The CMS Submission Infrastructure deployment

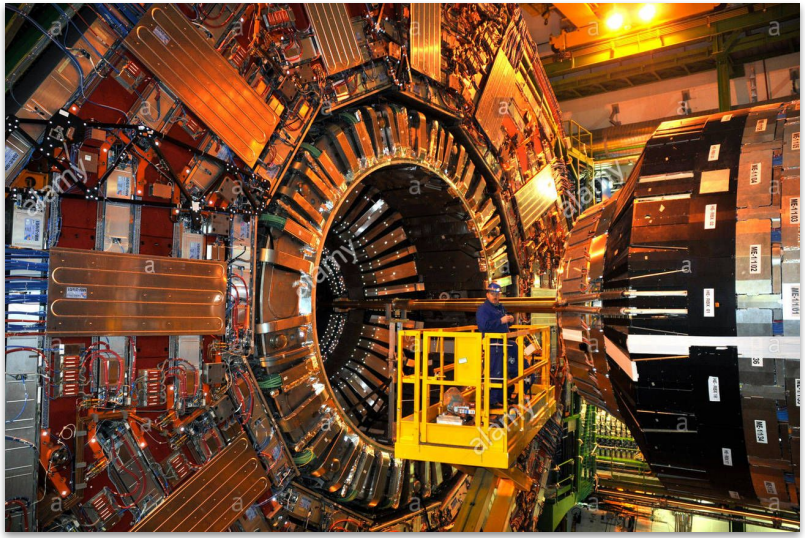
Saqib Haleem, Marco Mascheroni, Antonio Perez-Calero Yzquierdo
for the CMS Submission Infrastructure team

HTCondor Workshop Autumn 2021

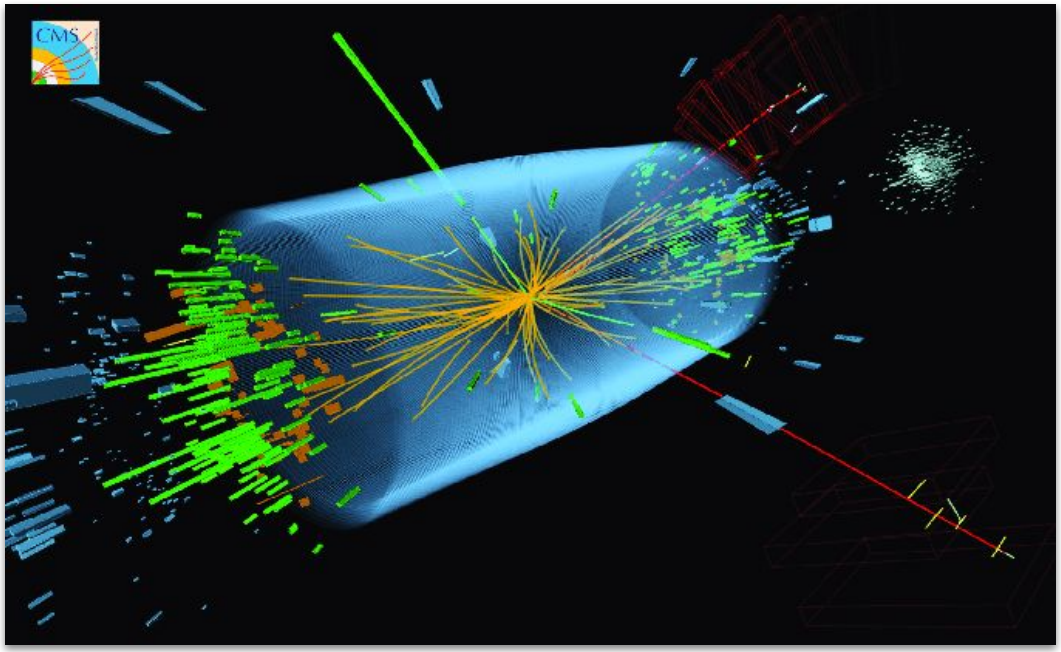
Outline

- The CMS experiment
- The CMS Submission Infrastructure setup: the Global Pool
- Additional pools and submit nodes
- Some non-standard resources
- Future prospects and Conclusions

