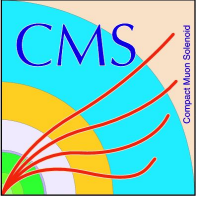


Reaching new peaks for the future of the CMS HTCondor Global Pool

A. Pérez-Calero Yzquierdo, M. Mascheroni, M. Acosta, J. Dost, S. Haleem, K. Hurtado, F. A. Khan, E. Kizinevic and N. Peregonov for the CMS collaboration

vCHEP 2021, May 19th April 2021

Outline

- The CMS Submission Infrastructure
- The Scalability challenge in the CMS Global Pool
- Addressing scalability limitations
- Conclusions

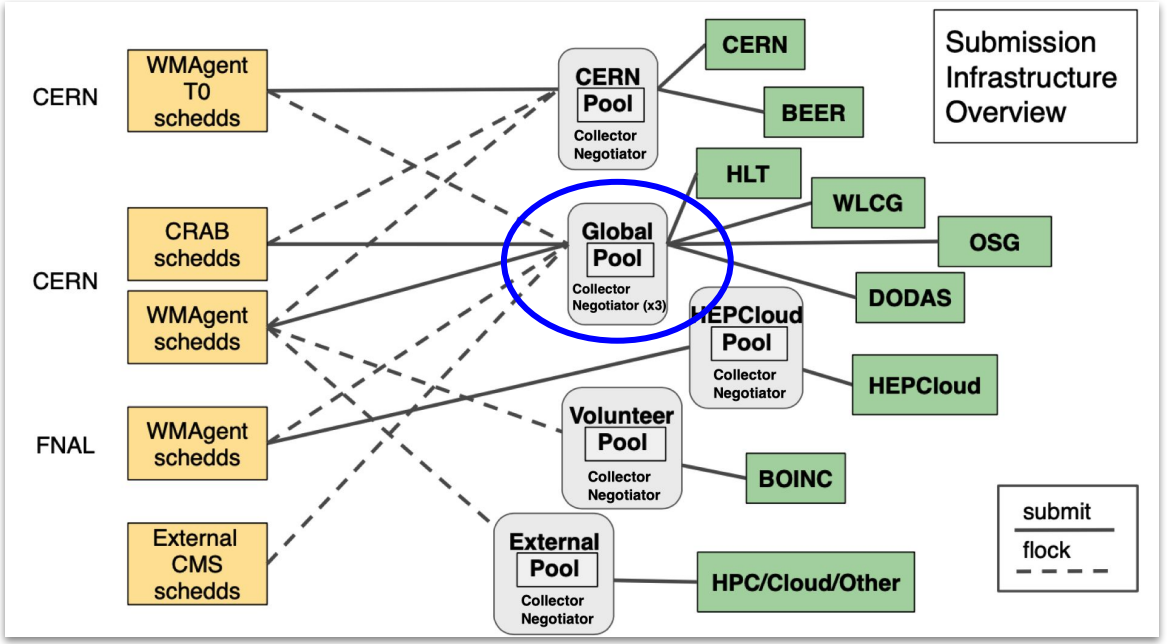
The CMS Submission Infrastructure

- **Mandate**: **Running the CMS distributed infrastructure** where all reconstruction, simulation, and analysis of physics data takes place!

- **In practice**:
  - **Operate a HTCondor pool** distributed over 70 Grid sites, plus non-Grid resources (HLT farm, HPC sites, Cloud)
  - Run the software and infrastructure required to build this pool on demand (GlideinWMS)
  - Communicate CMS priorities to the development teams of GlideinWMS and HTCondor in regularly held meetings where we discuss
    - Current operational limitations
    - Feature requests
    - **Future scale requirements**

A Complex Infrastructure

- **The CMS SI model evolved to use federated pools**, with extensive use of **flocking**
- Two sets of workflow managers: CRAB + WMAgent
- **The Global Pool is the biggest** and most important one:
  - **~300k CPU cores**
  - **100k to 150k running jobs**
  - 50+ schedds
  - 3 negotiators
- Redundant infrastructure for HA

