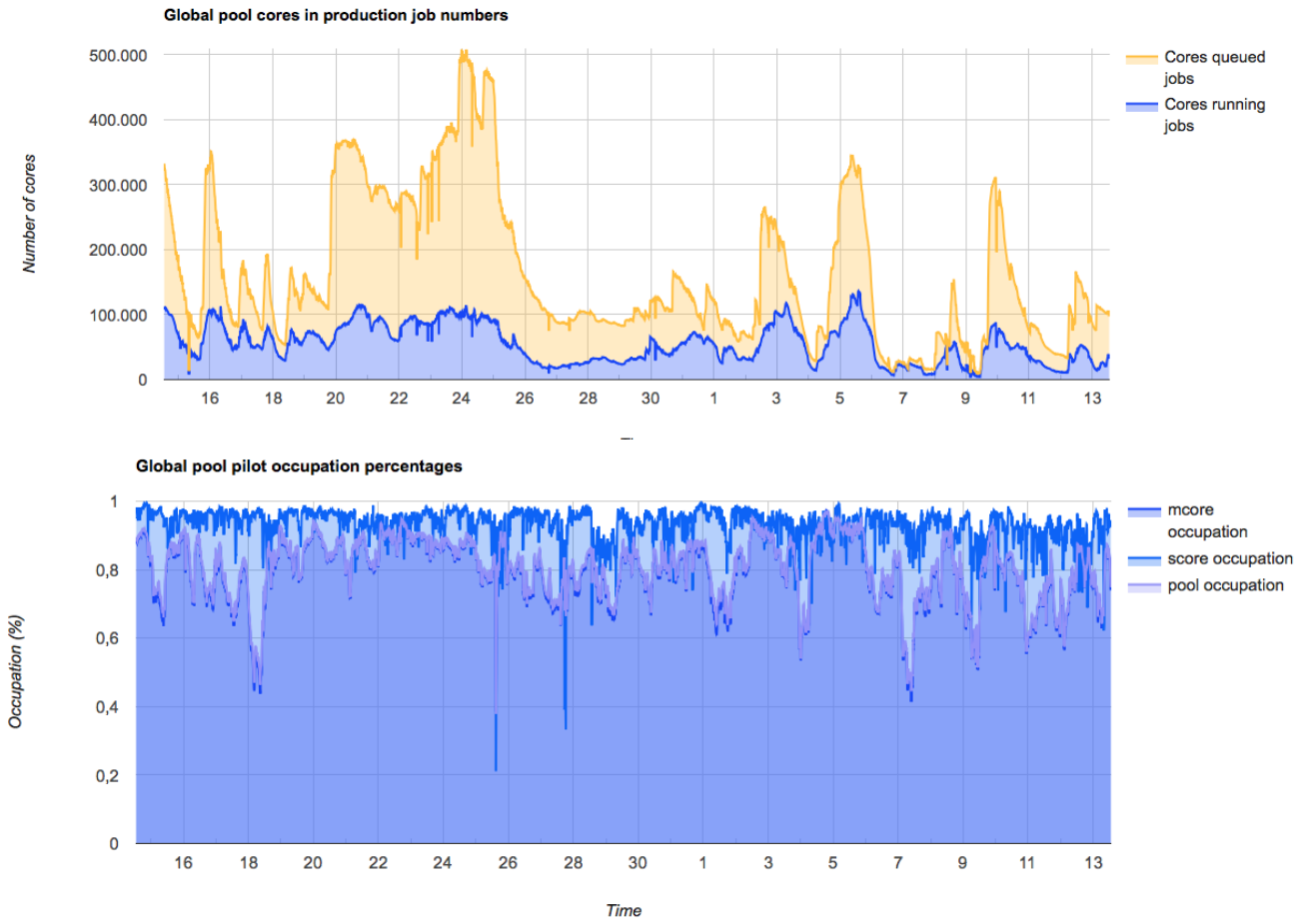
- Often stated as the cause of the inefficiency, despite the inefficiency existing long before we had this model
- Different from any other VO.
- Single core jobs can cause fragmentation and cause problems matching multicore jobs. But renewing finite lived pilots provides unfragmented pilots to the pool.
- Re-use of pilots in the CMS model by multiple jobs saves several minutes of wallclock time per pilot overhead in re-scheduling and re-validating pilots as well as re-negotiating schedulers' access to them.

Legitimate causes of unused cores in glideins

- High memory jobs
  - The CMS VO card says that we require 2GB/core. If we have jobs requiring more than 2GB/core then the memory can be exhausted before the CPUs.

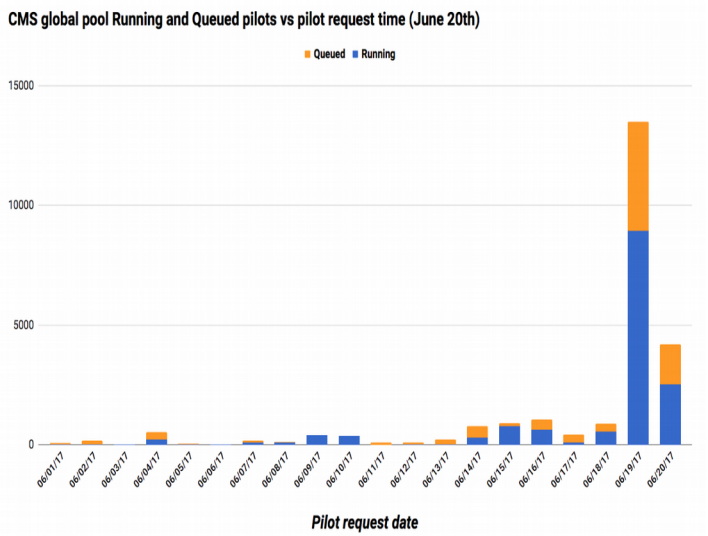
- The HLT farm
  - The VMs running on the older generation of machines have less than 2GB/virtual core.
  - For sometime the HLT has been able to suspend jobs and restart them up to 24 hours later depending on data taking and it use as the HLT. The accounting records the CPU time used and the total wall clock time – INCLUDING the time when it was suspended. This gives very poor CPU efficiency results!