The CMS experiment at CERN



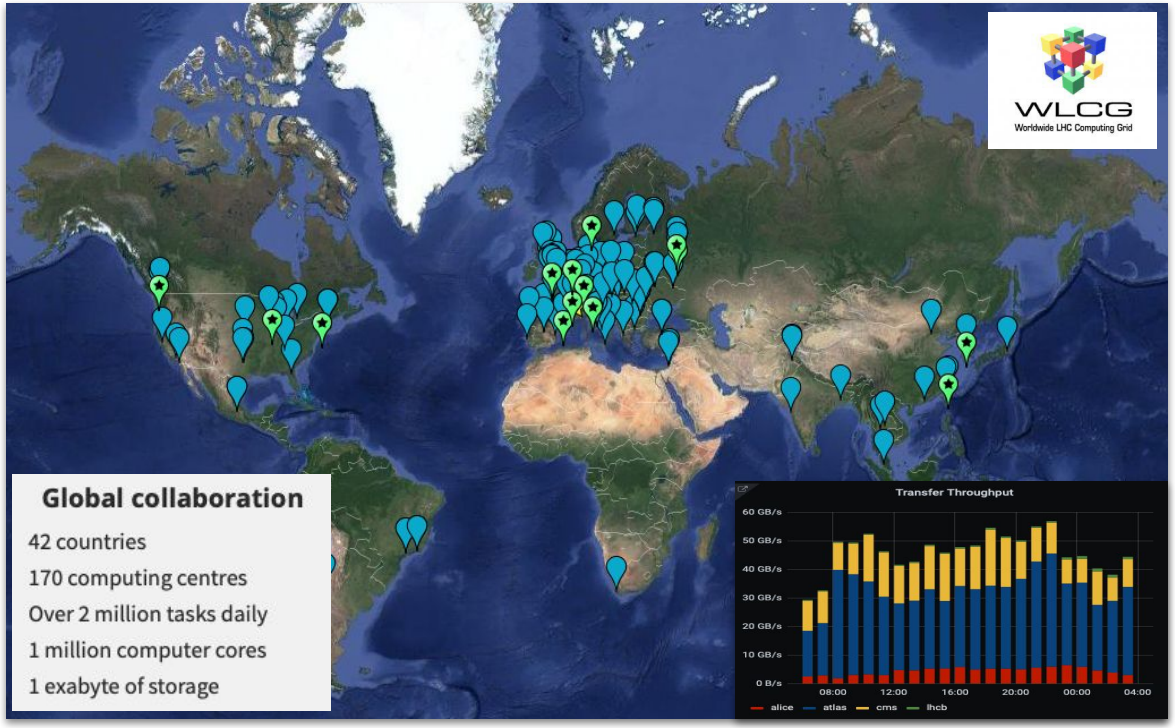
- **High Energy Physics general-purpose experiment** designed to precision study of proton-proton collisions **at the LHC** at **CERN**

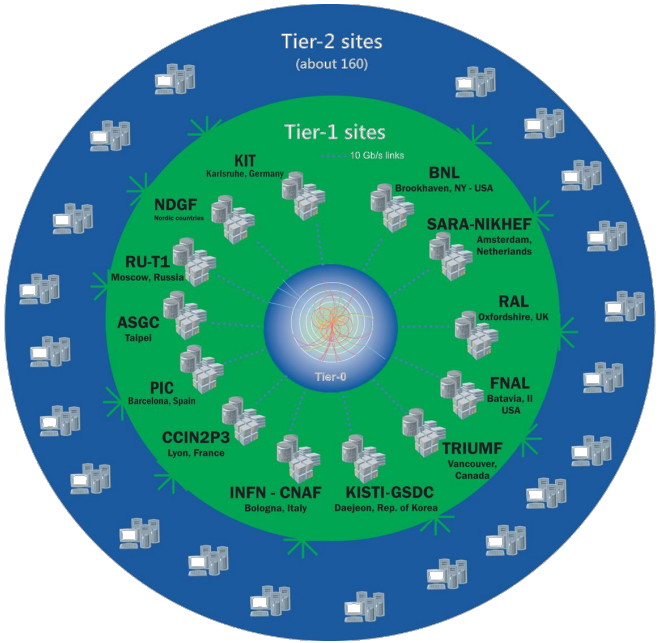


- **Massive amounts of data** (100s of PBs) need to be stored and distributed, processed, simulated and analysed to achieve **CMS scientific goals**

