# Exploring GlideinWMS and HTCondor scalability frontiers for an expanding CMS Global Pool

**Antonio Pérez-Calero (PIC & CIEMAT)**
**on behalf of the CMS Collaboration**

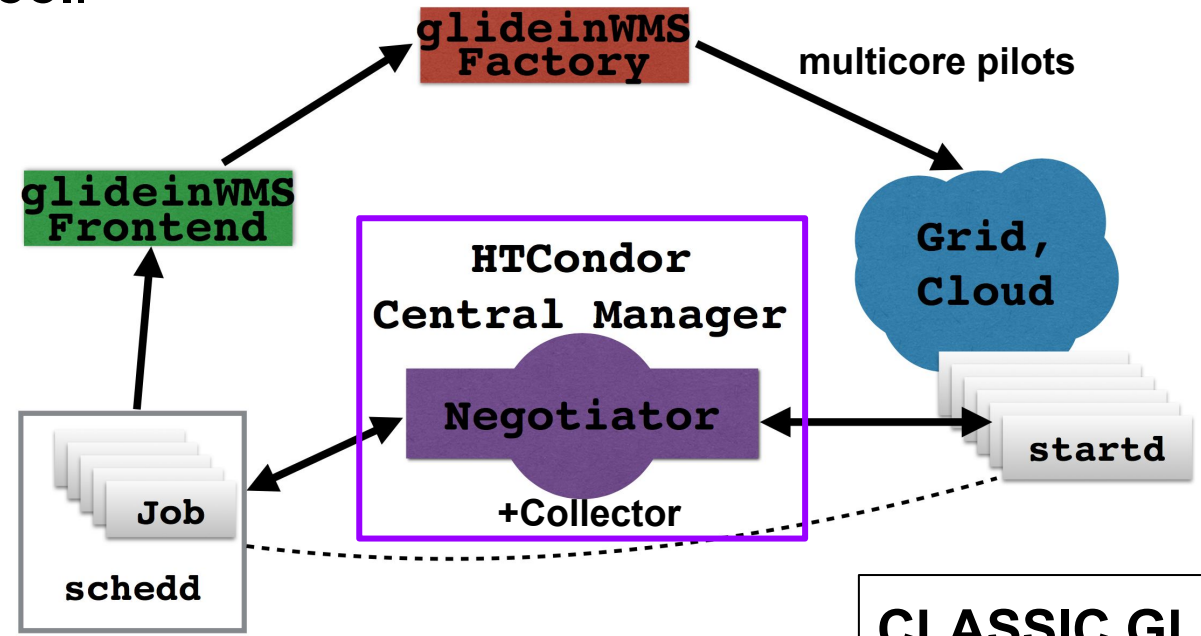

PIC port d'informació científica
IFAE Institut de Física d'Altes Energies
EXCELENCIA SEVERO OCHOA
BIST Barcelona Institute of Science and Technology
Ciemat Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas
EXCELENCIA MARÍA DE MAEZTU
CIEMAT física de partículas

# Outline

- The CMS Submission Infrastructure (SI) Global Pool and its current scales
- Motivation for HTCondor and GlideinWMS scale tests
- Scale tests setup and recent results
- Conclusions

# CMS Global Pool

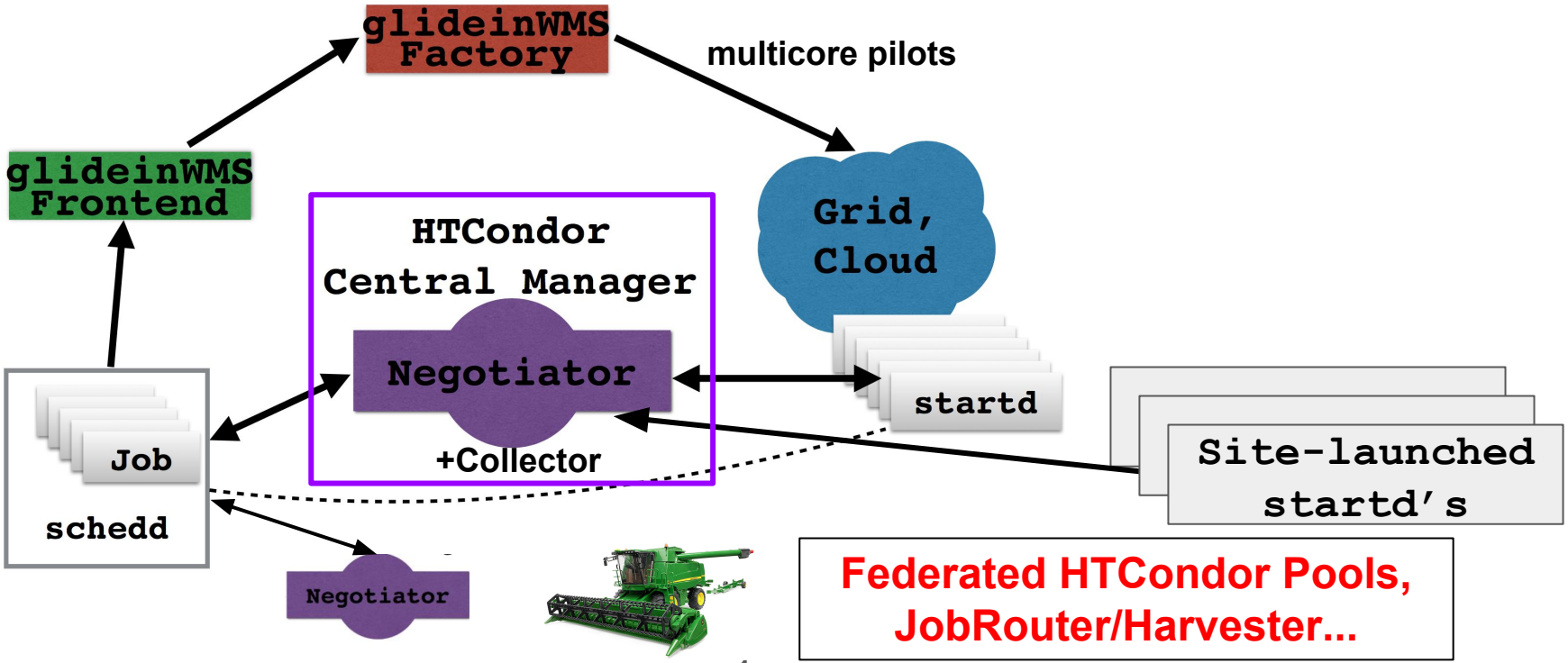
The CMS Global Pool is both a **glideinWMS instance** and a **HTCondor pool**.