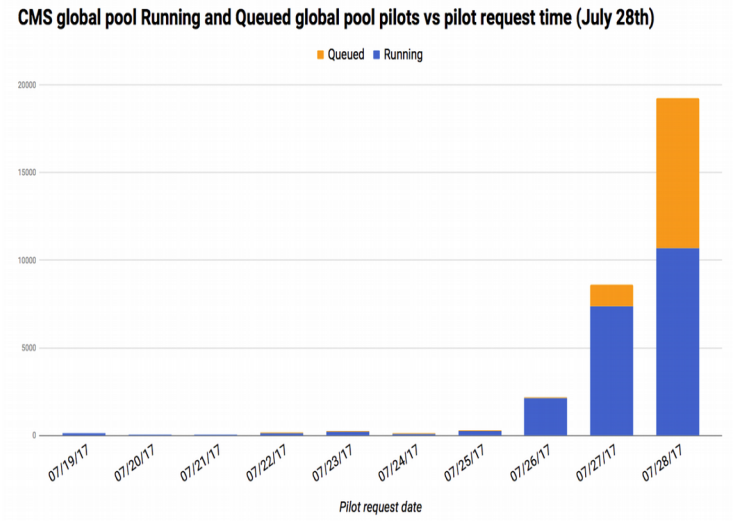
Inefficiency and job pressure patterns



Fluctuating Job pressure can cause dips in core use in glideins as the resources are no longer needed. This was exacerbated by pilots being started in response to pressure that was up to a week old as they had been queuing for that long and were no longer needed.

Inefficiency and job pressure patterns





Glidins queued up to 2 weeks in June 2017

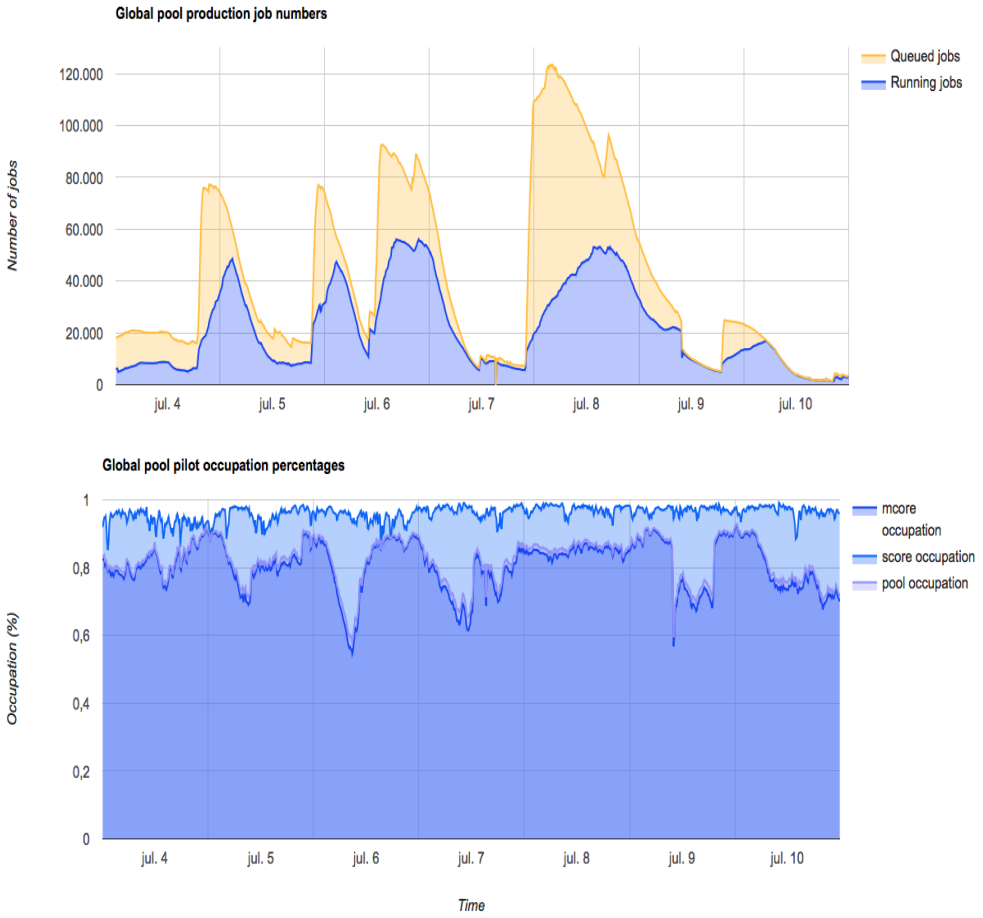
End of July 2017.

Introduced 3 hour limit on idle glideins in a queue on the 3rd July 2017.

Further reduced to 1 hour on the 24th July 2017.