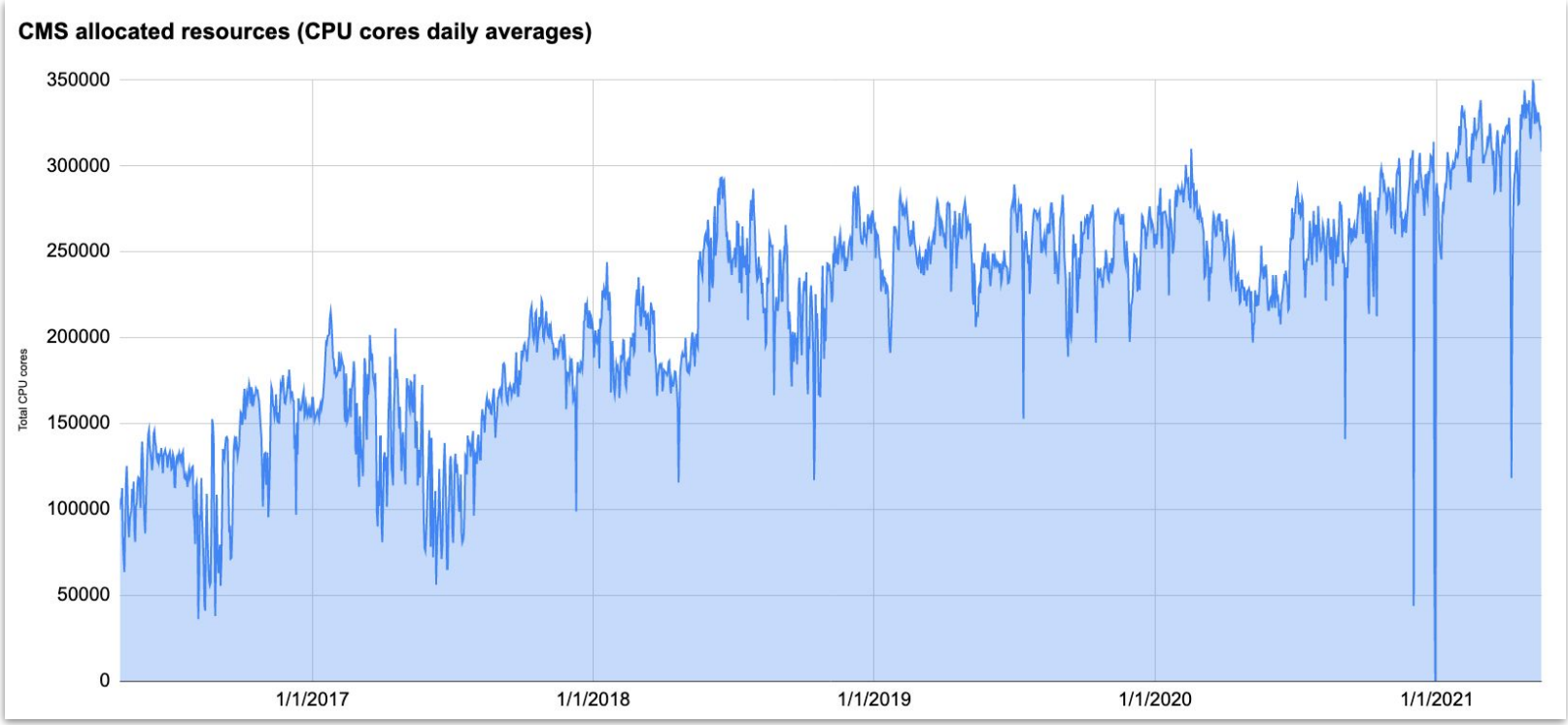


- Resources mainly acquired with GlideinWMS pilots
- Vacuum-like instantiated: slots (DODAS), BOINC(CMS@Home), opportunistic (HLT)...

- The Scalability challenge in the CMS Global Pool

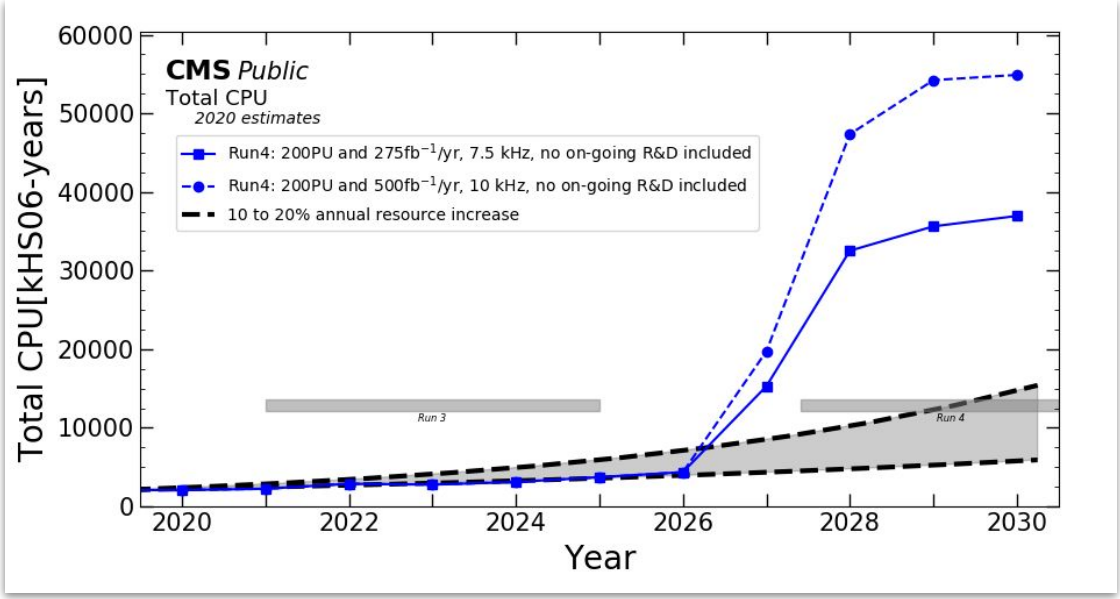
The scalability challenge (I)

- CMS pool of resources under **continuous growth: 3x in 5 years**



The scalability challenge (II)

- **Implications from HL-LHC challenge**



- Addressing scalability limitations

Pushing the limits of the Global Pool

- **We have been continuously detecting and solving bottlenecks to our Global Pool,** with the support of the HTCondor and GlideinWMS developers teams over the years

- Multiple **customized settings** (compared to a “standard HTCondor configuration”):
  - Using a **CCB** running on a separate host to the CM, with enlarged pool of connection sockets
  - Use of multiple **negotiator** daemons running in multithreaded mode
  - 32-bit binaries for the **shadow processes** running in the schedds
  - Hierarchy of **secondary collectors** connected to the main top collector process
  - Optimized **slot update conditions** (filter on update triggers, use UDP instead of TCP, enlarged UDP buffer)
  - **Classify queries** reaching the collector from the negotiator as high-prio, as opposed to those from the GlideinWMS FE and CMS WM and monitoring services
  - **Redirect non-high prio queries** to the slave secondary collector (HA infrastructure at FNAL)