**CLASSIC GLOBAL POOL VIEW**

# CMS Global Pool

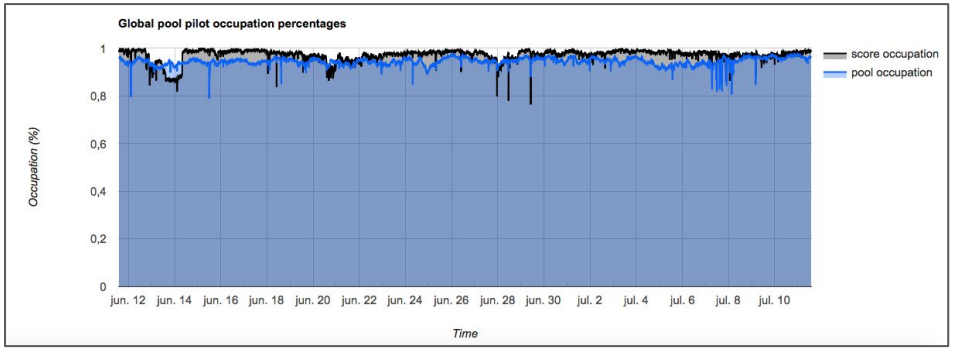
Already the resource and submission landscape is shifting …

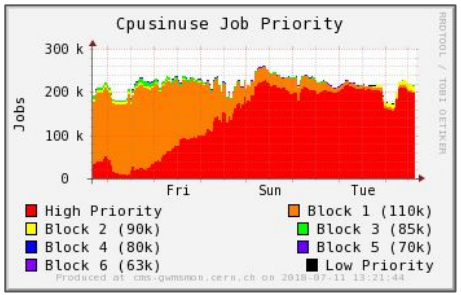


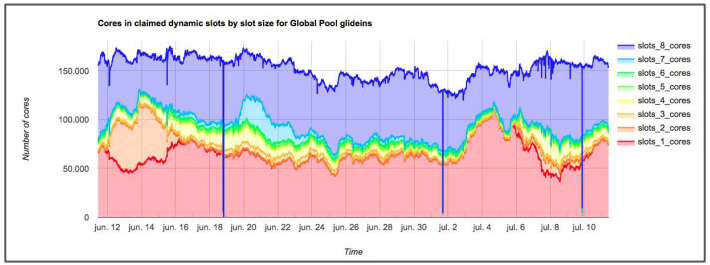
# CMS Global Pool

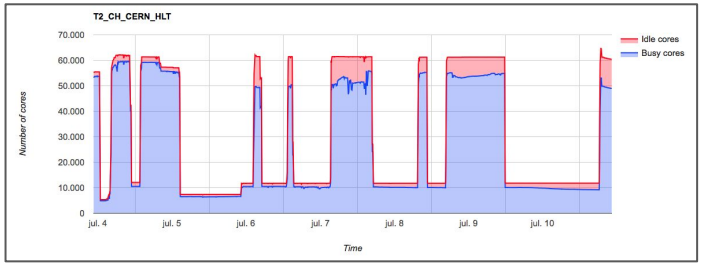
**The CMS Global Pool**, designed and implemented for LHC Run-2, **is a very successful infrastructure for CMS needs**, capable of:

- Single entry point for diverse job resource requests (e.g. single-core, multi-core, high mem, GPUs)
- Efficient scheduling
- Global Workload Prioritization
- Global Fair Shares (production, analysis)
- Integration of dynamic resources (e.g. HLT)



