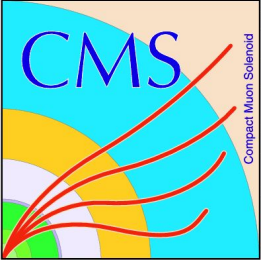


Pushing HTCondor boundaries: the CMS Global Pool experience

Antonio Pérez-Calero, James Letts & Diego Davila
on behalf of the CMS Submission Infrastructure

European HTCondor Workshop
September 6th 2018

Outline

- The CMS Submission Infrastructure (SI) Global Pool
- Motivation for exploring HTCondor scaling boundaries
- Dedicated scale tests: setup and results
- Recent storms in production environment
- Conclusions

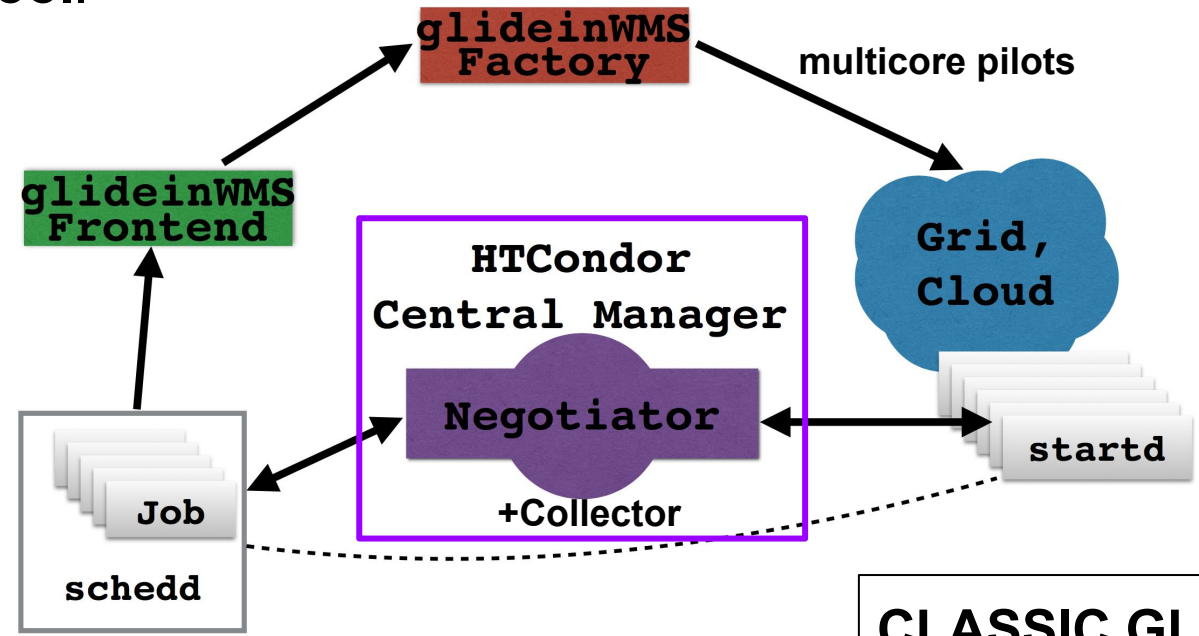
CMS SI Group Charge

- The charge of the **Submission Infrastructure Group with the CMS experiment** at CERN is to:
  - **Organize** glideinWMS and **HTCondor pool** operations in CMS
  - **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**

- In terms of **communication**, we regularly hold **meetings** with HTCondor developers team and other HTCondor interested parties, and usually **contribute** to the HTCondor Workshops in the U.S. and Europe as well as to international conferences (such as CHEP!)

CMS Global Pool

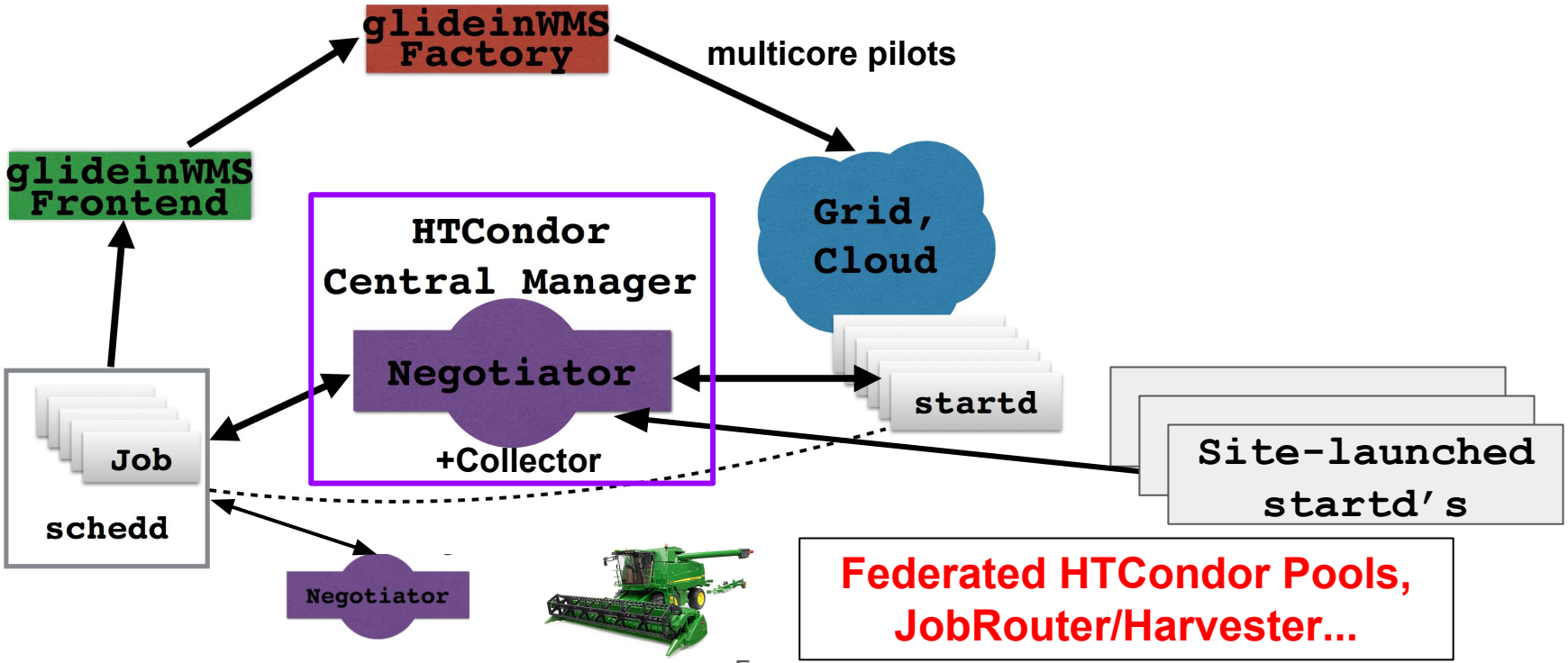
The CMS Global Pool is both a **glideinWMS instance** and a **HTCondor pool**.