The WLCG

- CMS compute needs are mainly covered by **WLCG resources,** a **global collaboration** of about 170 computing centers, aggregating **1M CPU** cores and **1 EB of storage** (disk and tape)

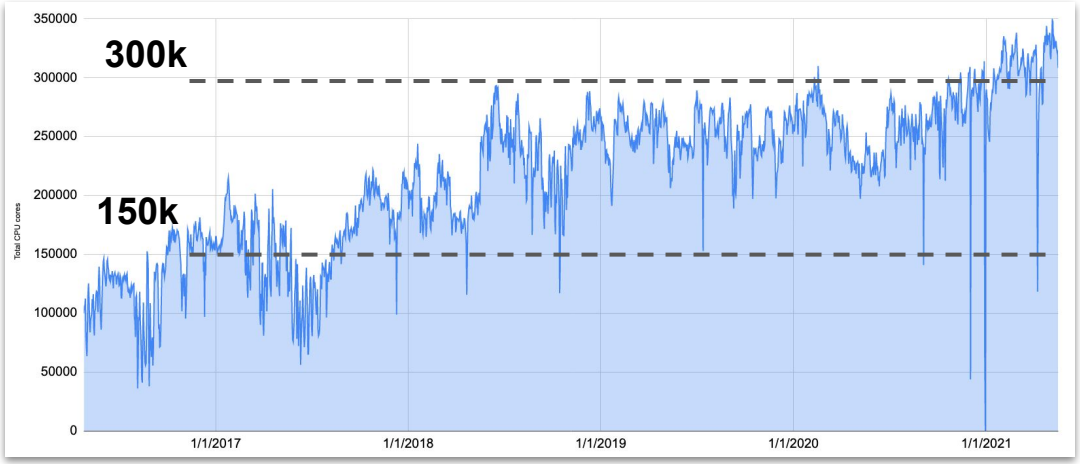




The CMS Submission Infrastructure Group

- Within CMS Computing operations, the **Submission Infrastructure (SI)** team is mandated to manage the major part of CMS CPU resources.
  - **HTCondor**, along with GlideinWMS, are our main **technologies of choice**
- In practice we operate a number of HTCondor pools peaking over **300k cores combined, distributed over 70 Grid sites, also aggregating non-Grid resources (HLT farm, HPC sites, Cloud)**

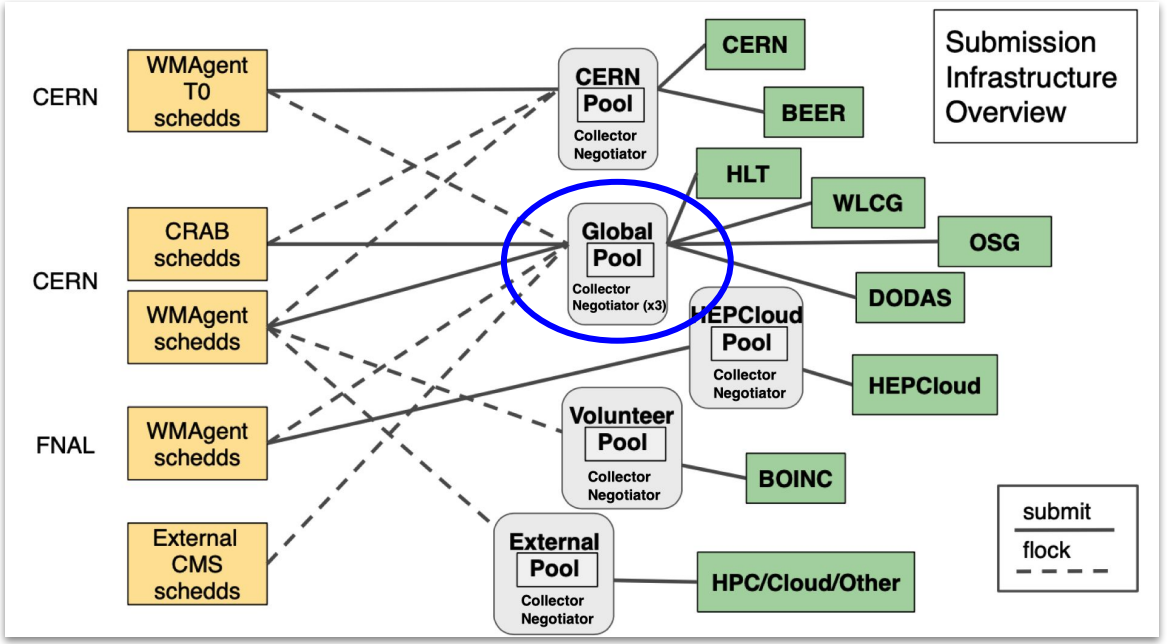
CPU cores in use by CMS over the past 5 years