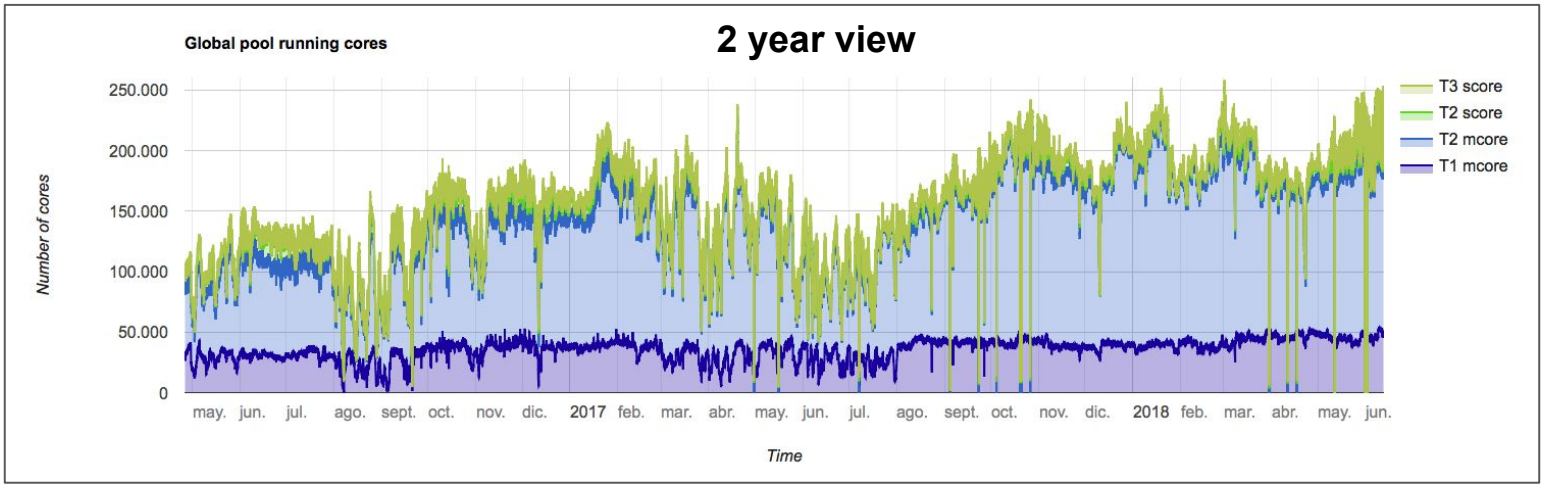



# Motivation (I): Increasing Scales

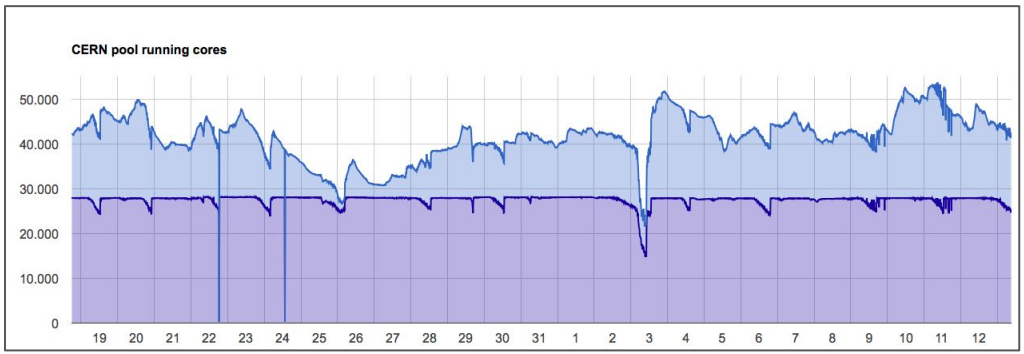
- **Size of the Global Pool** driven both by
  - resource **deployments** (tend to happen later in the calendar year)
  - resource **requests** (e.g. compare quiet 2016 and 2017 Summer months with now)
- During Run 2, **HLT farm** commissioned as opportunistic resource (peaks of **60k cores** in 2018)
- Also added **HPC** (e.g. CSCS), **Cloud** (e.g. HNSciCloud) and **opportunistic** (e.g. T3_US_OSG) as new resources into the pool
- **Routinely running at 250k cores nowadays**

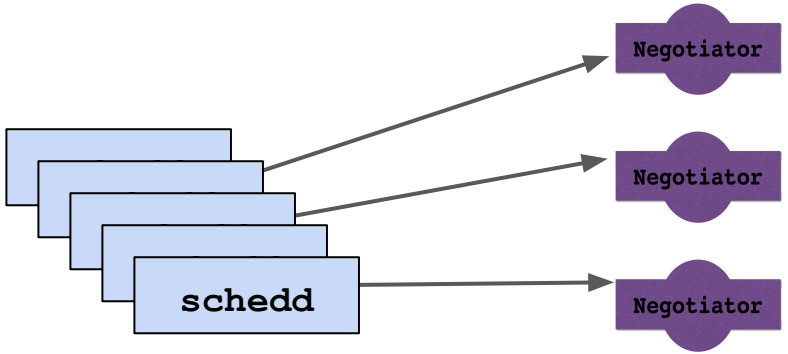


# Motivation (II): Multiple Pools

- **Global Pool model already evolving!**
  - **CERN Pool**: For the 2018 data taking period CMS and CERN decided to move away from Tier0 dedicated Openstack VMs on to HTCondor-managed resources.
    - **Two resources at CERN**: for **Tier0** tasks (shared with ATLAS) and the general multi-VO **shared** pool.
    - CMS decided to **merge** both a single **CERN pool**, **independent but federated to the Global Pool**
  - **HEPCloud** in the USA (joining grid, cloud and HPC), other (many?) national Clouds coming soon in Europe.
- Moving into a model of multiple **Federated Pools** with **centralised submitters** (schedds)
  - How many pools (Negotiators) can a schedd talk to before things break down?
    - CMS already using 3 (Global) + 1 (CERN) negotiators