CLASSIC GLOBAL POOL VIEW

CMS Global Pool

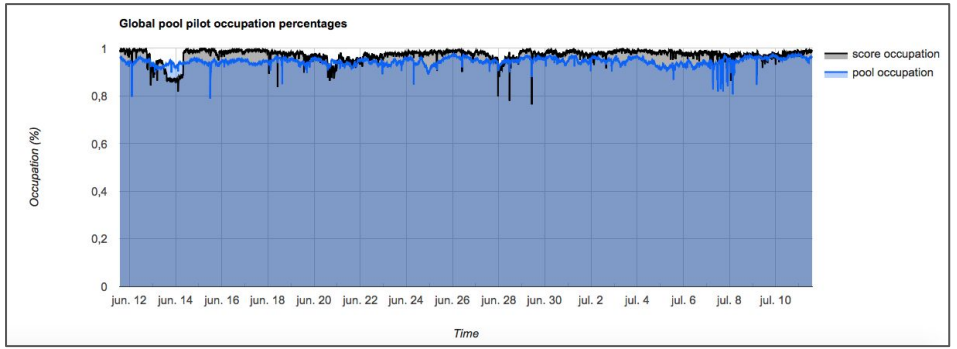
Already the resource and submission landscape is shifting …

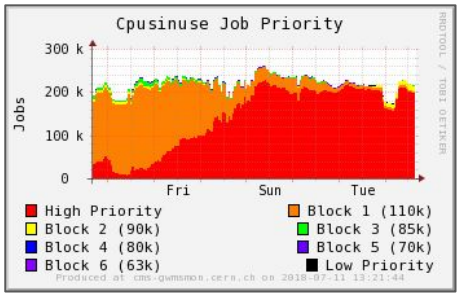


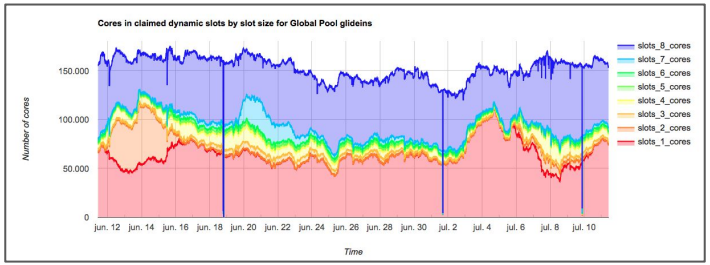
CMS Global Pool

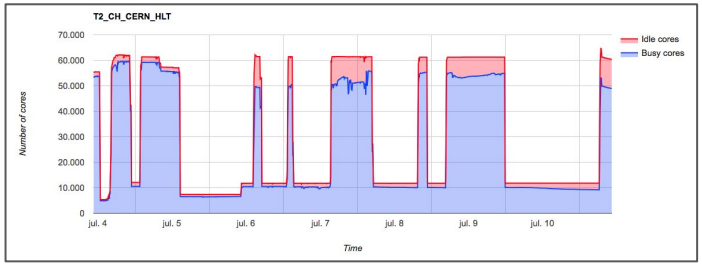
The CMS HTCondor Global Pool, designed and implemented for LHC Run-2, **is a very successful infrastructure covering CMS needs**, capable of:

- 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, tier0)
- Integration of dynamic resources (e.g. HLT)