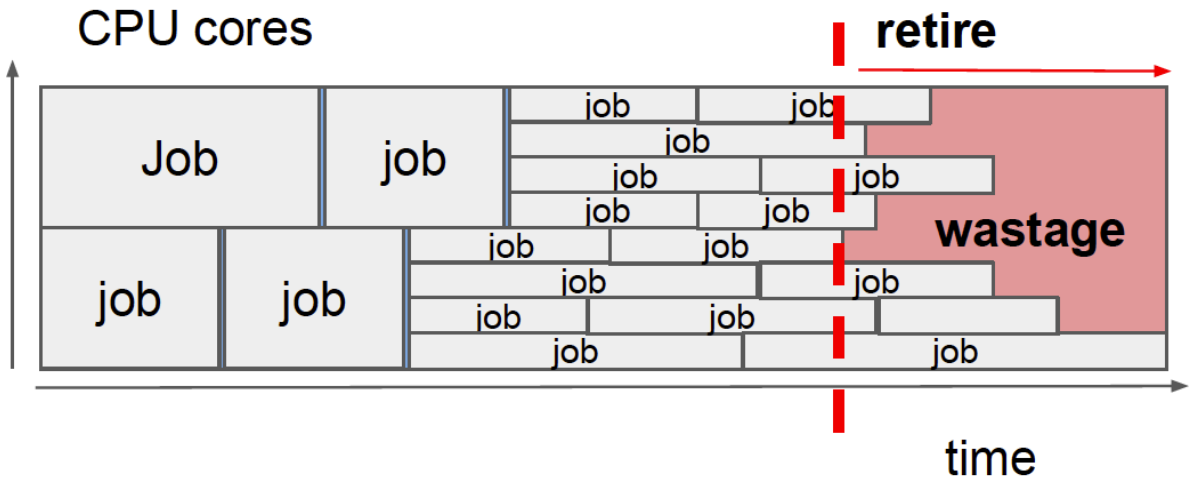
Can potentially cause churn is site batch queues.

Inefficiency and job pressure patterns



- After July fixes fluctuating job pressure still causes issues
- Perhaps the glideinWMS Frontend could actively manage the number of queued pilots in a continuous way, not only requesting new pilots be submitted when necessary given a increase in demand, but also cancelling those which are no longer needed when job pressure shrinks.
- Perhaps the introduction of backfill (like ATLAS)

Retiring Glideins



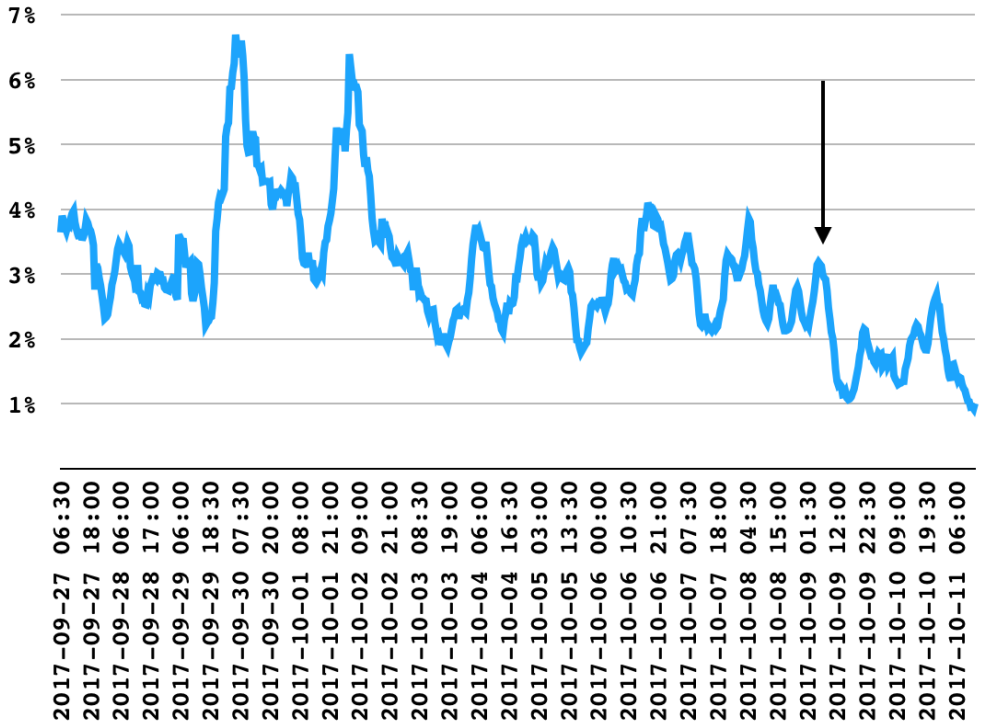
Note that this graphic does rather exaggerate the effect.

At some point the glidein stops accepting more jobs and allows the existing jobs to complete without being killed by the pilot being terminated. This period had been set to 580minutes (9:40).

Once a glidein has no more work it waits a period to allow for the condor negotiation process to see if it is still needed (was cut from 20mins to 10 in July) and then terminates. A multicore pilot must wait until after the last job has completed.