- Most of these actions directed at **avoiding the saturation of the top collector**
  - **Essential to build a exhaustive and reliable monitoring infrastructure**

Scalability tests 2021

2021 Q1 tests (HTCondor 8.9.10**): Focus on the impact of the virtualization layer** on the collector performance:

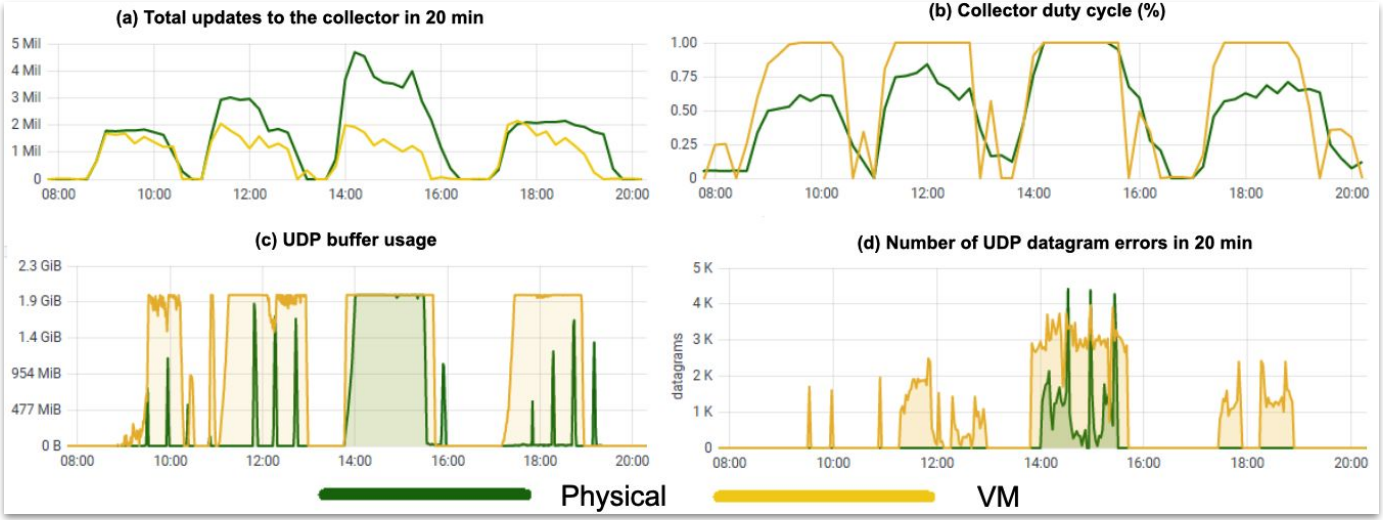
- Pool Central Manager (collector+negotiators) deployed in **physical (PM) and virtual (VM) machines, with identical configuration**

Run the following sets of tests

- **Direct stress tests on the collector**
- **Full infrastructure scalability test**

Condor_advertise tests: PM vs VM

- **Condor_advertise** commands launched to **create a stream of slot update messages (ads)** aimed at each of the two collectors.
  - Communication was performed via UDP packets with 2 GB buffer size
  - Gradually increasing their stress, with slot update rates from 60 to 180 ads/s.
- **Performance of the PM hosted collector is clearly superior**
  => saturation signs are only observed in the most extreme test, at **180 ads/s**



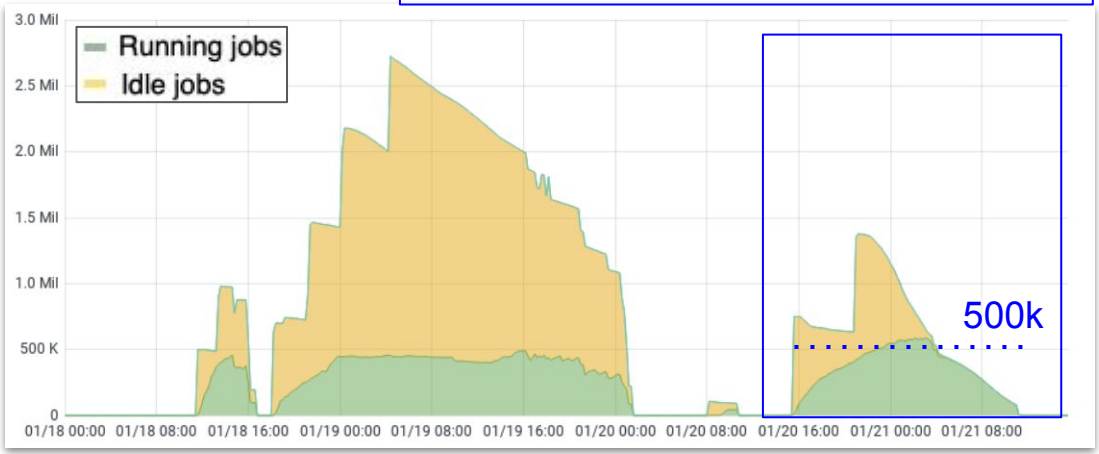
Full infrastructure test (I)

Explore increasingly larger test pools focusing on the PM

- Millions of single core jobs in the schedd queues to maximally fragment the pool for max stress on the collector
- Running multiple startds per pilot (überglideins)
- Job specifications (memory, disk, and site whitelist) randomly generated, runtime of 8+/- 2 h gaussian
- Schedds: went from 10 to 15 job submit nodes increase job on the pool. Each schedd managed to run ~45k jobs, constrained by memory