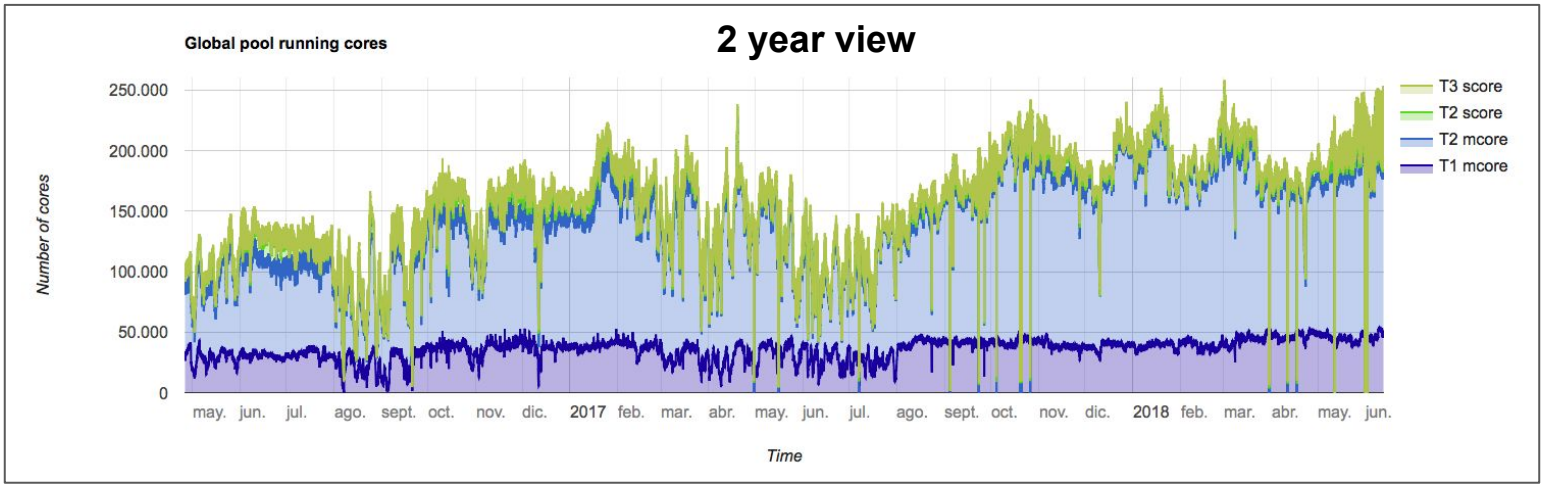






Motivation (I): Increasing Scales

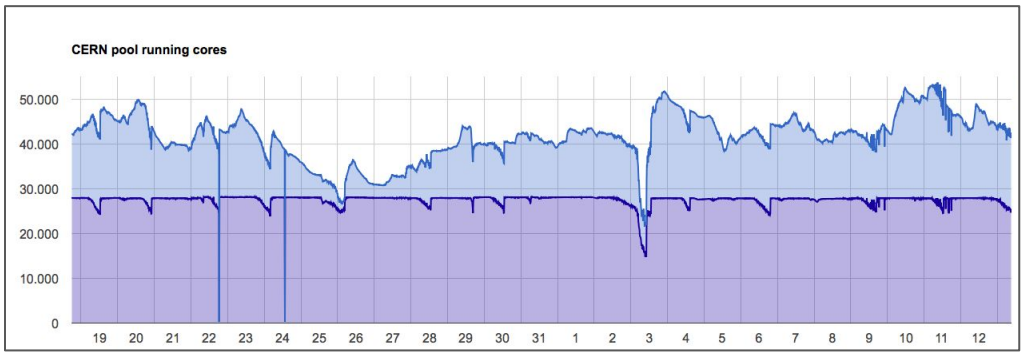
- **Size of the Global Pool** driven both by
  - resource **deployments** (tend to happen later in the calendar year)
  - resource **requests** (e.g. compare quiet 2016 and 2017 Summer months with now)
- During Run 2, **HLT farm** commissioned as opportunistic resource (peaks of **60k cores** in 2018)
- 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**

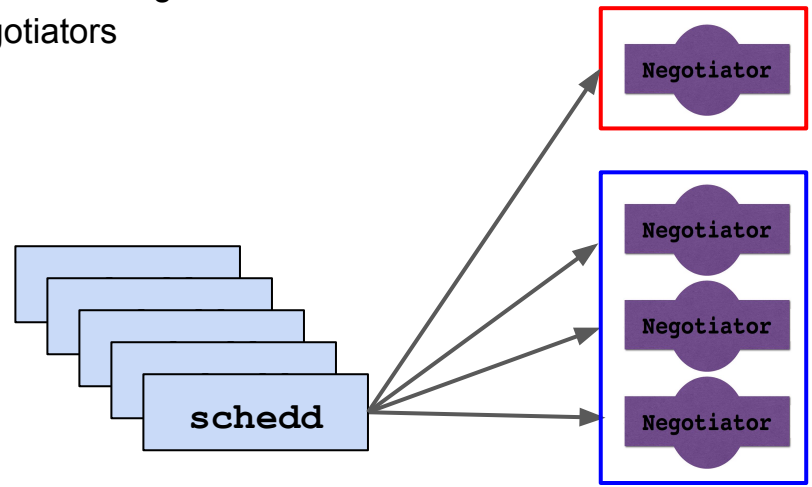


Motivation (II): Multiple Pools

- **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 national clouds** (many?) 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