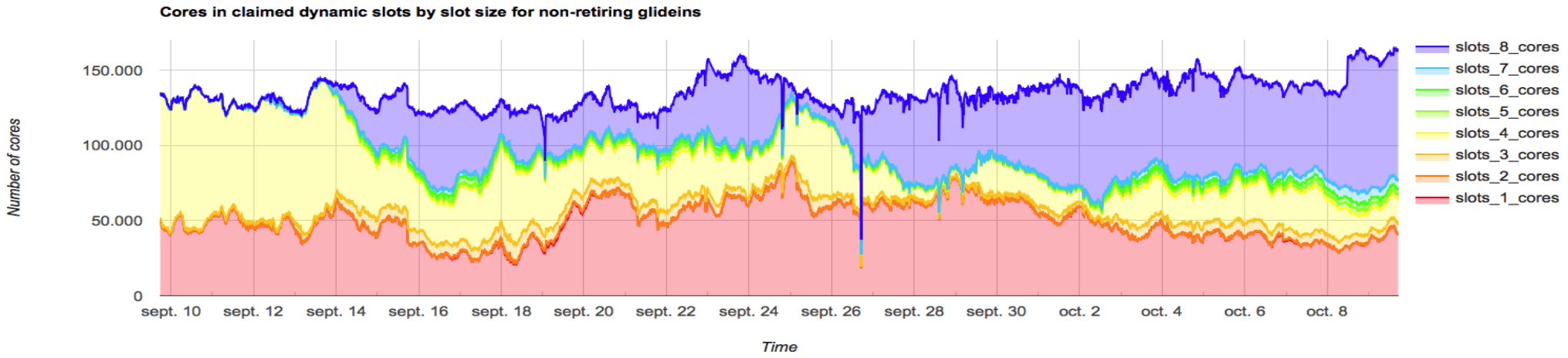
The retiring time was cut to 240mins in October

Retiring Glideins



Effect of introducing the change can be seen. However the retiring effect was already lower than historical values because we were running a large number of 8 core production workflows in 8 core pilots.

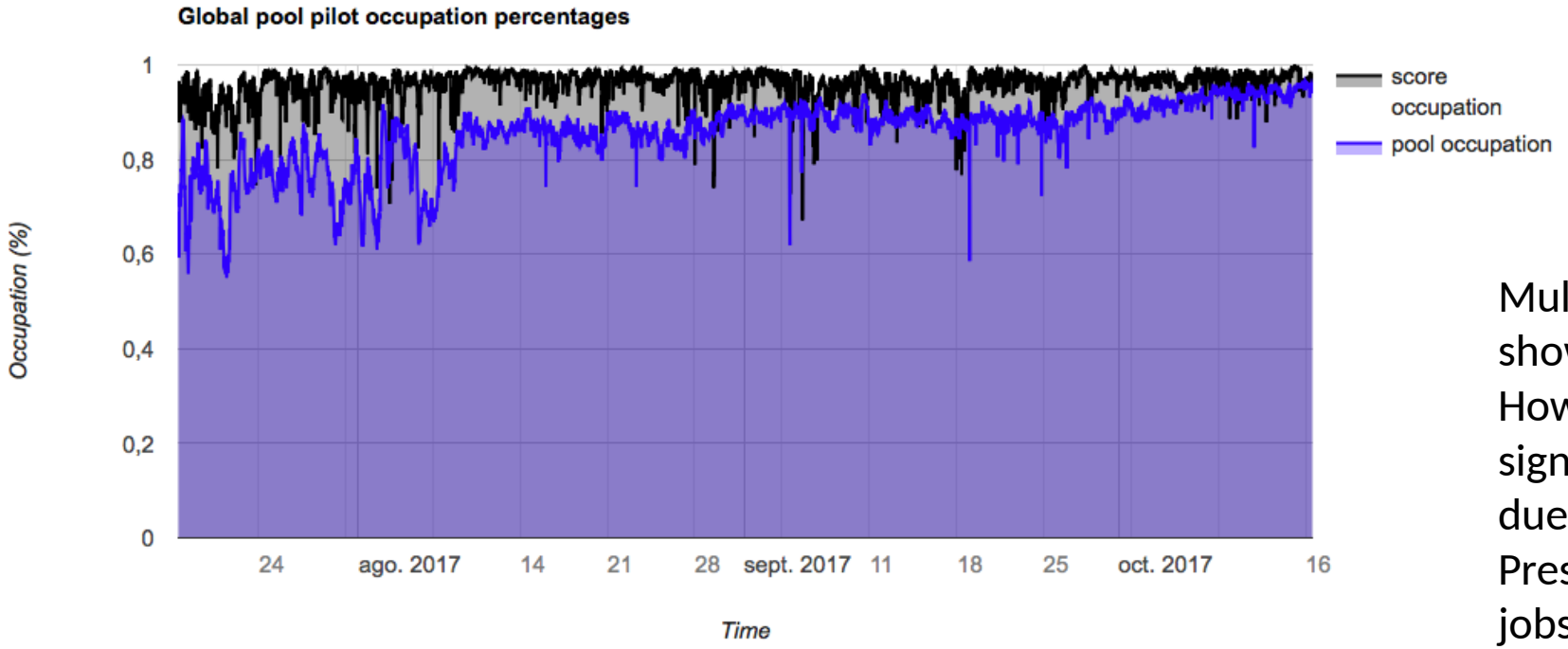
How many cores



[avg, min, max]: [49438.0, 18388, 89122] [140.0, 0, 2136] [5874.0, 729, 12552] [34545.0, 3196, 94984] [1851.0, 0, 4305] [1679.0, 0, 4854] [2168.0, 0, 7931] [38805.0, 0, 95632]

Global pool fragmentation: distribution of dynamic slot sizes weighted by the number of allocated CPU cores, for the period of 30 days since Sept. 9th.