- We regularly hold **meetings with HTCondor** and glideinWMS **developers** where we discuss current **operational limitations**, **feature** requests aligned with CMS **priorities** and **future scale requirements**

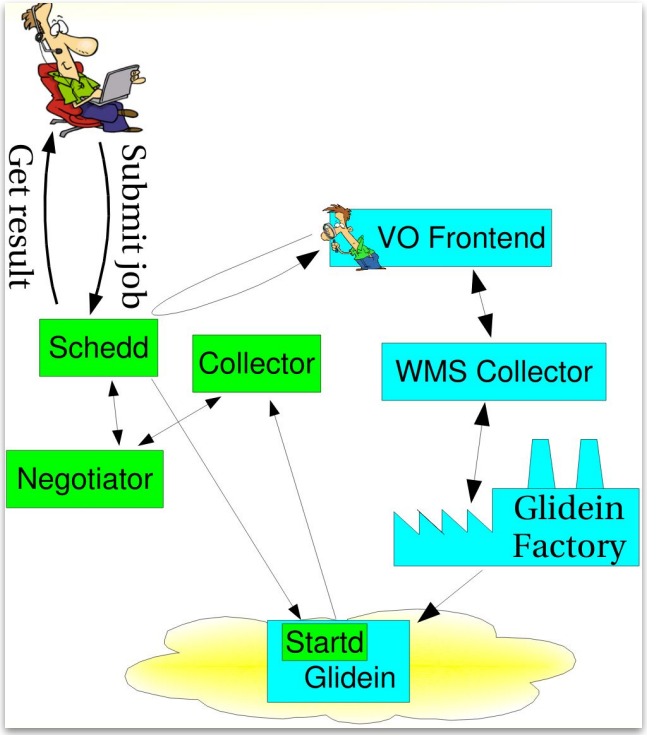
A complex infrastructure

- **The CMS SI model has evolved to running multiple federated pools**, with extensive use of **flocking**
- Multiple sets of specialized workflow managers (CRAB & WMAgent) attached to schedds
- **The main Global Pool:**
  - **Peaks at ~300k CPU cores**
  - **Up to 200k running jobs**
  - 50+ schedds
  - 3 negotiators
- Redundant infrastructure for **HA**



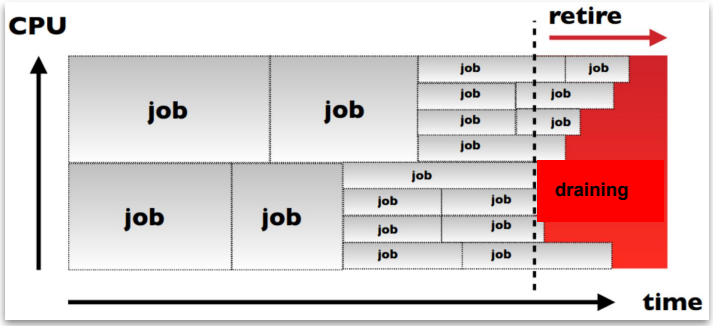
- Resources mainly acquired with **GlideinWMS pilots**
- ...but also **vacuum-like** instantiated: DODAS, BOINC(CMS@Home), opportunistic (HLT), HPC...