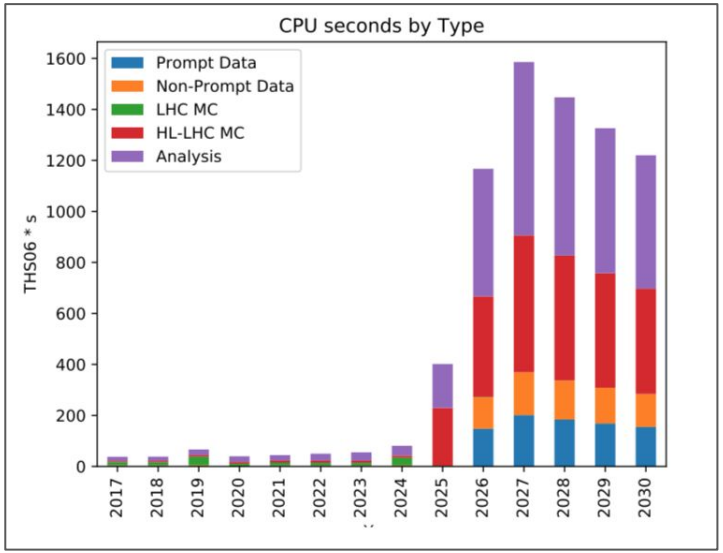
~50k-core new pool for CERN resources





Motivation (III): HL-LHC Challenge

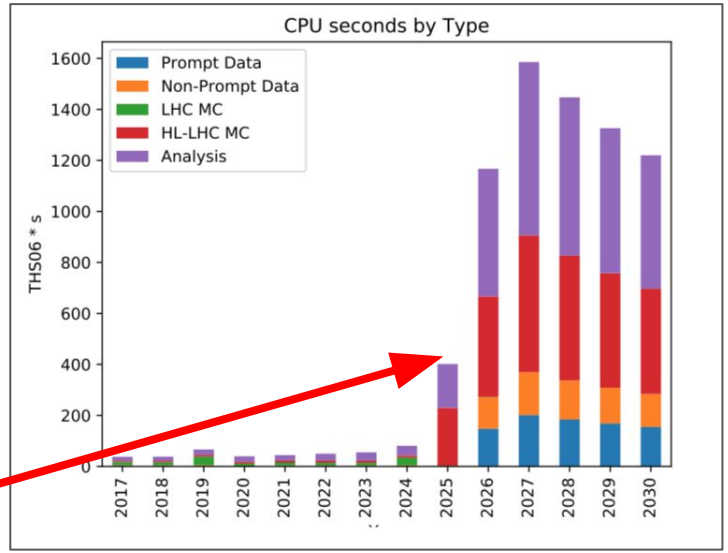
- Consider CMS processing needs projected for HL-LHC era (HSF).
- Factor of **~20x jump in scale (jobs and cores) around 2025**
  - Limited progress in further parallelization expected: current mix is in the 1 to 8-threads per job range, perhaps a **factor of 2-4 more only**.
  - Still, a significant fraction of workflows foreseen to **remain single-core**
- **How many schedulers** will we need to handle such load?



- Expect the **resource landscape to accelerate move to mixed model including**
  - **Grid resources:** based on similar architectures and OS, sitting behind a small number of different Computing Elements (i.e. HTCondor-CE, ARC, CREAM).
  - **HPC & Cloud:** actually already part of our infrastructure, at higher scales soon

Motivation (III): HL-LHC Challenge

- Consider CMS processing needs projected for HL-LHC era (HSF).
- Factor of **~20x jump in scale (jobs and cores) around 2025**
  - Limited progress in further parallelization



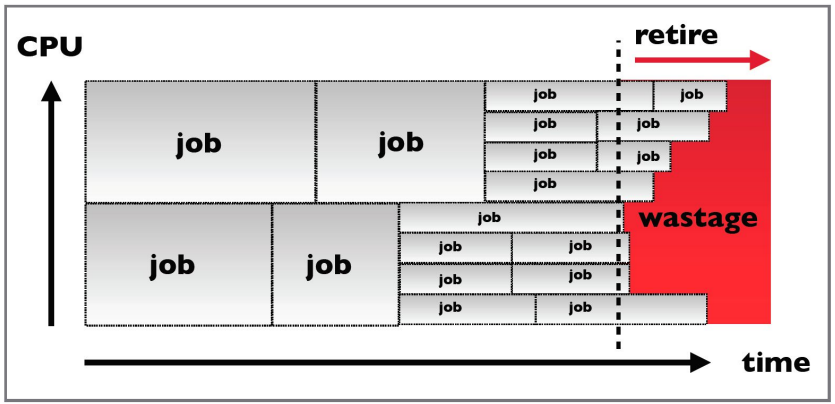
WHY TODD's CHALLENGE AREA #3 is KEY FOR CMS!!

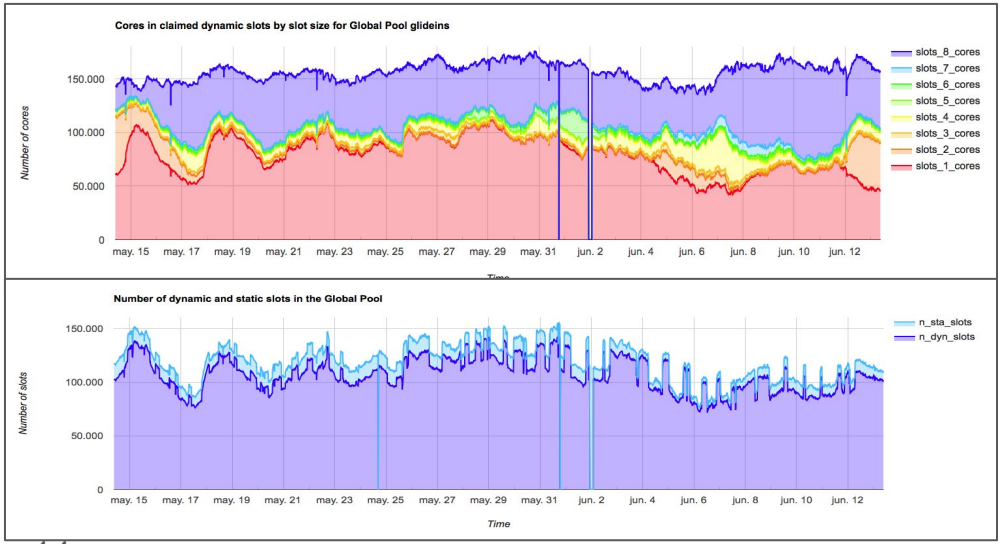
to mixed model including

- **Grid resources:** based on similar architectures and OS, sitting behind a small number of different Computing Elements (i.e. HTCondor-CE, ARC, CREAM).
- **HPC & Cloud:** actually already part of our infrastructure, at higher scales soon

Multi-core pool fragmentation

- The Global Pool is a multi-core slots pool with dynamic partitioning of resources (CPU, Memory, Disk)
- **Partitionable slots** can become fragmented into lower core count **dynamic slots,**
- **Pool fragmentation** oscillates according to the composition of the CMS workload mix
  - Mainly single core and 8-core requests, with exceptions
  - Also, CMS jobs can be **resized** at matchmaking stage (e.g. desire 8, accept 3 to 15)
- Even for a pool of a stable size, the **number of dynamic slots** can greatly oscillate
  - **Main scaling driver of the Collector/Negotiator**
    - E.g. last month av**: 110k dynamic slots**
    - **Peaks at 150k (**“single core storms”**)**





Scale Tests (I)

Considering our continuous growth over the last years, CMS SI (and **OSG**) have been regularly assessing mainly **scalability and stability of a single HTCondor pool**:

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.

