# CPU efficiency

- Since May CMS O+C has launched a dedicated task force to investigate CMS CPU efficiency
  - We feel the focus is on CPU efficiency is because that is what WLCG measures; **punishes those who reduce CPU needs** or where one **reduces CPU efficiency to get better physics throughput**.
  - Still, task force has found opportunities to improve CPU efficiency *and* resource utilization *and* physics throughput.
  - TF chaired by D.Colling, UK/IC

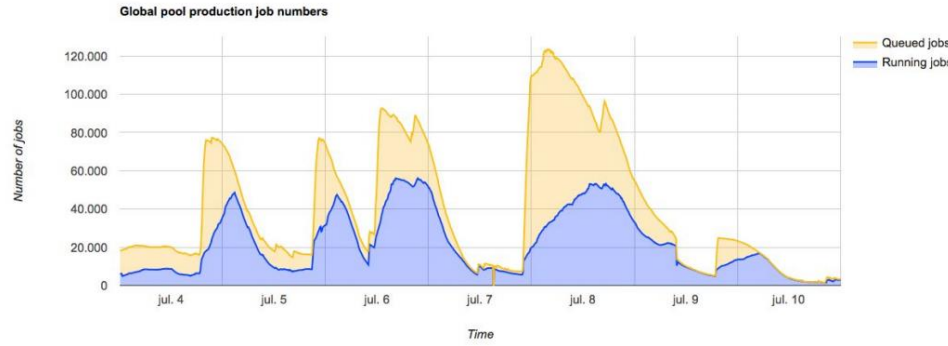
- For a significant amount of 2017, we were not CPU resource constrained. Situation is changing in the last month; insufficient statistics for some plots.
- Still, a lot of gained understanding which led to implemented solutions
- Global efficiency can be factorized into infrastructure_efficiency * payload_efficiency

Pilot infrastructure, interaction with batch systems, job matching, pilot draining …

Calibration / code loading IOWait, final stageout, job length …

# Infrastructure “inefficiency” – implemented solutions

- Causes and mitigations for idle CPU, with estimations of their contributions to the average **~10%** and **~15%** maximum sustained idle CPU due originally to multi-core glidein scheduling issues.
- EARLY IMPROVEMENTS:
  - Implementation of depth-wise filling of glideins (pilots) in June (improvement estimated at **~1-2%)**
  - Shortening the amount of time a glidein can sit completely idle from 20m to 10m (July), driven by Negotiator cycle length (improvement estimated at **~1%**)
  - Removing glideins from site and factory batch queues after a certain amount of time (1-3h) in July:
    - When job submission is bursty, we found that the glideinWMS factory had requested new glideins which finally ran hours or even days after job demand evaporated resulting in massive, unpredictable wastage of CPU, up to **60%** over many hours.
    - Had the effect of more tightly coupling demand and supply of resources, but perhaps not the perfect solution (causes some batch queue churn at sites by removing jobs).