Building dynamic HTCondor pools with GlideinWMS



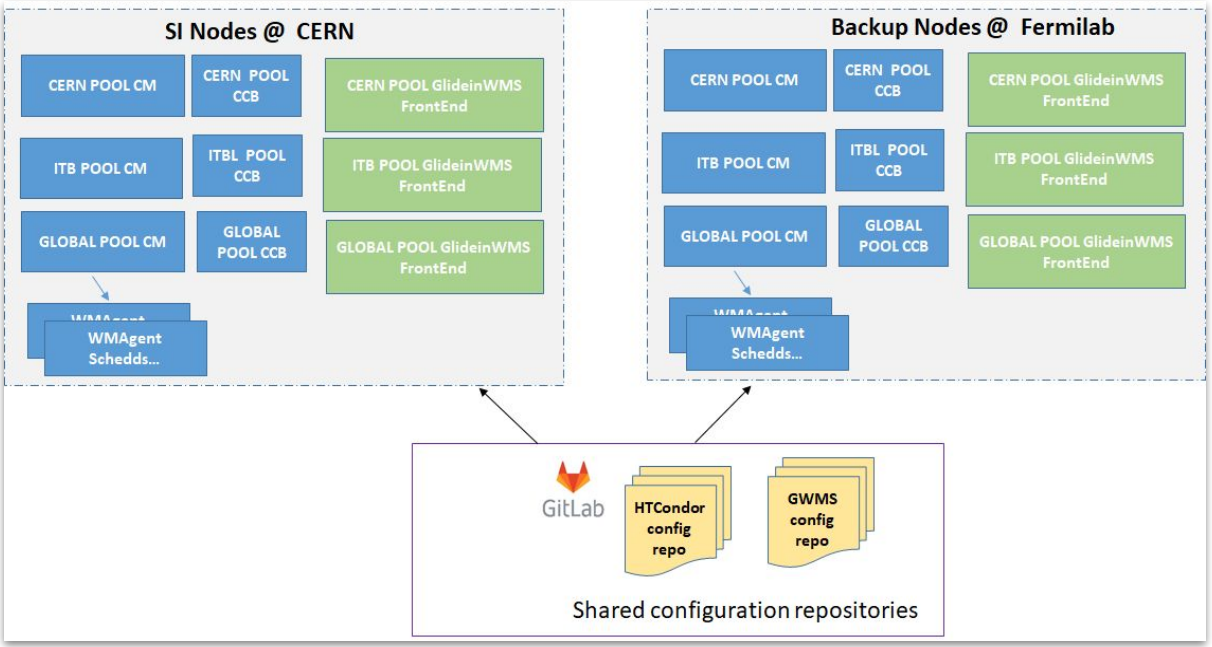
- GlideinWMS and **pilot job model: Frontend generate resource requests, based on schedd queue content** (job **pressure), and matchmaking** with resources (CEs) defined in the pilot factories.
  - **Factories** then **sends pilot** jobs using condor Grid universe
  - **Pilot job** will then **start** the **HTCondor startd** which will connect to the pool

- CMS employs **multicore pilots** with internal **Partitionable** slots (typically 48h pilot lifetime)



The Sub. Infrastructure HA setup

- Infrastructure **hosted at CERN** (primary) is also **replicated at Fermilab** (backup)
- Hosts: physical node for central manager (collector & negotiators), VMs for CCB, schedds, FE and factories
- Mostly running HTCondor 9.1.2
- Shared configuration via CERN GitLab



More details in Saqib's talk!

Notes on Global Pool settings

- The main groups of users of our Global Pool are labelled:
  - ***production*:** executing centralised data processing workflows (e.g. MC data production, experimental data reconstruction)
  - ***analysis**:* end users (physics groups) running analysis tasks on the Grid
- Mapped to separated accounting groups. GROUP_NAMES = production, analysis, highprio, tier0
  - Corresponding to dedicated CRAB (CMS Remote Analysis Builder) and WMAgent schedds in the pool