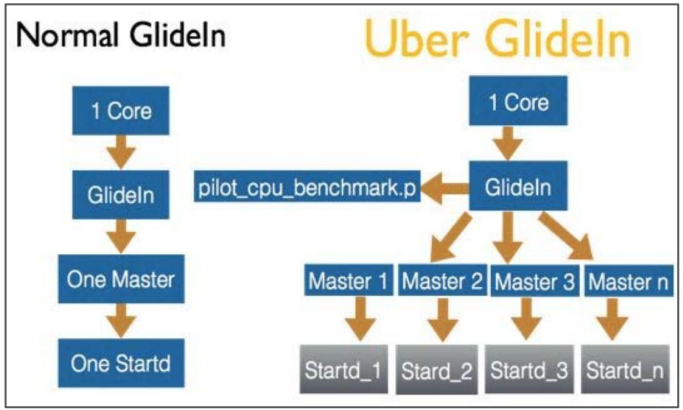
And now we are **also interested in**:

- 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**
- ...while retaining high efficiency usage and reliably enforcing CMS WF prioritization policies

Scale Tests (II)

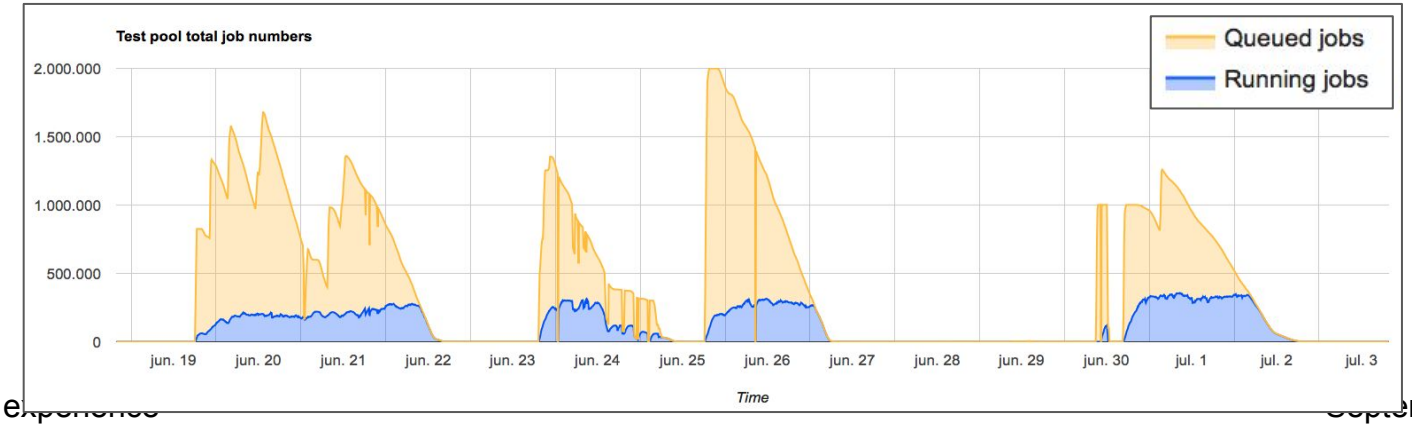
- **Scaling tests capability** deployed at CERN:
  - **Central Manager** comparable to that of the Global Pool (40 CPU cores and 120 GB), HTCondor 8.7.8, plus **CCB**
  - 10 **Schedds** (32 cores, 60 GB, highIO disks), as used in production, HTCondor 8.6.11
- Using **über-glideins** i.e. run **32 startd's 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+T3s



Scale test rounds

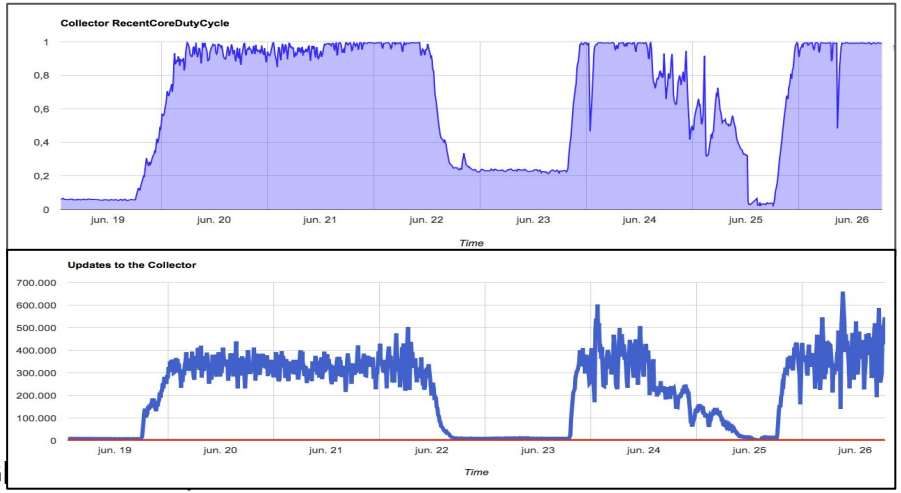
- Consecutive rounds increasing pressure and targeting maximum pool size
  - Injected up to **2M jobs** in the system
  - **All jobs injected as single-core, to maximize pool fragmentation and stress on Collector+Negotiator**
  - **Tested scaling dependency with average job length: 1h to 4h in tests 1 to 3, 8h jobs in the last iteration (CMS target)**
    - longer jobs: less load on the schedds, less frequent slot updates to the collector, shorter nego cycles...
- **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 dynamic slots**
  - **About ~x3 size of production pool**, **equivalent to ~600k core pool at present n_core/job average**
  - **Limitations observed at Collector and Negotiators (**with current setup and software)