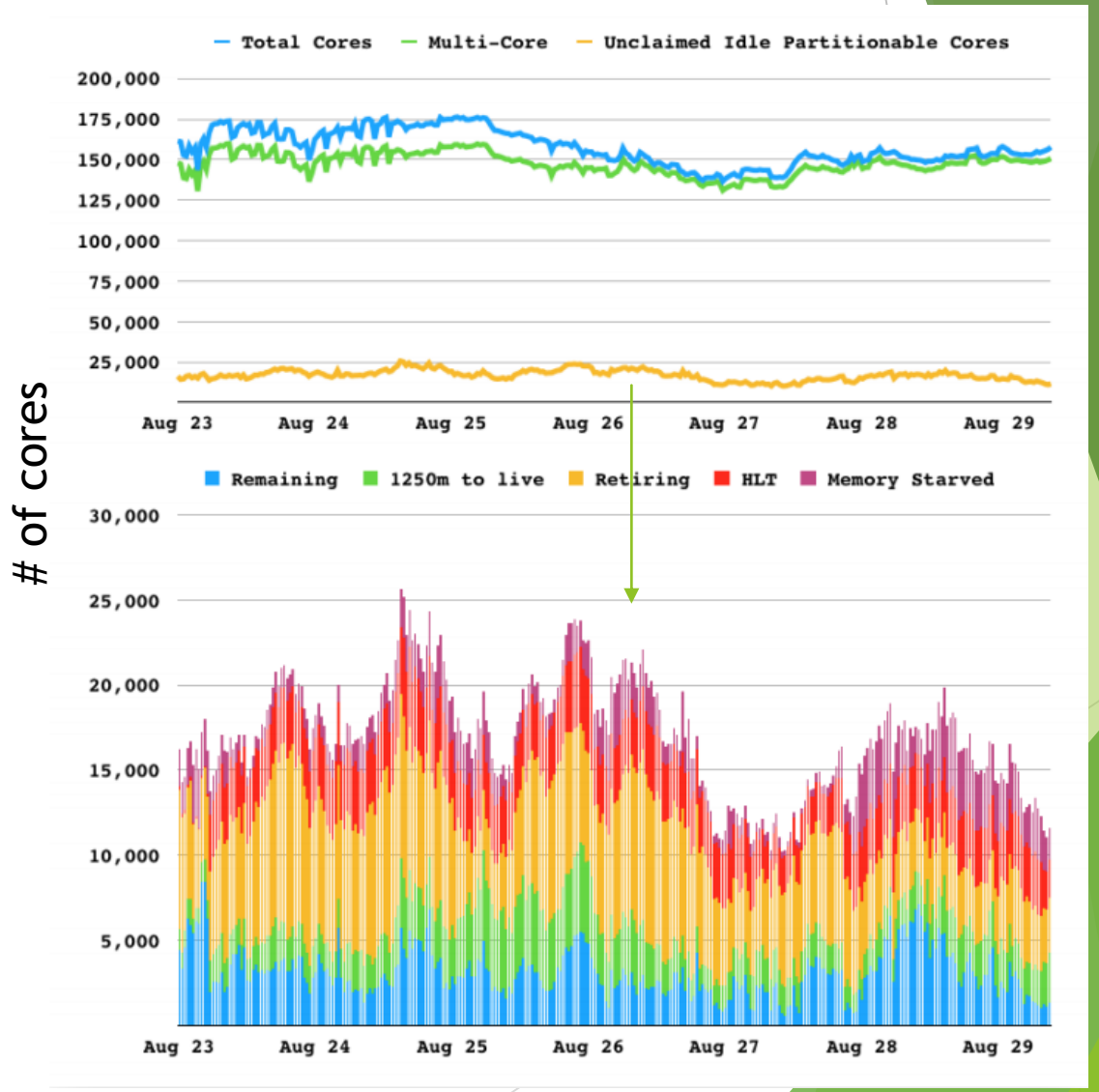



# Infrastructure “inefficiency” – analysis and plans

- Reference study period after improvements, with sustained job pressure**: Aug 23rd-29th, 2017**. Total remaining idle CPU in multi-core pilots: **11.4%**.
  - Legitimate use case for leaving CPU idle: high-memory dynamic slots for RAM-hungry workflows (**3.4%**) on the Grid and the HLT.
  - Analysis jobs with unrealistic wall clock time requests cause idle CPU (**1.6%**) since the finite-lifetime glideins (typically 48h) will not live long enough to match them.
    - Automatic job splitting in CRAB3 under test
  - Draining glideins after 30h (**4.2%**): We are currently optimizing against job failure rates and may consider not draining at all to eliminate this component.
  - **Total understood: 9.2%, Remaining: 2.2%, or ~1h avg. for a 48h pilot.**
  - **About ~6% of next gains may come from improving job wall clock time estimates and eliminating draining of pilots.**
    - Reference period was during steady production job pressure, not bursty submission. Comparisons with bursty mode in previous weeks shows that this increases idle CPU (due to additional cyclical draining of pilots) an additional **~6%** (even after limiting idle glidein queue time to 1h).
    - Scale tests in the Global Pool ITB with über-glideins may allow us to estimate the individual effects of improvements by backing them out one-by-one

# Submission Infrastructure Monitoring

- Multi-core pool to >97%
- Breakdown of idle core reasons: note the order of magnitude difference in scale.
- HLT & Grid memory starved slots (red and purple) a valid use case from high-memory workflows
- Retiring policy and requested job length accuracy can fix the yellow and green, in principle
- Remaining 2% (blue) can be investigated but is possibly irreducible.



# CMS (Payload) Workload

- Over the last 1B of core hours, here's the approximate usage:

| Job Type | Percent CPU resources | Payload CPU efficiency | Percent of idle *payload* CPU |
|---|---|---|---|
| Analysis | 37% | 64% | 41% |
| Simulation | 27% | 74% | 22% |
| MC reconstruction | 27% | 68% | 29% |
| Data reconstruction | 7% | 73% | 5% |
| Other | 2% | 56% | 8% |
| | Overall: | 68% | |

- For each row:
  - Different source of inefficiency. Some inefficiencies may be irreducible.
  - Different approach (and cost!) for improvements.
- Goal: given finite effort, maximize payoff.

WLCG CPU efficiency =
(pilot scheduling eff) *
(payload CPU eff)
**WLCG measurements roughly match collected CMS monitoring!**

# Difficulties: MC reconstruction

The most CPU efficient workflows are those with infinite loops.

- **Example**: CMS MC reconstruction can be done in either **premixed** or **classic** pile-up.
- Premixed mode reduces CPU-per-evt by **2x** and data-per-evt by **10x**.
  - It also requires 10x larger background (**~1PB of disk**).
  - We host the samples at large disk sites (FNAL, CERN) and stream them to reprocessing sites; these provide **30% of the CPU resources** for this campaign
  - ~60% CPU efficiency overall; 65% CPU efficiency when run at CERN / FNAL.
- Today: **We take the CPU efficiency hit and deliver >2x more events per month than prior approach.**
- Alternate options with higher CPU efficiency:
  - Only run premixing at CERN and FNAL – far less CPUs.
  - Run classic mode only – involve more sites, but far slower application.
  - Distribute premixed background more widely
- Also under investigation: Have ROOT better interleave IO and CPU usage.

**Take home message**: We sometimes trade off CPU efficiency for better physics or to avoid IO (which is not metered by WLCG).

# Difficulties: Input Data

Much ado about a measurable quantity.

- Over the last year, **17% of our wall-clock hours went to jobs that read data from offsite.**
  - On average, this **reduces CPU efficiency by 10%.**
- The CPU-efficiency cost of remote reads is about **2%** globally.
- This assumes we have other jobs that could have utilized the same CPU: **bad assumption**, given we spent most of 2017 under-utilized.
  - Sites with "popular" input data and performant storage may have less remote-IO jobs.
- Today: **We take the CPU efficiency hit.**
  - Cost: 2% decrease in overall CPU efficiency.
- Alternate options:
  - Run lower-priority jobs.
  - Leave sites or cores unused.
- **Simulation jobs (no input dataset) are 27% of CMS's CPU workload.**
  - Remaining requires varying amounts of IO.

**Take home message**: Remote IO provides us with the flexibility in workflow planning; complex tradeoff of utilization, prioritization, and CPU efficiency.