**40k-core new pool for CERN resources**



# Motivation (III): HL-LHC Challenge

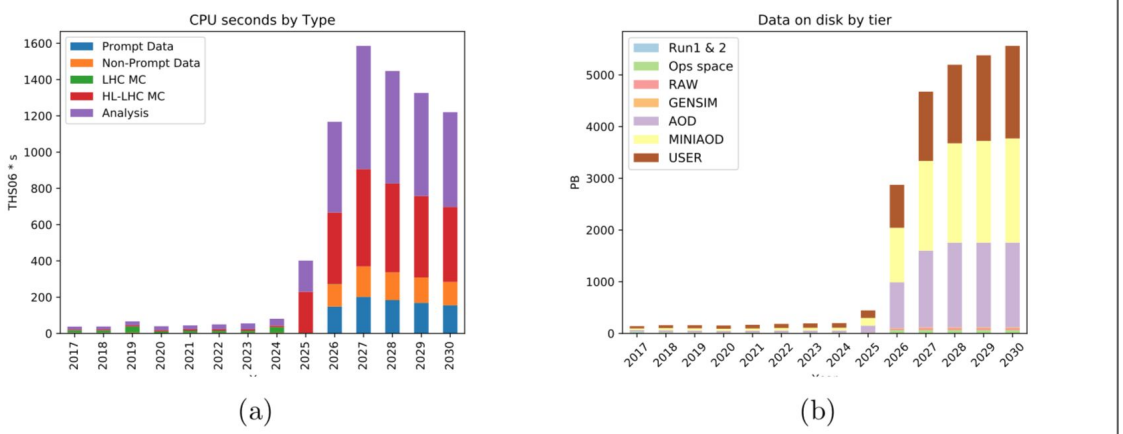
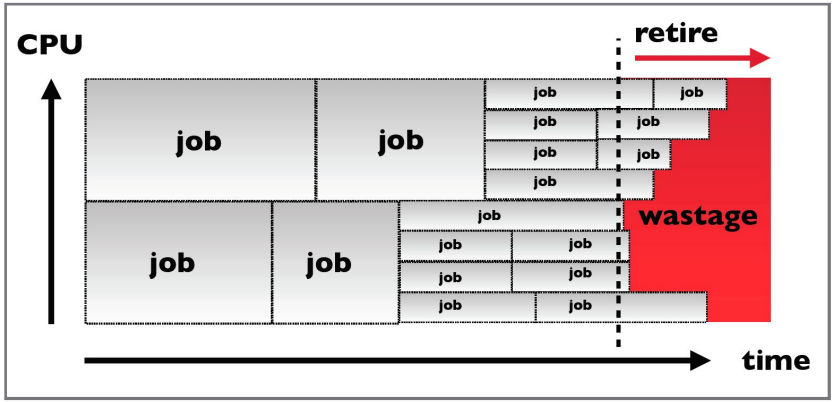


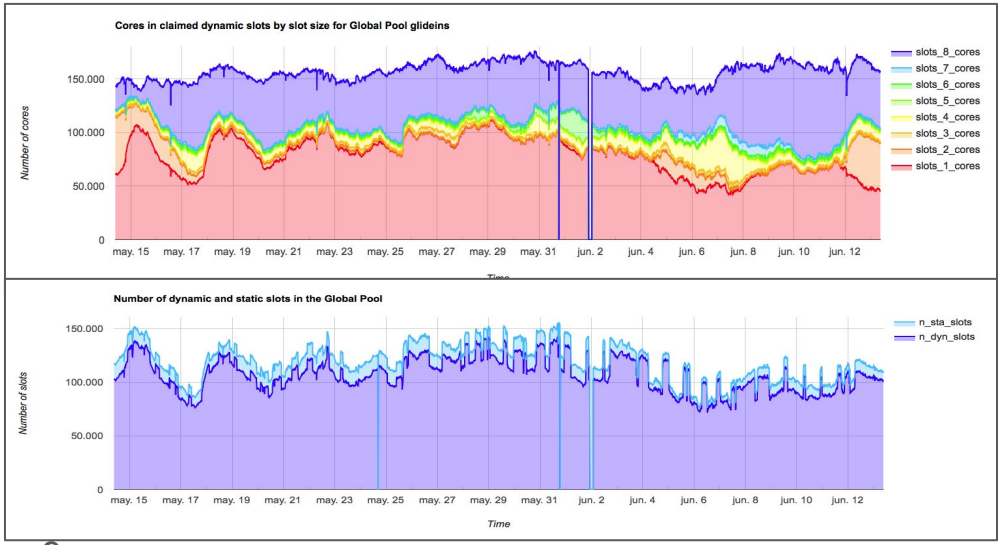

**Figure 3**: CMS estimated CPU (3a) and disk space (3b) resources required into the HL-LHC era, using the current computing model with parameters projected out for the next 12 years.

- Plot of CMS HL-HLC processing and data scales, from HSF.
- Factor of **~20x jump in scale (jobs and cores) around 2025**.
- Expect the **resource landscape to accelerate move to HPC & Cloud sooner**.

# Multi-core pool fragmentation

- The Global Pool is a multi-core pool with dynamic provisioning of slots (CPU, Memory, Disk)
- **Partitionable slots** can become fragmented into lower core count **dynamic slots**
- **Pool fragmentation** oscillates according to the composition of the workload mix
  - Mainly single core and 8-core requests
  - Also, CMS jobs can be **resized** at matchmaking stage
- Even for a pool of a stable size, the **number of dynamic slots** can greatly oscillate
  - **Main scaling at the Collector/Negotiator**
    - E.g. last month av**: 110k dynamic slots**
    - **Peaks at 150k (**“single core storms”**)**



# Scale Tests (I)

**CMS SI** and **OSG** have studied mainly **scalability and stability of a single HTCondor pool** provisioned via **glideinWMS**:

- I/O between and within HTCondor pool components
- Combinatorics at Negotiator matchmaking
- Speed and RAM usage of individual components
- Ability to scale horizontally (schedulers)

We'll continue exploring limits for a **single pool, pushing dynamic slot limits to higher scales** (>200k, up to 500k?).

- Multi-core workflows provide some headroom
- But "**single-core storms**" may expose current setup vulnerabilities.

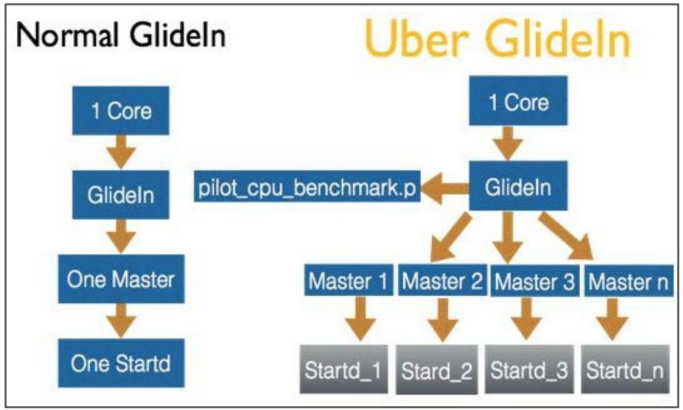
And we are **also interested in**:

- Scalability of **federated HTCondor pools**:
  - How many different Negotiators (pools) can schedulers actually talk to?
- Explore max schedd **submission rate (**1 Hz, 10 Hz, 50Hz?**)**
  - Need sufficient **submit pressure** to fill continuously growing resources
- Study the new **multi-threaded Negotiator**
- Running **without schedd claim leftovers** (for improved efficiency and prioritization)

# Scale Tests (II)

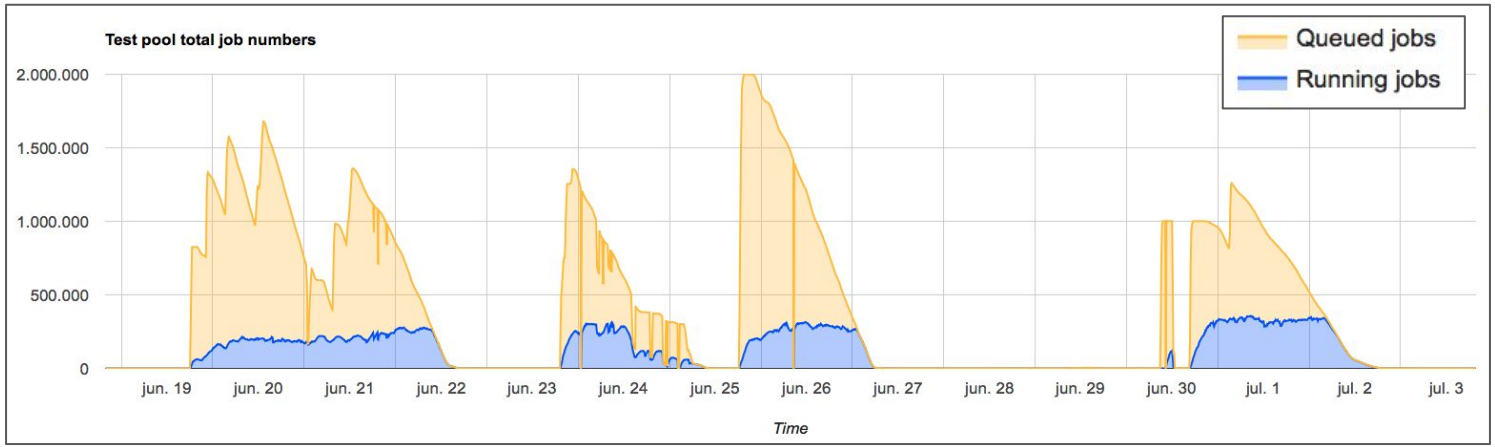
- **Scaling tests capability** deployed at CERN:
  - **glideinWMS Front-End** (24 cores, 46 GB, same as production), running with glideinWMS 3.2.21 and HTCondor 8.6.10, plus a dedicated **Factory** at CERN
  - **Central Manager** comparable to that of the Global Pool (40 CPU cores and 120 GB), HTCondor 8.7.8, plus **CCB**
  - 10 **Schedds** (32 cores, 60 GB, highIO disks), as used in production, HTCondor 8.6.11
- Using **über-glideins** i.e. run **32 startd's on one batch slot**
  - E.g. allows creation of **500K core pool** on only 16K real batch cores

- **Sleep jobs** (plus 1 CPU consumer) injected with a random submit script to maximize the **job diversity** in the system.
  - JDL parameters chosen by:
    - Ncores: **flat** (n_cores=randint(1, 8)), **realistic**, fully **single-core**, fully **8-core**
    - Random memory/core and disk/core
    - Walltime, also random, e.g flat in 480+/-60 min range
    - Selected_sites: targeting a variable number and mix of T1s+T2s



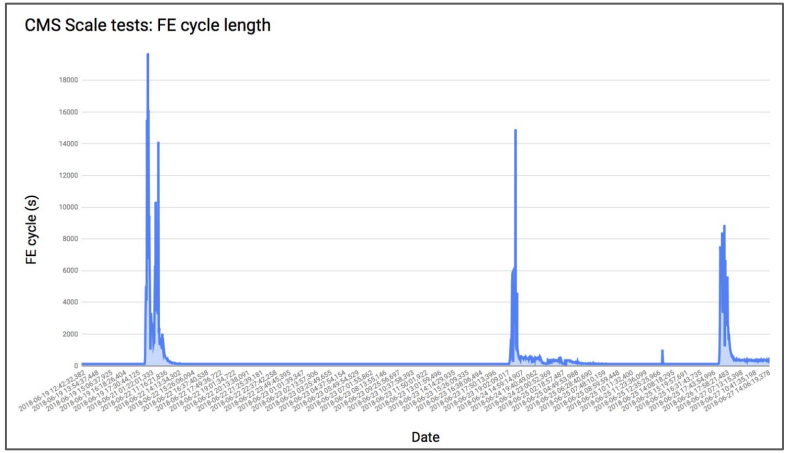
# Scale test rounds

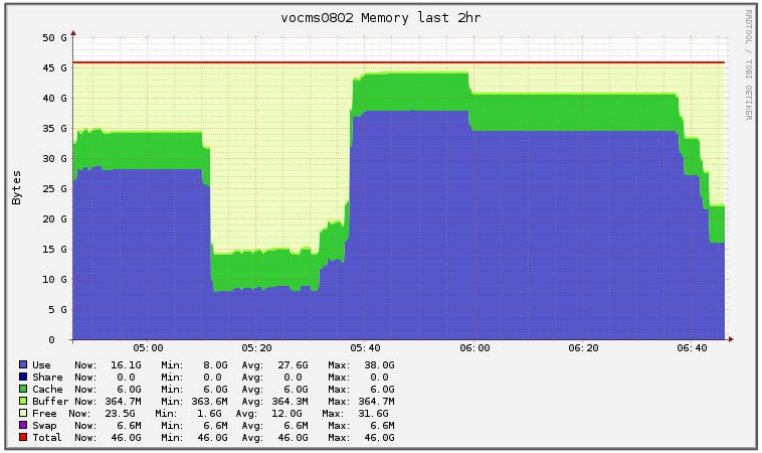
- Consecutive rounds increasing pressure and targeting maximum pool size
  - In total injected up to **2M (8h) jobs** in the system
  - **All jobs injected as single-core, to maximize pool fragmentation and stress on Collector/Negotiator**
- **Rounds 1 and 2**: limitations per schedd detected at 50k jobs
  - Could not grow beyond 300k slots due to **saturated submit capacity** (6x50k)
- **Round 3 and 4**: grew up to **10 schedds**: Max scale achieved at **350k running jobs** (or dynamic slots)
  - **About ~x3 size of production pool**, **equivalent to ~600k core pool at present n_core/job average**
  - **Limitations observed at FE, Collector and Negotiators** with present setup and software!