- Negotiator policy: GROUP_ACCEPT_SURPLUS = true with GROUP_QUOTA_DYNAMIC, adjusted for each of the 3 negotiators, e.g.
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_highprio = 0.99
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_production = 0.0038
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_analysis = 0.0002
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_tier0 = 0.0060
- Fixed negotiation order:
  NEGOTIATORT1.GROUP_SORT_EXPR = ifThenElse(AccountingGroup is null, 0.0,ifThenElse(regexp("^highprio", AccountingGroup), -5.0,ifThenElse(regexp("^tier0", AccountingGroup), -4.0,ifThenElse(regexp("^production", AccountingGroup), -3.0, ifThenElse(regexp("^analysis", AccountingGroup), -2.0, -1.0)))))

- Minimize wastage with conservative approach: **no preemption** and NEGOTIATOR_DEPTH_FIRST = TRUE
- Multithreaded negotiators: NEGOTIATOR_NUM_THREADS = 4
- Time spent in matchmaking is limited to avoid very busy schedds: NEGOTIATOR_MAX_TIME_PER_SCHEDD=60
- Defrag daemon is OFF: natural defragmentation of p-slots due to finite pilot lifetime (~48h)

- To reduce negotiator pressure on the collector: NEGOTIATOR_CYCLE_DELAY=150

- Authentication: SEC_DEFAULT_AUTHENTICATION_METHODS = FS,GSI

- HA setup: CONDOR_HOST1=vocms0814.cern.ch and CONDOR_HOST2=cmssrvz02.fnal.gov

Notes on Global Pool settings

Users

- The main groups of users of our Global Pool are labelled:
  - ***production*:** executing centralised data processing workflows (e.g. MC data production, experimental data reconstruction)
  - ***analysis*:** end users (physics groups) running analysis tasks on the Grid
- Mapped to separated accounting groups. GROUP_NAMES = production, analysis, highprio, tier0
  - Corresponding to dedicated CRAB (CMS Remote Analysis Builder) and WMAgent schedds in the pool

Shares

- Negotiator policy: GROUP_ACCEPT_SURPLUS = true with GROUP_QUOTA_DYNAMIC, adjusted for each of the 3 negotiators, e.g.
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_highprio = 0.99
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_production = 0.0038
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_analysis = 0.0002
  - NEGOTIATORT1.GROUP_QUOTA_DYNAMIC_tier0 = 0.0060
- Fixed negotiation order:
  NEGOTIATORT1.GROUP_SORT_EXPR = ifThenElse(AccountingGroup is null, 0.0,ifThenElse(regexp("^highprio", AccountingGroup), -5.0,ifThenElse(regexp("^tier0", AccountingGroup), -4.0,ifThenElse(regexp("^production", AccountingGroup), -3.0, ifThenElse(regexp("^analysis", AccountingGroup), -2.0, -1.0)))))

Efficiency

- Minimize wastage with conservative approach: **no preemption** and NEGOTIATOR_DEPTH_FIRST = TRUE
- Multithreaded negotiators: NEGOTIATOR_NUM_THREADS = 4
- Time spent in matchmaking is limited to avoid very busy schedds: NEGOTIATOR_MAX_TIME_PER_SCHEDD=60
- Defrag daemon is OFF: natural defragmentation of p-slots due to finite pilot lifetime (~48h)

- To reduce negotiator pressure on the collector: NEGOTIATOR_CYCLE_DELAY=150
- Authentication: SEC_DEFAULT_AUTHENTICATION_METHODS = FS,GSI
- HA setup: CONDOR_HOST1=vocms0814.cern.ch and CONDOR_HOST2=cmssrvz02.fnal.gov

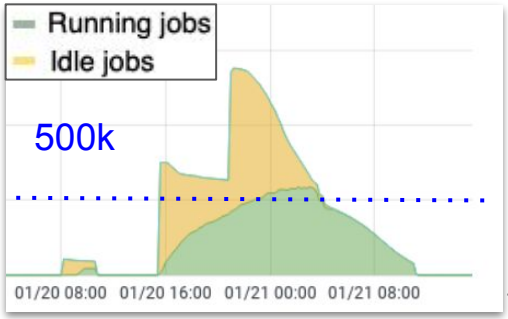
Pushing the limits of the Global Pool

- **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

- **Customized settings:**
  - Using a **CCB** running on a separate host to the CM, with enlarged pool of connection sockets
  - Use of multiple **negotiator** daemons (T1s, US_T2, T2_T3_other) running in **multithreaded** mode
  - Used to run **32-bit binaries** for the **shadow processes** running in the schedds (lower RAM)
  - Hierarchy of **secondary collectors** (~100) 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, in contrast to those from the GlideinWMS FE and CMS WM and monitoring services (condor_q & condor_status)
  - **Redirect non-high prio queries** to the secondary collector (HA infrastructure at FNAL)
  - **Reduced GlideinWMS FE impact** on collector via query caching