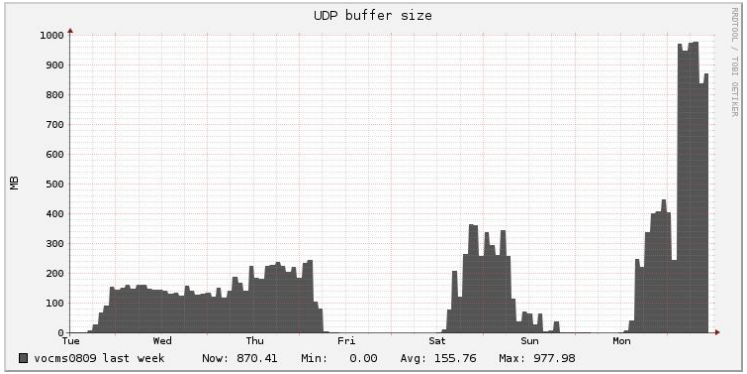


Collector Scalability

Collector fully stressed!

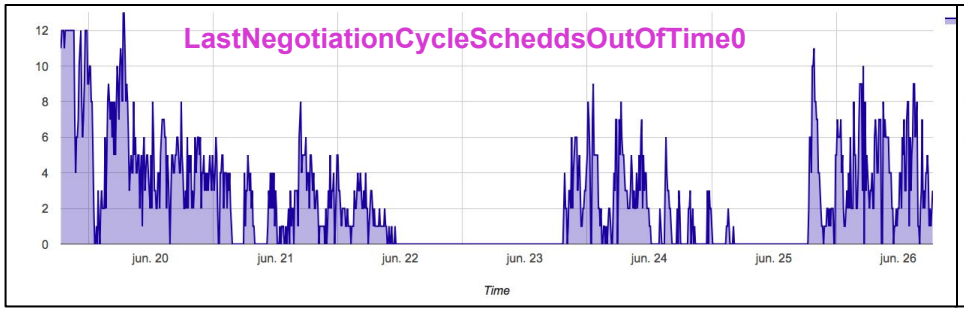
- **Duty cycle at 0.995**: receiving slot updates reaching above **400k / 20min** accounting interval
- **Children (110) to top collector UDP buffer saturation,** even after successive increments (128 to 192 to 256 to 512 MB each for w/r buffers)
- **CM host depleted of memory: OOM killing collectors** once every hour in the last round
  - Main collector thread being killed at **20 GB**

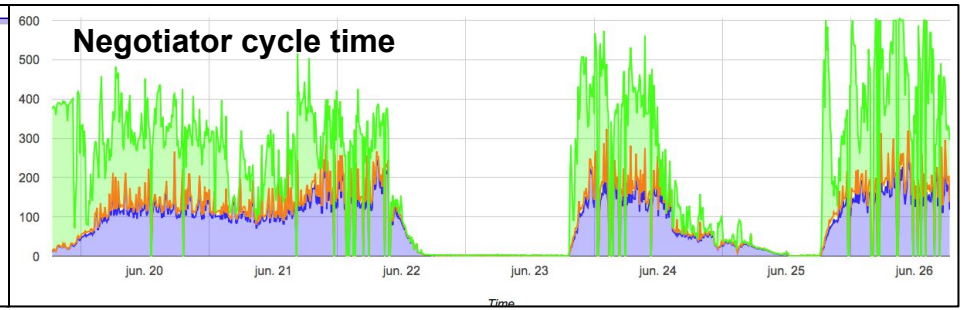




Negotiator Scalability

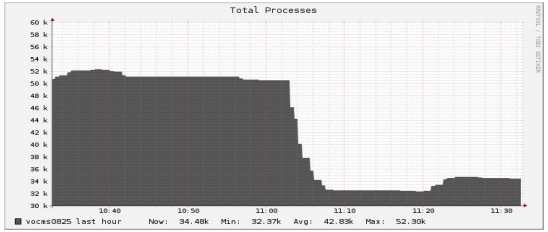
- **Negotiators**: test pool runs 3 negotiator instances (as in the Global Pool)
  - No real scalability issue observed for the Negotiators, although cycle many times reaching target time limit at **600s**, for efficient use of the slots
- **Cycle time** about equally split on **collecting slot updates** and **matchmaking** phases
  - Production pool dominated by matchmaking
  - Matchmaking time driven by **slow/loaded schedds**, which frequently are dropped of the cycle (after 60s timeout per schedd vs negotiator 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