# GWMS Front-End scalability

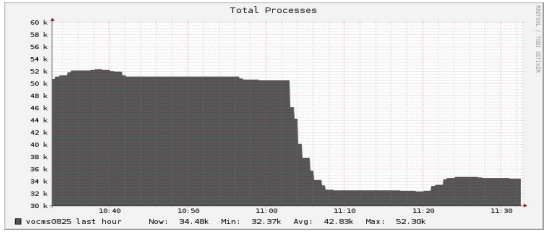
- **Observed very slow FE matchmaking cycle:** running for **more than 2h** while cycle typically at 15 mins in the global pool!
  Two main issues:
  - **Inefficient queries**, schedds slow in reporting their queue content
  - **Combinatorics problem, up to 2M jobs** in the schedd queues
- **Increase parallelism**: FE could be configured to deal with multiple workers in parallel (tested up to 8)
  - **Memory limitations**! At these scales, each worker adding about **4 GB**
- Ongoing Work:
  - Increased logging verbosity and added new print-outs to the code to **understand the cycle in detail**
  - **We might need a bigger host (the size of the pool Central Manager)**

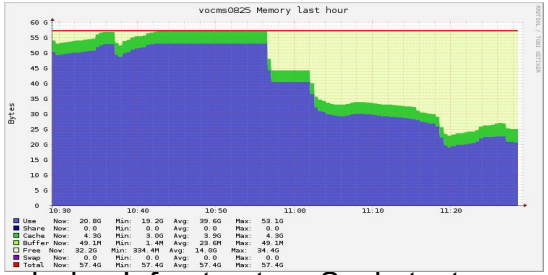


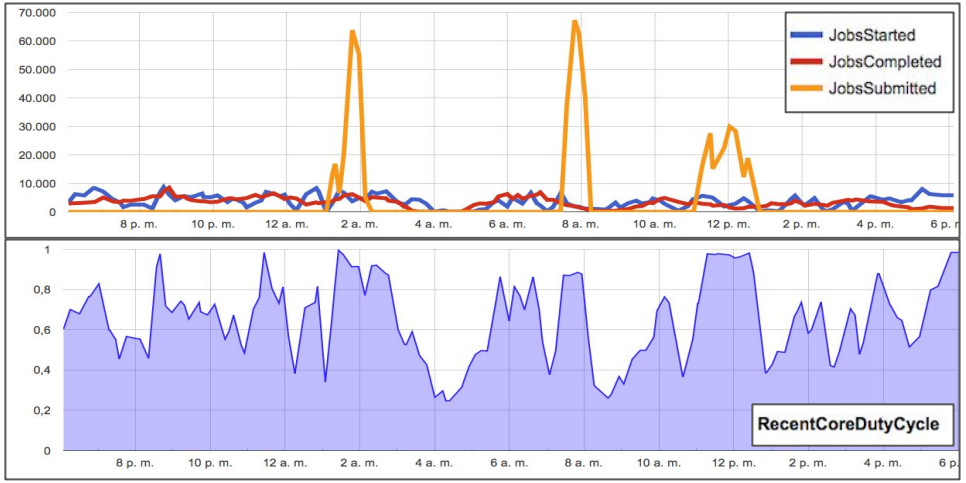



# Schedd Limits

- Exploring performance limitations of our schedds, running at hosts identical to production ones
- Q: How many jobs can our schedds handle at once, **running and queued**?
  - **A**: Our current **schedds can run up to about 50k jobs (plus 150k jobs in queue)**.
  - Memory increases at about **~1 MB/shadow process** (32-bit binary version)
- Q: What are the **maximum sustainable job submit/start/complete rates**?
  - **A**: RecentJobsStarted metrics per schedd indicate 10k jobs started in 20 min seem feasible => **O(10) Hz**