Reached the milestone of >500k running jobs

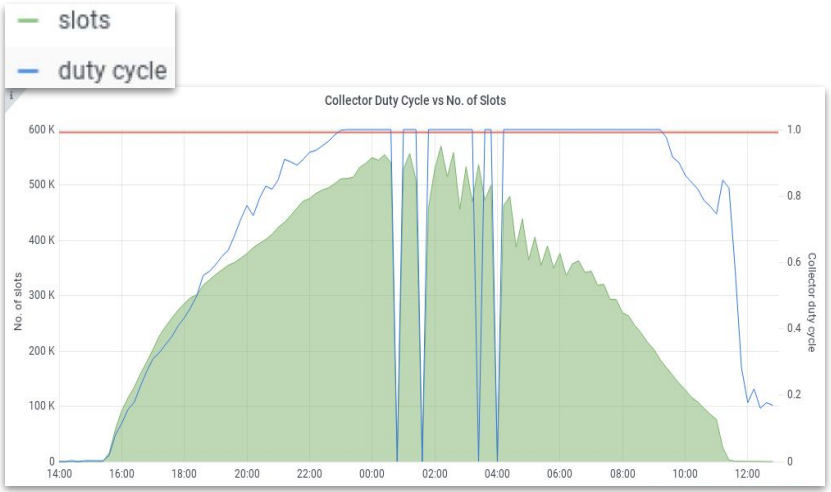
15 schedds, nearly 590k running jobs achieved!
Combined job start capacity at ~60 Hz

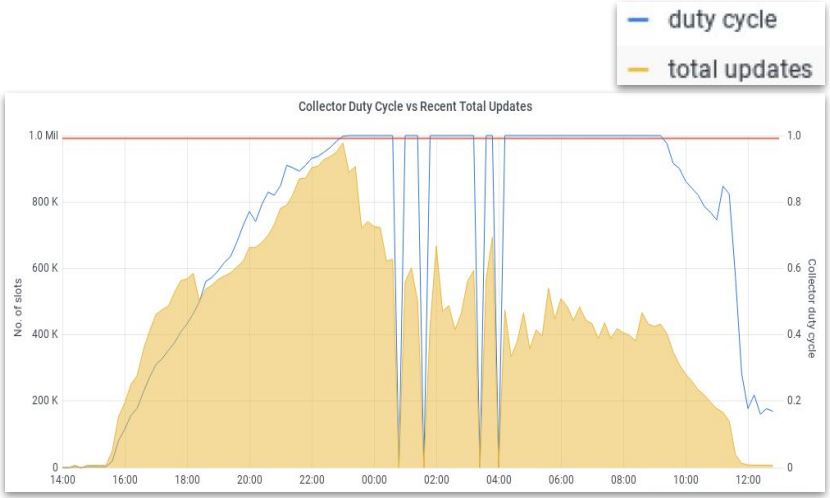


Full infrastructure test (II)

Collector duty cycle in correlation with the number of slots in the pool, and the rate of slot updates (last test phase)

- **The collector saturated (duty cycle %) at 100% around 500k slots and with 1M slot updates (in 20 min integration window)**
- Beyond saturation collector performance is progressively degraded. Missing slot updates:
  - impact on slot utilization efficiency (missed matchmaking opportunities)
  - unreliable monitoring of pool resources

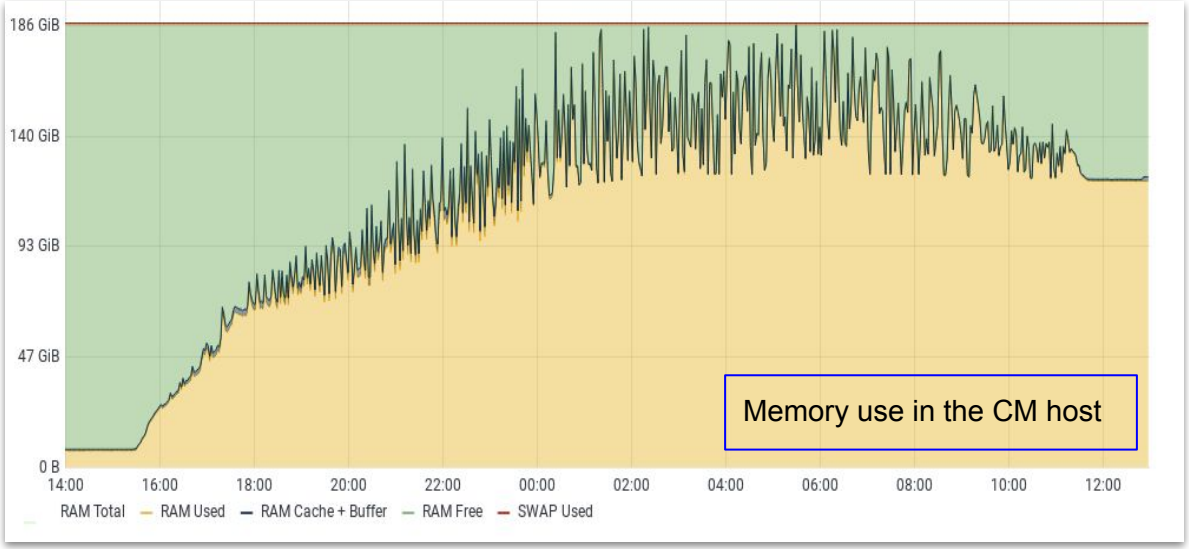




Full infrastructure test (III)