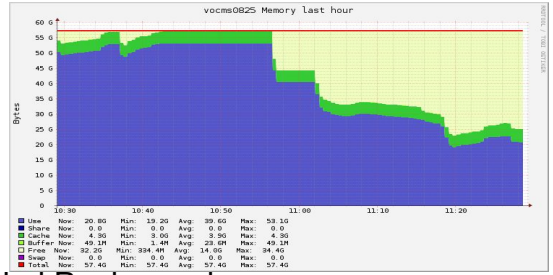


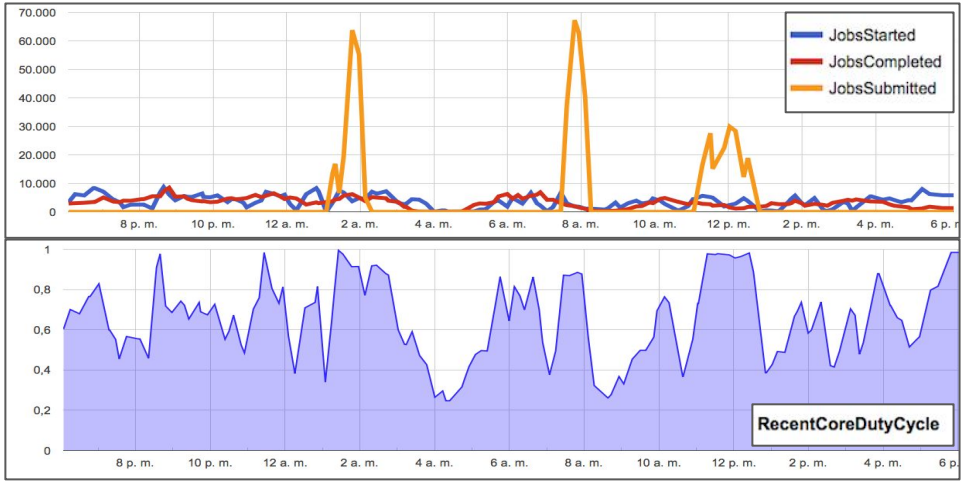


Schedd limits

- Exploring performance limitations of our schedds
- 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 **O(10) Hz** should be feasible







Revisiting schedd limits

- **Caveats**: In the tests we ran, we considered submission, start and completion rate limitations in the assumption of basically **null-sized input and output sandboxes**
  - For a more **realistic approach**, we should explore schedd limitations including now **non-zero sandboxes**
  - E.g. measured from O(kB) to O(100 MB) input in the analysis schedds
- **Also** schedd hosts run other services (WMAgent, JobRouter), so memory available for the schedd is actually tighter than in the test pool
  - Hence actually support lower max run/idle jobs

Adding a fourth negotiator in the pool

Why 3 negotiators:

- **Scalability** (divide and conquer)
  - Each group roughly equal in number of cores: **Tier-1s; Tier-2_US; Tier-2 general+ Tier-3I**
- **Resource-based fair share**: different policies in different Negotiators
  - “analysis” and “production” represent the two major workload categories CMS has to satisfy
  - CPU quotas shares defined by CMS as policy
    - The policy goal for Tier-1 sites is 95% production, 5% analysis
    - The policy goal for Tier-2 sites is 75% production, 25% analysis
  - Implemented as separate accounting groups, which allows us to set dynamic quotas for each of them in each resource group
    - set in the configuration of each negotiator
  - CMS would ideally like the **policy to be enforced per site**: e.g. 25% of each Tier2 is available for analysis tasks
    - however, the technology (at least as we are using it in our implementation) only allows settings **per resource group**, e.g. all Tier-1 sites

Adding a fourth negotiator in the pool

The **problem** appeared in the **general T2+T3 group** of sites, managed by a single negotiator

- A very **diverse mix** of about 70 different T2 + T3 sites
- **Production** tasks **targets only certain sites** (good site readiness, highIO available storage,...):
  - Production dominates the sites it wants to use
    - **analysis share** at 10 “popular” EU T2 sites **basically zero**
  - Analysis ends up spending its quota only at the sites production doesn’t want

IDEA to solve this: Introduce a **4th negotiator** to further break up the third group in two: **main T2_EU; general T2s + T3s**

-> Naively expected to simply work

Adding a fourth negotiator in the pool

Anyway, decided to test before deployment…

- **Realistic approach** in the test
  - Reconfigured test pool with 4 negotiators
  - Limited load to max 250k running jobs
  - Job length at 4h
  - **Max collector workers for high-prio increased from 3 to 4** (max total reduced from 8 to 6)

- **Results**: Hit the limits on the collector side, similarly to previous scaling tests:
  - again, **saturating collector duty cycle** (0.995) and **udp buffers** (512 MB)
  - CM host running close to **memory starvation**, due to the inclusion of the **4th high prio collector worker**
  - Ultimately, max scale achieved at **180k dynamic slots**
- Limitations too close to the scale we currently run at: **we can't afford it :(**

Recent storms

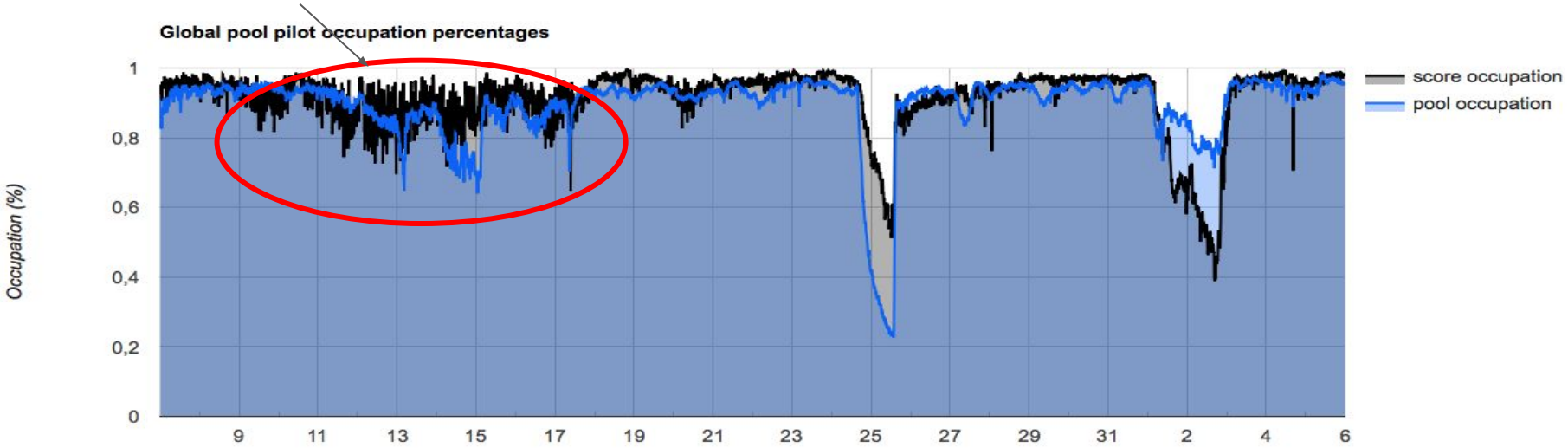
- Recent storms in the production system reveal potential weaknesses, aka what can go wrong in a perfectly working system when some parameters deviate from the usual and ideal values
  - Already mentioned **single-core job storms**, fragmenting the pool and affecting collector and negotiators