The effects of changes

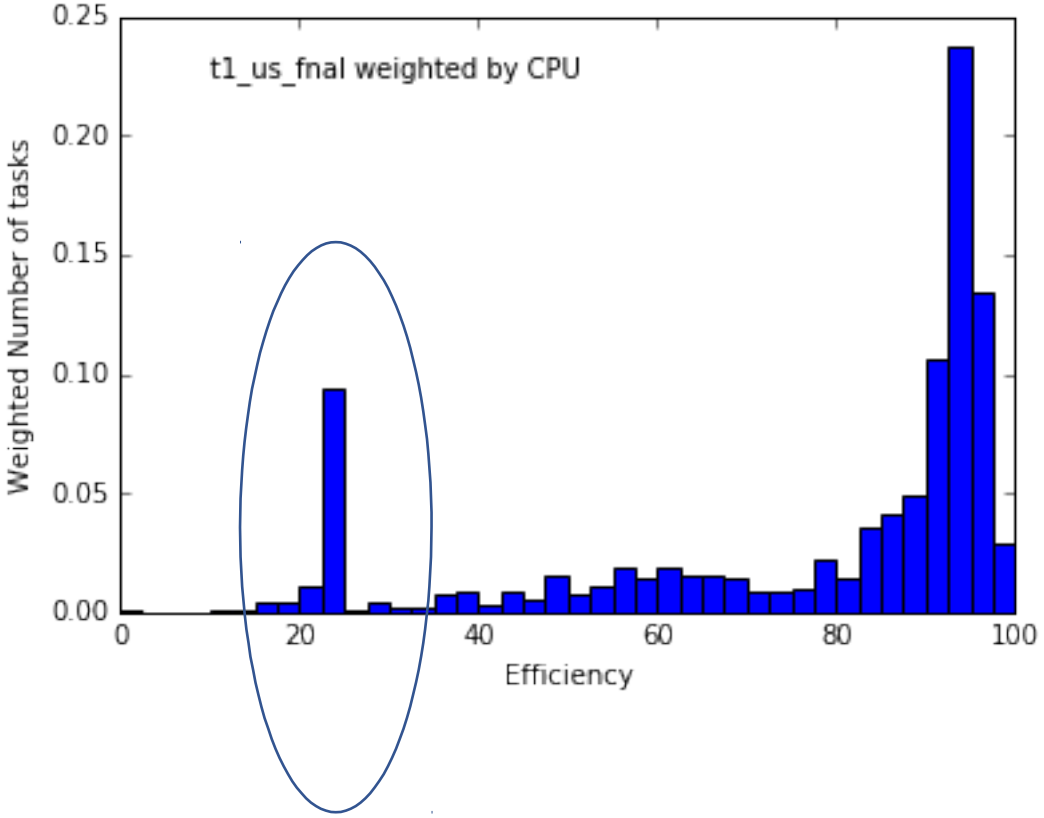


Multiple changes show improvement. However some significant fraction is due to consistent Job Pressure (with 8 core jobs)

Turning to the payloads

- Having concentrated on the infrastructure for the first few months of the task force, we then started to look at the payloads themselves.
- Main tool is data collected in elasticsearch at CERN.
- Some of the details of the plots that I show are wrong for technical reason but the overall message is correct (I could explain the caveats on plot but that would be tedious)
- Found that different job types told different stories.
- These investigations are current and I will concentrate on a few job types as examples. Not much of what this shows is yet implemented.