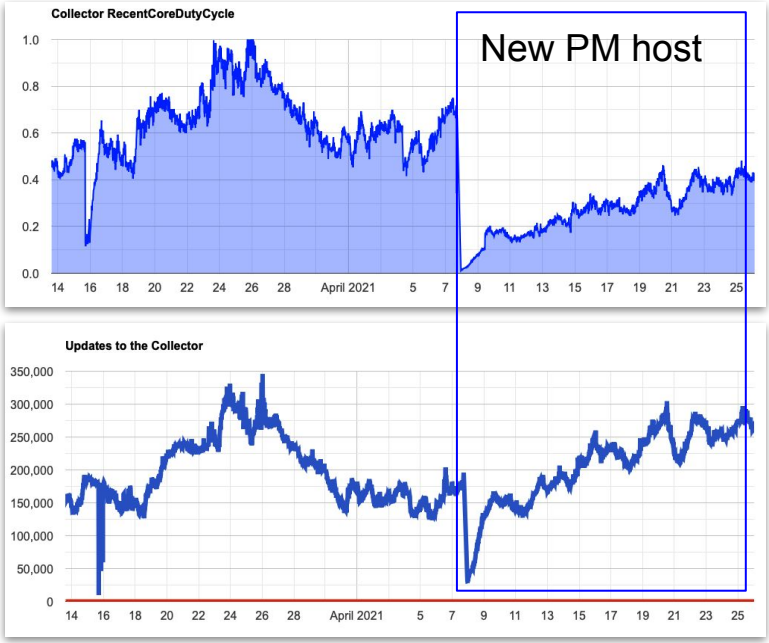
- **Memory consumption in the Central Manager scales with the size of the pool**
- Two components:
  - Baseline memory usage
  - Bursts for each negotiation cycle

- Total **memory usage nearly reached saturation in the tests**
  - **CM host memory represents a clear limiting factor to pool growth**

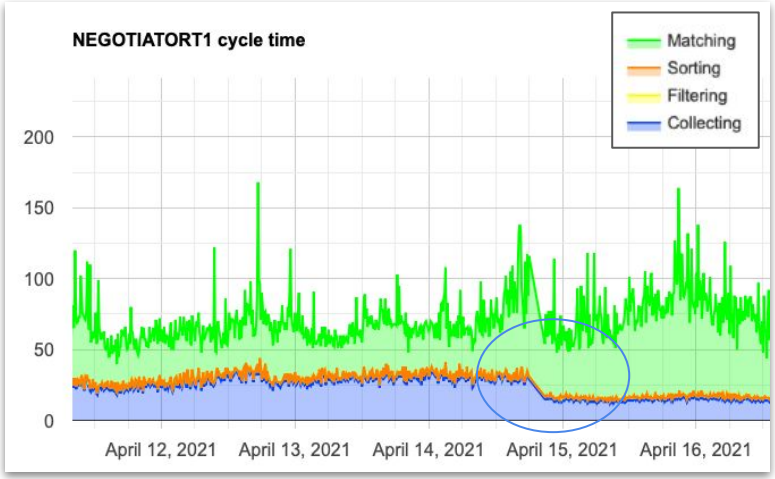


Already applied to CMS Global Pool

- Following tests, the CM of the **production pool** has been **migrated from a VM to a PM:**
  - **No disruption of the activities**
  - **Noticeable improvements in collector performance**



Reduction in matchmaking cycle



- Conclusions

Conclusions

- **The Submission Infrastructure is a stable and performant piece of CMS Computing**, **continuously being reviewed, upgraded and expanded**, in particular prevent scalability bottlenecks

- **The Q1 2021 scalability tests** of the CMS Global Pool:
  - Continued tuning of the infrastructure settings
  - Observed **improved performance** of the collector running on a **PM** instead of a VM
  - **Bottlenecks** identified in the **collector** capacity to process slot updates and **memory** use in the CM host
  - **Exceeded the significant value of 500k concurrently running jobs**

- **Outlook:**
  - **Scalability of our main HTCondor pool has been pushed to confidently cover the expected CMS CPU requirements in the medium term**
  - **Still, continually evaluate new HTCondor releases and HW infrastructure to ensure we stay ahead of the installed and required CPU capacity**

We thank the HTCondor and GlideinWMS development teams for their continued support, a model of excellent partnership!

BACKUP SLIDES

Abstract

The CMS experiment at CERN employs a distributed computing infrastructure to satisfy its data processing and simulation needs. The CMS Submission Infrastructure team manages a dynamic HTCondor pool, aggregating mainly Grid clusters worldwide, but also HPC, Cloud and opportunistic resources. This CMS Global Pool, which currently involves over 70 computing sites worldwide and peaks at 300k CPU cores, is capable of successfully handling the simultaneous execution of up to 150k tasks. While the present infrastructure is sufficient to harness the current computing power scales, CMS latest estimates predict that at least a four-time increase in the total amount of CPU will be required in order to cope with the massive data increase of the High-Luminosity LHC (HL-LHC) era, planned to start in 2027. This contribution presents the latest results of the CMS Submission Infrastructure team in exploring the scalability reach of our Global Pool, in order to preventively detect and overcome any barriers in relation to the HL-LHC goals, while maintaining high efficiency in our workload scheduling and resource utilization.