Recent storms

- Recent storms in the production system reveal potential weaknesses, aka what can go wrong in a perfectly working system when some parameters deviate from the usual and ideal values

Short jobs storm!

O(10min) instead of hours long jobs. Active schedds maxed at ~11Hz, can't keep up with the job submit/start/complete rates needed to keep full pool resources busy,



Recent storms

- Recent storms in the production system reveal potential weaknesses, aka what can go wrong in a perfectly working system when some parameters deviate from the usual and ideal values

Autoclusters explosion!

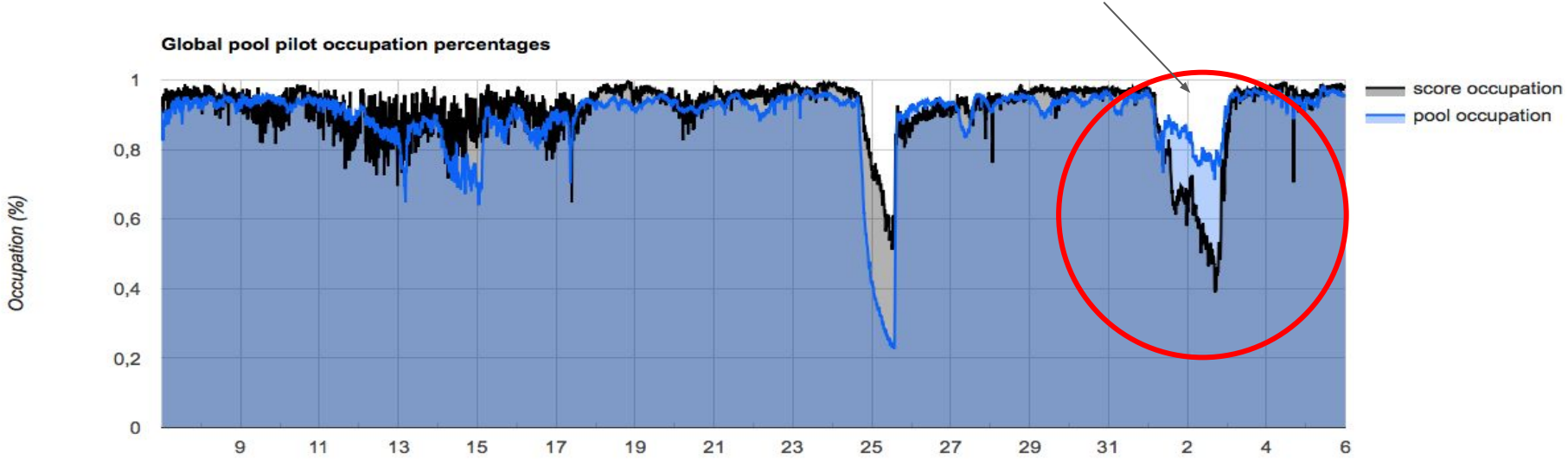
JobID inserted in job autoclustering significant attributes by mistake: one autocluster per job!

Kills schedds and negotiator: Even if CURB_MATCHMAKING = true, the schedd still builds the full RRL for the negotiator, with the negotiator waiting during that time

Recent storms

- Recent storms in the production system reveal potential weaknesses, aka what can go wrong in a perfectly working system when some parameters deviate from the usual and ideal values

CRAB schedds expired network rules!
With puppet disabled, relevant **ports closed** for outside communication from CERN
Schedds still negotiate and claim slots, but can't push jobs into the startd's!!



Conclusions

- We are analyzing our current (and very successful) HTCondor infrastructure to **detect bottlenecks preventively and assess scalability into CMS projected needs**
- Pushed the size of our test pool to about **350K dynamic slots** with a configuration that **maximizes pool fragmentation**
- Explored (and found) **scaling limitations on key components**: Schedds and Collector
  - But at about a **factor 3** in size with respect to our production pool **(~600k cores**), so we are **not worried about the immediate scale** limits,
  - ...but must start planning **next steps,**
- **which includes further scale testing, with refined model and more detailed monitoring**

We wish to **thank** the HTCondor developers community for their **continued support** and **close collaboration** with CMS

Additional slides

Abstract:

In recent times, the CMS HTCondor Global Pool, which unifies access and management to all CPU resources available to the experiment, has been growing in size and evolving in its complexity, as new resources and job submit nodes are being added to the design originally conceived to serve the collaboration during the LHC Run 2. Having achieved most of our milestones for this period, the pool performs efficiently according to our present needs. However, looking into the coming years, and particularly into the HL-LHC era, a number of challenges are being identified and preliminarily explored. In this contribution we will present our current Global Pool setup and operational experience and how it is expected to extrapolate to meet the near and long-term future challenges.