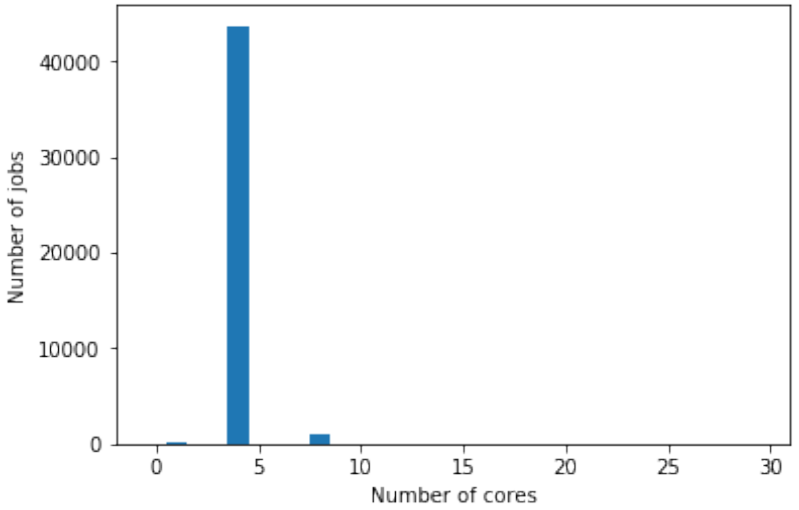
Misconfiguration

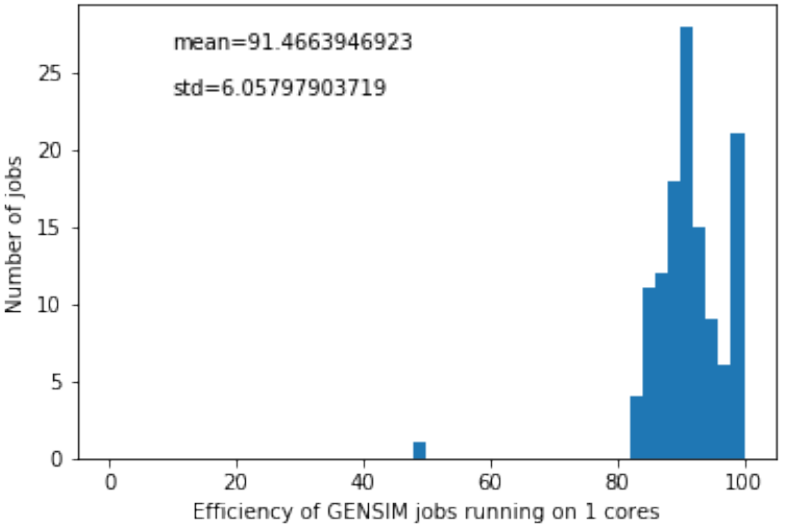


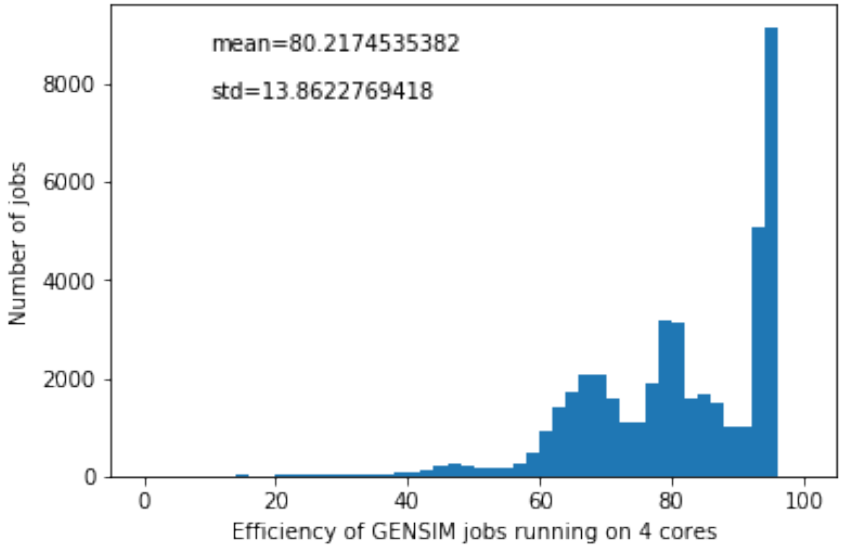
Only using 1 out of 4 cores

- One of the first things that we found was some production runs had been misconfigured
- CPU efficiency had not been part of the standard checks performed by production before a workflow is submitted. It is now so hopefully these are a thing of the past.

GenSim

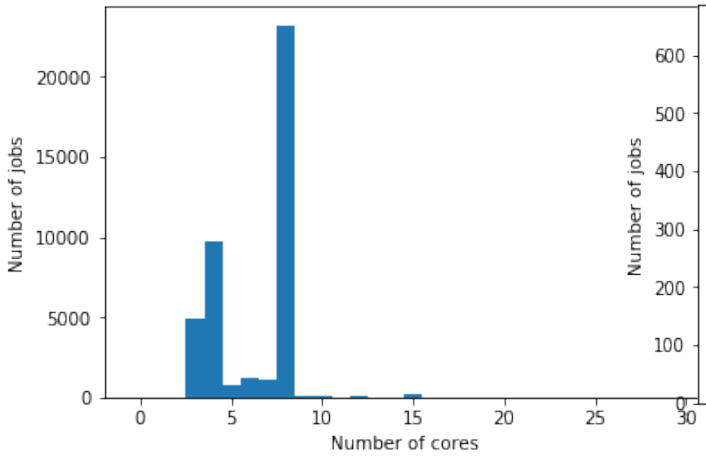


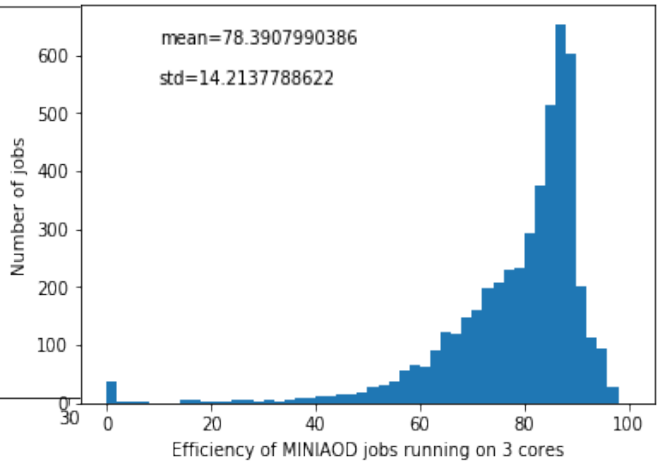


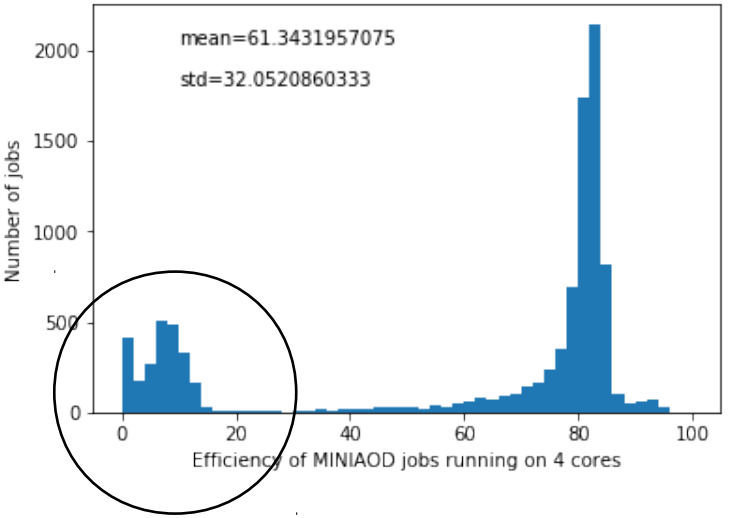


- GenSim should be >~90% efficient
- But plots show that it clearly isn't (and I have seen it far worse than this)
- The major problem here is that while CMSSW is truly multicore very few of the generators are so that we spend some fraction of each of each multicore pilot running a single core only.
- Had meeting with generator group group last on how we can fix this and there are some plans.