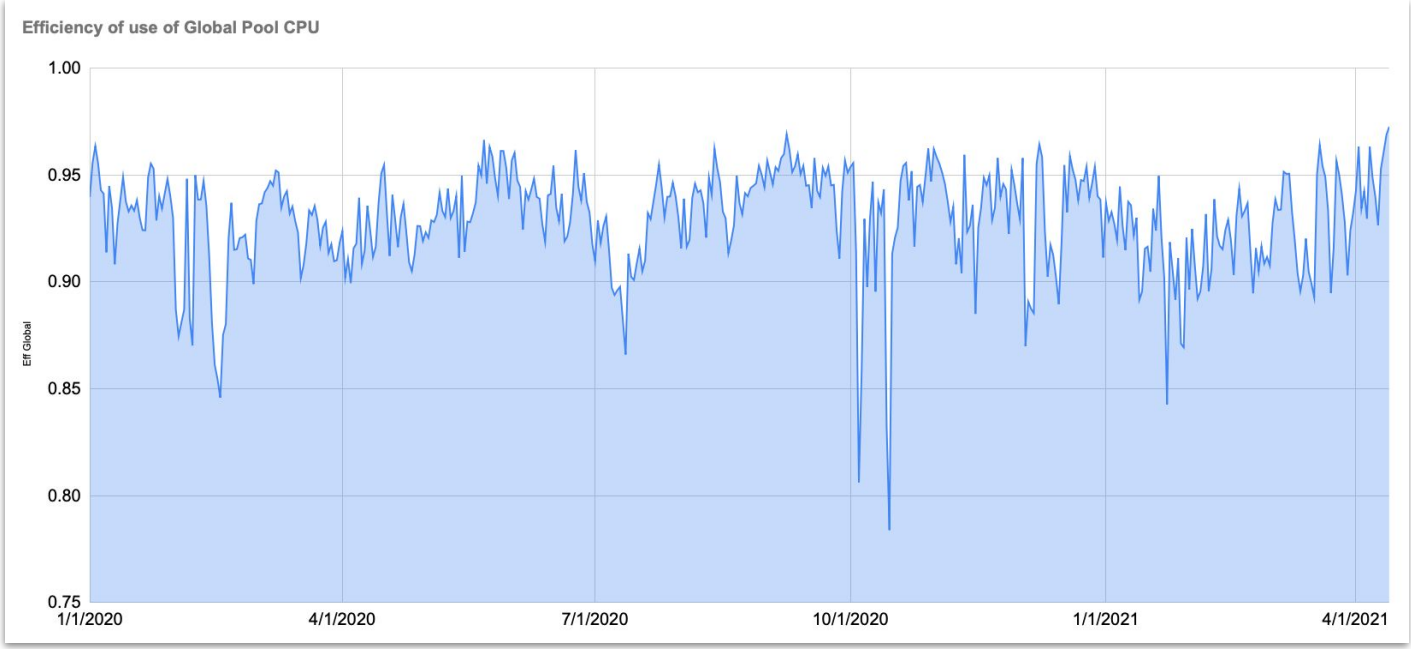
Job scheduling

The submission infrastructure manages workload to resource matchmaking covering diverse directives:

- Users want workflows to complete in a predictable amount of time and with limited manual intervention
  - Either centrally launched with WMAgent, or personal CRAB workflows
  - Don't usually care about efficiency or cost, just time to completion and success rate
  - Sometimes there are *crazy* requirements (e.g. one CPU core + 20GB of RAM)
- Priorities must be respected (*higher prio work should complete sooner/faster than lower prio*)
- Resources used all the time efficiently!

Efficiency

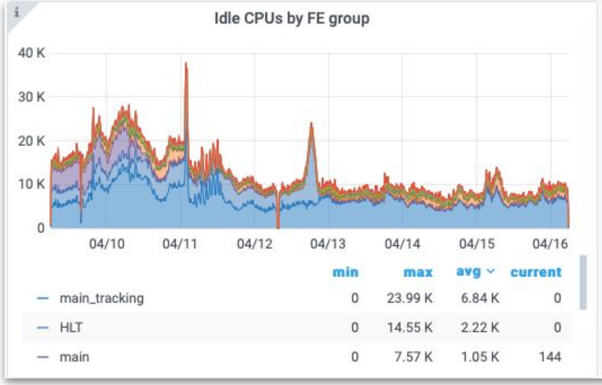
Efficiency results in the global pool typically in the **90%-95% range** (see e.g. since Jan 2020 until today)

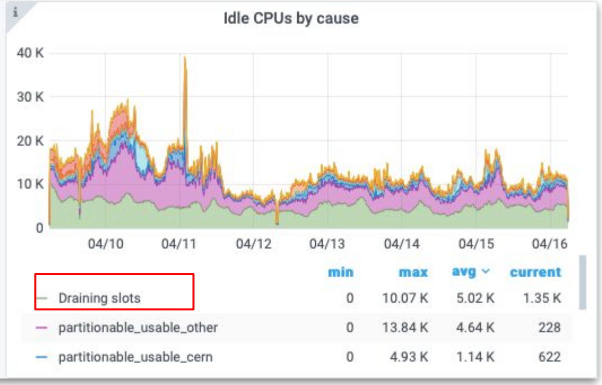


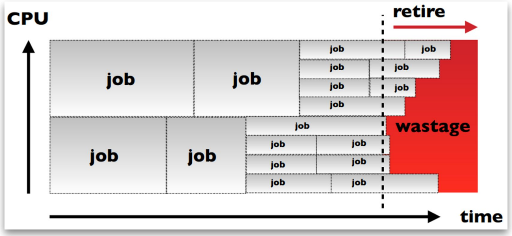
Efficiency

Work in identifying and reducing the sources of slot usage inefficiency

- **Pilot retire time**, inherent to the pool design, can be reduced with optimal timing parameters
- **Memory constraints**, depending on the **requests** from users (Production + Analysis)