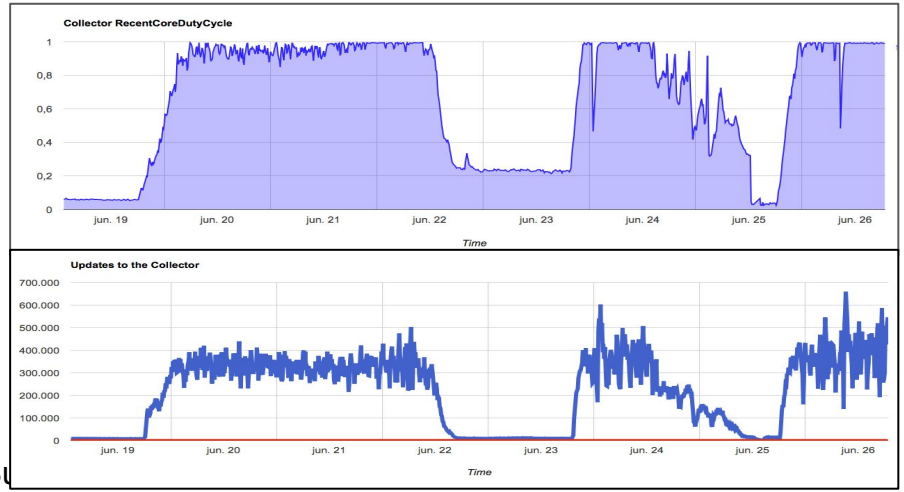


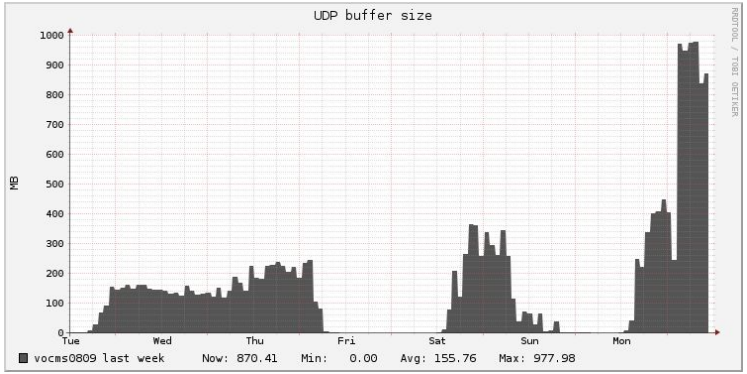



# Collector Scalability

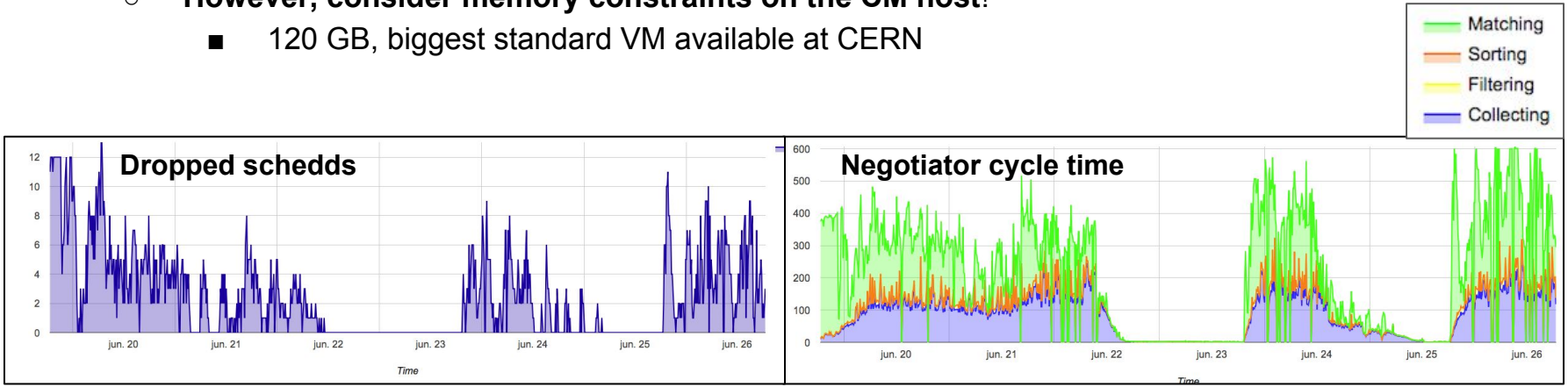
- Many elements in the infrastructure **rely on the capacity of the collector to fast and efficiently collect and distribute information on workload and allocated resources**
  - Negotiators (of course)
  - But also, GlideinWMS Front-End, JobRouters, monitoring tools, etc
- **Collector fully stressed**!
  - **Duty cycle at 0.995**: receiving slot updates reaching above 400k / 20min accounting interval
  - **UDP buffer saturation,** even after successive increments (128 to 192 to 256 to 512 MB each for w/r)
  - **Memory spikes and OOM killing collectors** (once every hour in the last round)
    - Main collector thread being killed at **20 GB**



# Negotiator Scalability

- **Negotiator**: Pool runs with 3 negotiator instances (as in the Global Pool)
  - No real scalability issue observed for the Negotiators, although cycle many times reaching target time limit at **600s**, for efficient use of the slots
- **Cycle time** about equally split on **collecting slot updates** and **matchmaking** phases
  - Production pool dominated by matchmaking
  - Matchmaking driven by **loaded schedds**, which frequently time-out of the cycle (dropped with 60s timeout per schedd at matchmaking)
- Speeding up matchmaking could benefit from running **forked negotiators**
  - **However, consider memory constraints on the CM host**!
    - 120 GB, biggest standard VM available at CERN



# Conclusions

- We are analyzing our current SI to **detect bottlenecks preventively**
- The **resource landscape is changing rapidly** as we move to Grid + HPC + Cloud. During 2018 we will be studying the **scalability of federated pools**, in addition to **maximum pool size** and **schedd properties**
- Longer term, we will be **looking to HL-LHC scales** and the sustainability of **HTCondor** and **GlideinWMS** to serve our workload management needs in the future.
- In our recent scale test rounds, pushed the size of our test Global Pool to about **350K dynamic slots** with a configuration that **maximizes pool fragmentation and matchmaking**
- Explored (and found) **scaling limitations on key components**: GlideinWMS FE and HTCondor Schedds and Collector
- Reached about a **factor 3** in size with respect to our production pool (equivalent to **600k core**), so we are **not worried about the immediate scale** limits, but must start planning **next steps**
- We wish to **thank** the development communities of **HTCondor** and **GlideinWMS** for their continued support and close collaboration with CMS.

# Additional slides

# Abstract:

The CMS Submission Infrastructure Global Pool, built on GlideinWMS and HTCondor, is a worldwide distributed dynamic pool responsible for the allocation of resources for all CMS computing workloads. Matching the continuously increasing demand for computing resources by CMS requires the anticipated assessment of its scalability limitations. Extrapolating historical usage trends, by LHC Run III the CMS Global Pool must be able to manage stably and efficiently 0.5M CPU cores, about a factor 2 from current size. In addition, the Global Pool must be able to expand in a more heterogeneous environment, in terms of resource provisioning (combining Grid, HPC and Cloud) and workload submission. A dedicated testbed has been set up to simulate such conditions with the purpose of finding potential bottlenecks in the software or its configuration. This contribution will provide a thorough description of the various scalability dimensions in size and complexity that are being explored for the future Global Pool, along with the analysis and solutions to the limitations proposed with the support of the GlideinWMS and HTCondor developer teams.