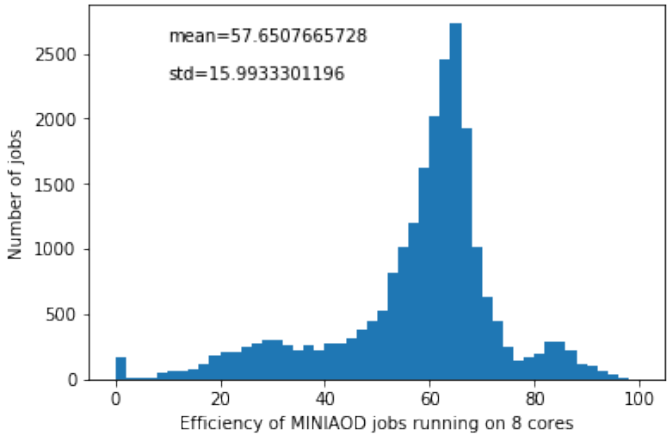
MiniAOD production







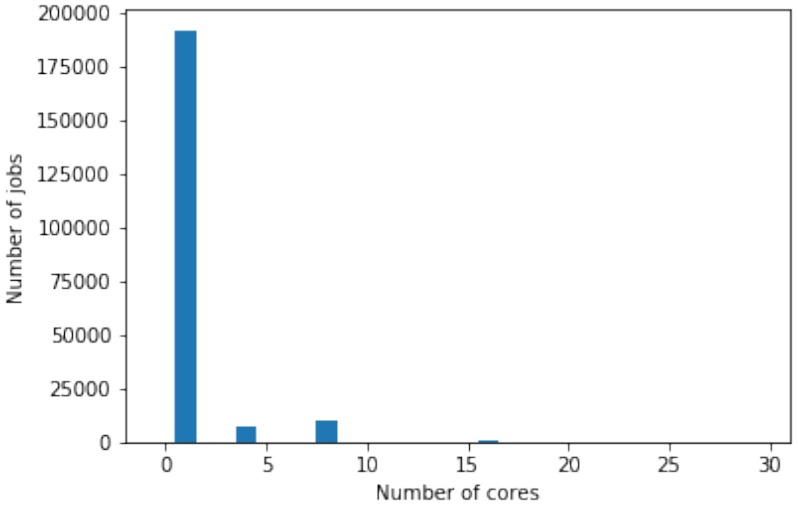
Ignore this it was an artefact this weekend . Didn't use CPU/wallclock (I think)

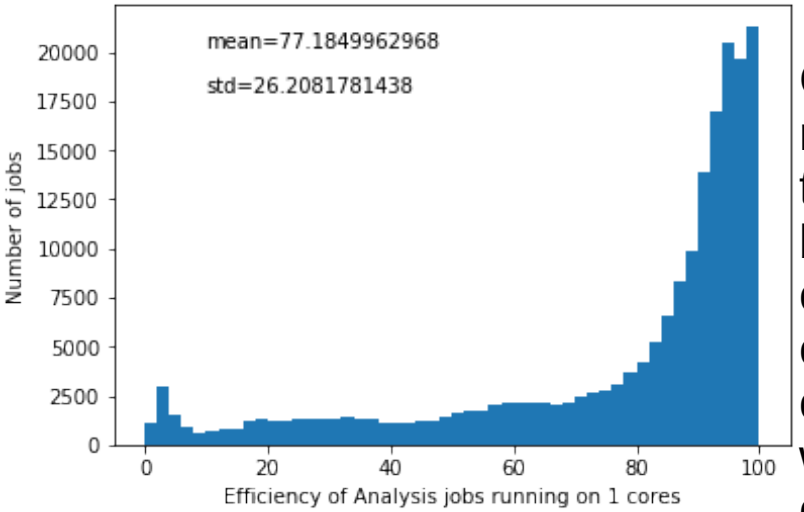


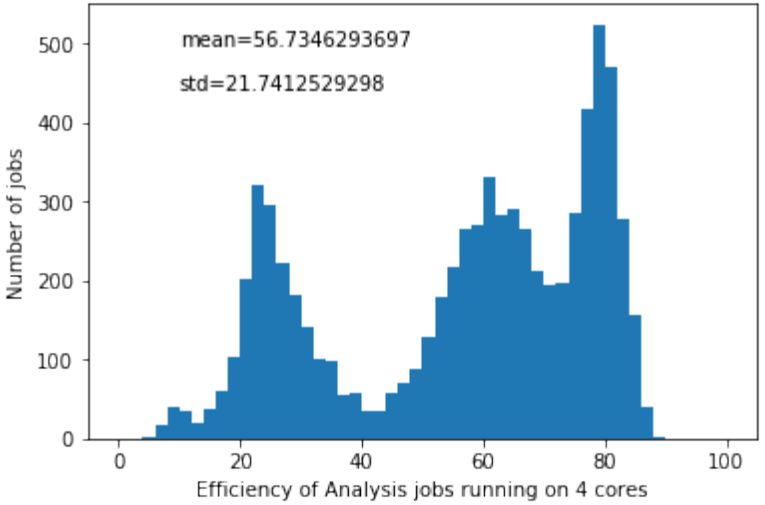
Is IO bound so throwing more cores at it makes the efficiency decrease. Yet 8 cores was most popular.