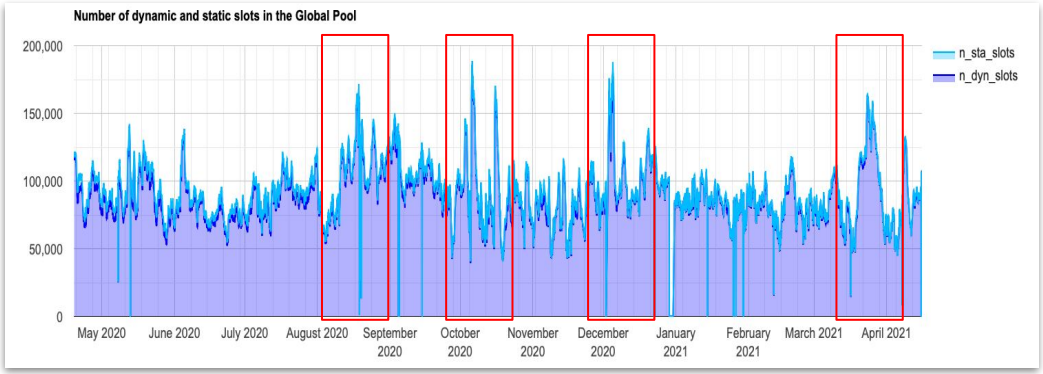






The scalability challenge

- Even during regular operations, there is a risk of excessive load on the pool due to unexpected so-called **single-core job storms**
- Dynamic partitioning of the slots can lead to a maximally fragmented Global Pool, when the CMS workload is dominated by single-core tasks.
- This can easily double the number of dynamic slots and slot update rates in the pool, which the collector has to cope with
- These episodes, although infrequent, may represent a severe impact on resource utilization for days



Scalability tests 2021 (details)

Testbed HTCondor (8.9.10) pool: use **2 CM hosts** to estimate the **impact of the virtualization layer** on the collector performance:

- **vocms0809: a virtual machine (VM),** with 24 CPU cores, hosted on a node based on the Intel Xeon CPU E5-2650 v4 @ 2.20GHz processor, with 256 GB of RAM and 4 SSDs of 1.2 TB each.
  - VM not co-hosted with any other service, so performance hits only from HV (Openstack).
- **vocms0803: a physical machine (PM)**, with 32 cores, based on Intel Xeon Silver 4216 CPU @ 2.10GHz processor, with 187 GB of RAM and 2 SSDs of 1.8 TB each.

A **collector** and 3 **negotiator** daemons were deployed on both nodes with an identical configuration.

Job submit from a set of **schedds** (10, later 15), installed on VMs with 32 cores, 57 GB of RAM and 512 GB of High IO disk.

2021 Q1 tests:

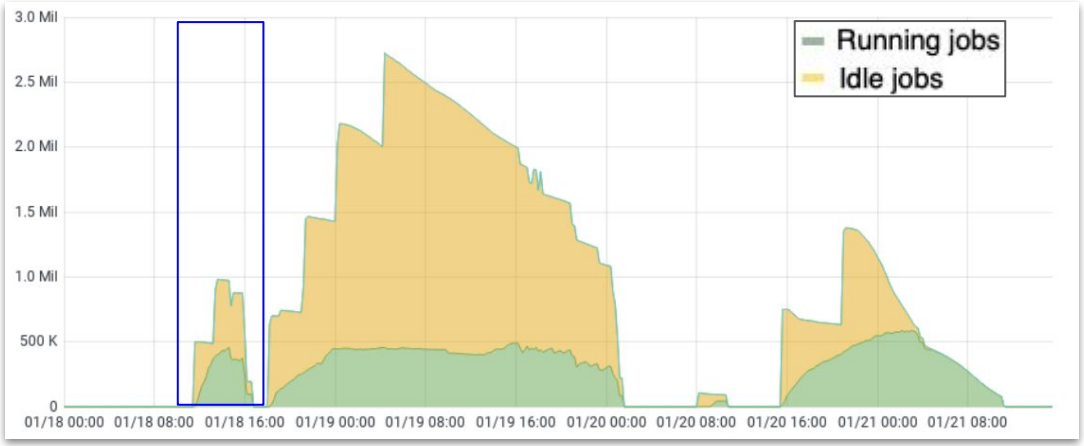
- **Condor_advertise stress tests on the collector**
- **Full infrastructure scalability test**

Full infrastructure test (I)

Explore increasingly larger test pools focusing on the PM

- Millions of single core jobs in the schedd queues to maximally fragment the pool for max stress on the collector
- Running multiple startds per pilot (überglideins)
- Job specifications (memory, disk, and site whitelist) randomly generated, runtime of 8+/- 2 h gaussian
- Schedds: went from 10 to 15 job submit nodes increase job on the pool. Each schedd managed to run ~45k jobs, constrained by memory

First test: scale limited by number of available schedds: 10 submit nodes, saturated at running ~45k jobs each