# CMS SI Group Charge

- The charge of the **Submission Infrastructure Group with the CMS experiment** at CERN is to:
  - **Organize** glideinWMS and HTCondor pool **operations** in CMS, in particular of the Global and CERN HTCondor Pools
  - **Communicate** CMS **priorities** to the development teams of glideinWMS and HTCondor
- SI activities broadly fall into several categories:
  - Overcoming **current operational limitations** or problems
  - Preparing for **future scales** or feature requirements (i.e. next year's problems and longer-terms problems)
  - Integration of new, diverse **resource types** and **job submission methods**
- We regularly contribute to the HTCondor Workshops in the U.S. and Europe as well as to international conferences (such as CHEP!)

# Related talks to CMS SI at CHEP2018

- **Improving efficiency of analysis jobs in CMS**
- **Improving the Scheduling Efficiency of a Global Multi-core HTCondor Pool in CMS**
- **Producing Madgraph5_aMC@NLO gridpacks and using TensorFlow GPU resources in the CMS HTCondor Global Pool**
- **Recent developments in glideinWMS: minimizing resource wastages**

# Motivation: HL-LHC Challenge (II)

We expect a factor 20 increase in the numbers of jobs and cores:

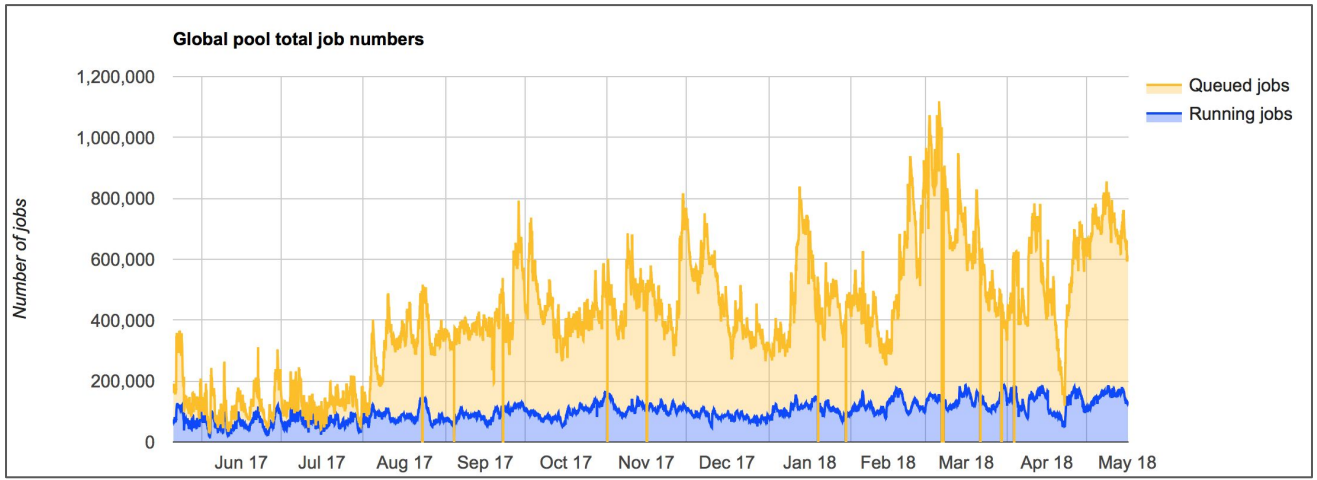
- Only some evolution in parallelization expected. Current mix is in the 1 to 8-threaded jobs range. Perhaps a factor of 2-4 more only.
- Still, expect a significant fraction of single-core workflows foreseen.

Resource allocation challenge:

- The pilot model works best when all of your resources are more or less the same: Grid resources based on similar architectures and OS, sitting behind a small number of different Computing Elements (i.e. HTCondor-CE, ARC, CREAM).
- The resource landscape is expected to change (in fact it is already changing …)

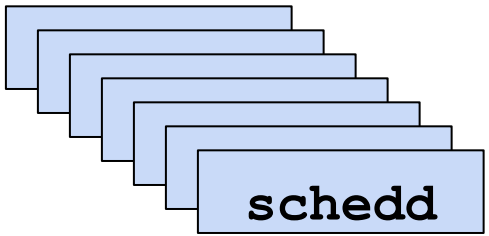
# Demand for Resources

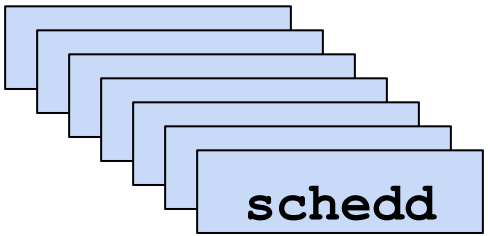
- Demand for resources is also constantly increasing, and increasing in core count as well.
  - There are seasonal peaks and random noise.
- Challenge of SI is to provision resources **stably** and **efficiently** in whatever demand environment is facing us.



# Schedulers

- CMS Production and Analysis workflows are submitted to HTCondor schedd's at CERN and Fermilab.
- Job submission can **scale horizontally** - currently ~5-10 active production and ~15 active analysis schedd's.
- Going to O(100) schedulers on stronger hardware in the HL-LHC era should handle the load? To be studied in our 2018 scale tests. Are there limitations?



# Schedulers in the global pool

- Current metrics from one of our current production schedds
  - An example of a heavily loaded schedd, getting dropped from the negotiation cycle