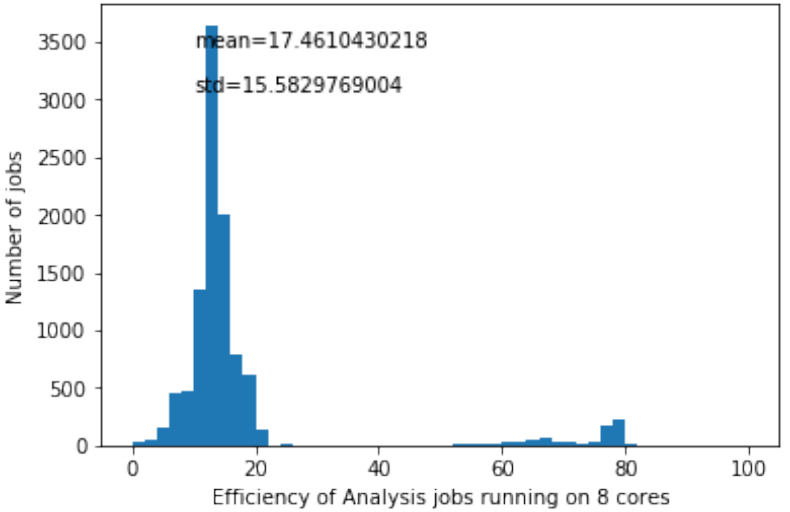
Decided not use more than 4 cores in future.

User analysis







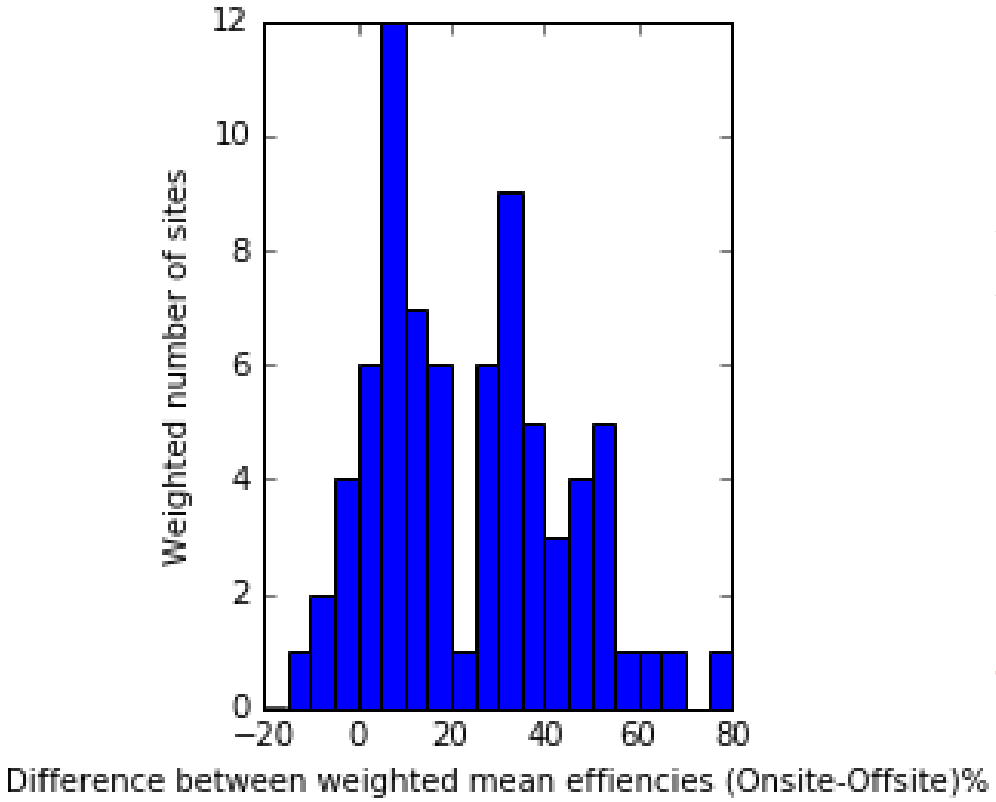


Ahhh…
Users, don't you just love'em?

CMSSW is multithreaded so their analyses can be as well and we even mildly encourage it, but only if they know what they are doing.

This has caused discussion over the weekend and so the first thing is that users will get a message if they try to submit multicore analysis jobs asking them if they really want to do that

Onsite v Offsite data



Note that this plot is weighted by the CPU time used by jobs using onsite and offsite data at a given but then all sites are treat equally so it doesn't tells very much of overall use. However what it does tell us is that there is a huge variation between site. This needs further investigation.

However, not all the sites with large differences are small.

That said, offsite data is an essential part of CMS computing (not least for the premix libraries).

DIGI

- I had hoped to make a few plots here, but if the merest whiff of wifi in my room was not enough to listen to the Today programme at 6am it was never going to be enough for an ES query. By 8am indico was a step too far. So you have a narrative instead.
- CMS serve pileup from CERN and FNAL to all DIGI jobs (Premixing). Saves vast amounts of disk space and generally works fine (small efficiency loss but worth it). However sometimes it doesn’t… still under investigation.

Grim Reaper time?

Looking at individual log files we found some jobs that would pause. Sometimes for considerable periods but then continue normally. These are still under investigation.

We do have the ability to “reap” the jobs but we want to understand what is going on before we consider this approach

Conclusions

- The taskforce has been in existence for ~10months
- The situation is more complex than we thought.
- Can factorise infrastructure and payload inefficiencies but it is the product that counts.
- First concentrated on infrastructure but now also following the different payloads.
- More work to do and fixes to implement.