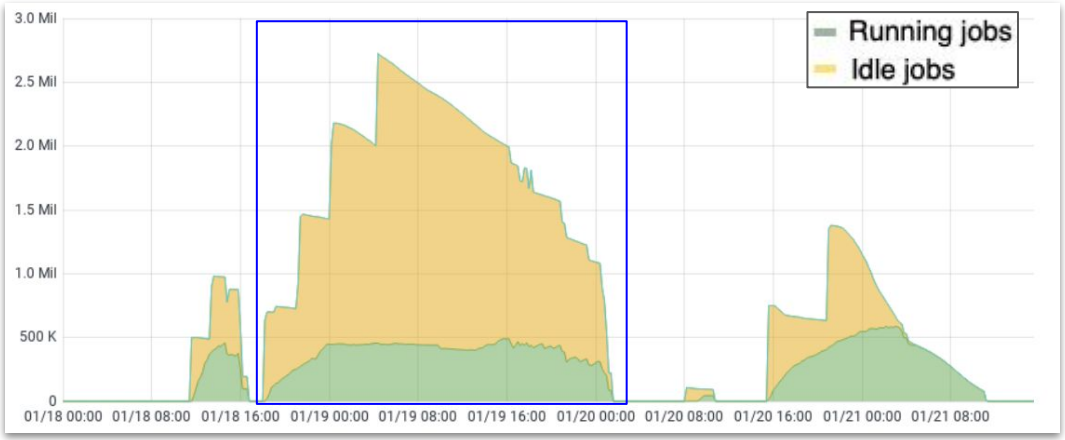


Full infrastructure test (I)

Explore increasingly larger test pools focusing on the PM

- Millions of single core jobs in the schedd queues to maximally fragment the pool for max stress on the collector
- Running multiple startds per pilot (überglideins)
- Job specifications (memory, disk, and site whitelist) randomly generated, runtime of 8+/- 2 h gaussian
- Schedds: went from 10 to 15 job submit nodes increase job on the pool. Each schedd managed to run ~45k jobs, constrained by memory

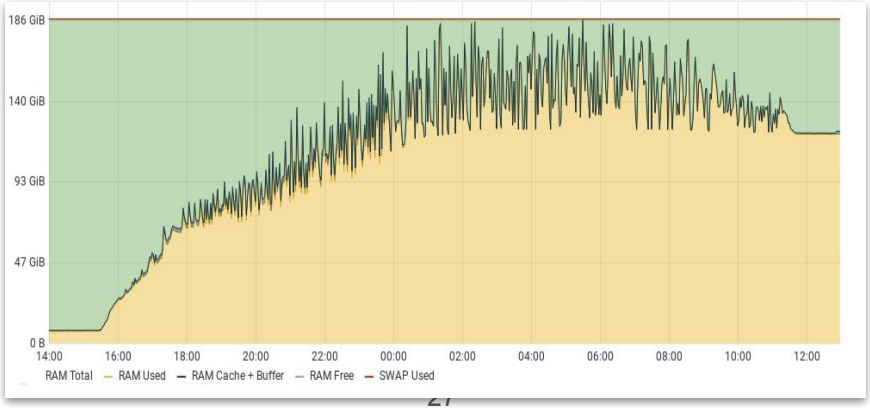
15 schedds in use, but reached the limit of number of TCP sockets in CCB: slots are allocated but can't join the pool!



Full infrastructure test (III)

- The CM runs the main collector daemon, about 100 secondary collectors (intermediary agent between slots and the main collector), plus 3 negotiator processes handling matchmaking. All these processes contribute to memory consumption, which scales with the size of the pool
- Two effects: increasing baseline memory usage, up to ~120 GB in this tests, plus pattern of bursts, each peak representing an increment of an additional 40% to 50% over the baseline memory value.
  - bursts caused by forked collector workers spawned to reply to negotiator queries (each peak represents a new negotiation cycle)
- Total memory usage nearly reached saturation in the tests
  - CM host memory represents a clear limiting factor to pool growth

Memory use in the CM host



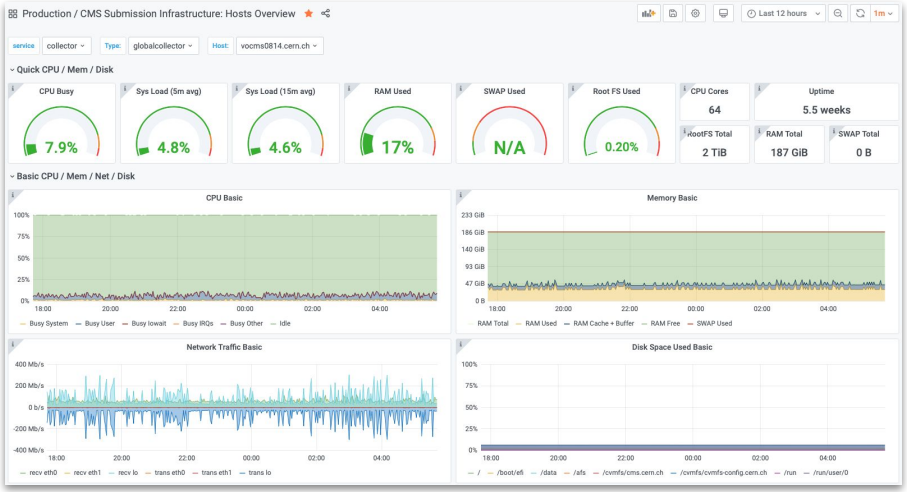
Monitoring efforts

Monitoring such a complex system as CMS Sub. Inf. **is essential to its successful operation!**

- A complete set of monitoring dashboards developed for SI operations by our team
  - Including alerts to improve CMS SI response (no 24/7 coverage though)
  - Collector and negotiator host performance
  - Better understanding and mitigation of sources of scheduling inefficiency

Service Alerts

CM monitor



Schedd monitors