- Most of these actions directed at **avoiding the saturation of the top collector**

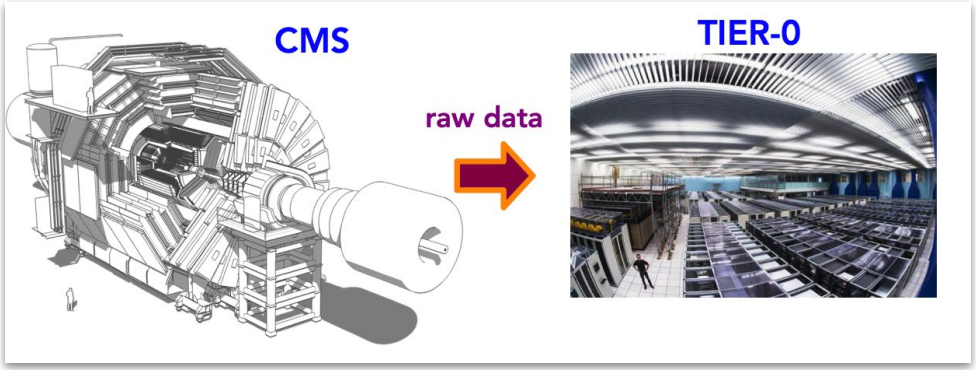
- **2021 tests showing capacity in our testbed to run 500k simultaneous jobs!**



The CERN pool (I)

We employ an **independent infrastructure** to manage the compute **resources at CERN**, which are **critical** in support to the CMS data taking activities: **the Tier-0 tasks**

- **Isolate the very critical T0 tasks** from potential issues on the general infrastructure of the main pool:
  - Dedicated CM, CCB and GlideinWMS FE
  - Dedicated T0 schedds (based on WMAgent)
  - T0 tasks with the highest priority in this pool
  - Forced slot defragmentation available for faster multicore slot creation (at a cost in terms of efficiency)
  - T0 workloads can take over the whole pool if needed, also **flock** into the Global Pool (e.g. T1s) as required
- When not taking data, use the full CERN resources (T0+T2) for offline computing:
  - Global Pool schedds can **flock** production and analysis tasks into the CERN Pool
- Successfully in use during LHC Run 2. Long Shutdown 2 is approaching its end, data in Run 3 is coming
  - **Ready for the new data taking phase!!**

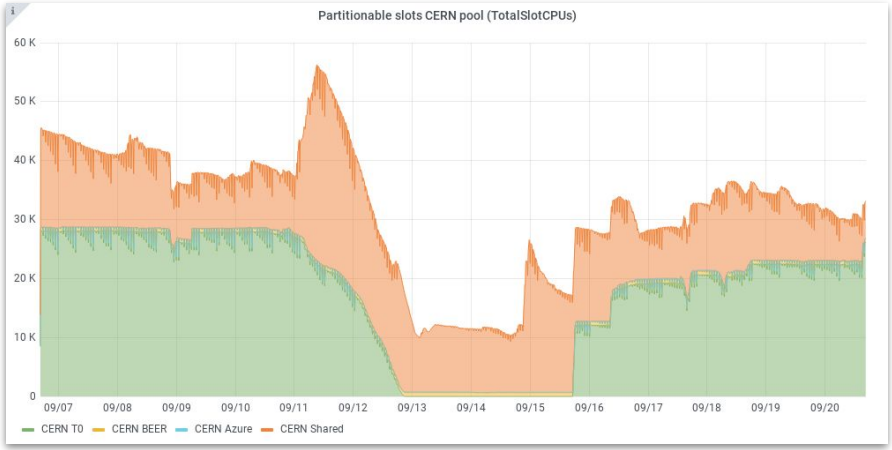


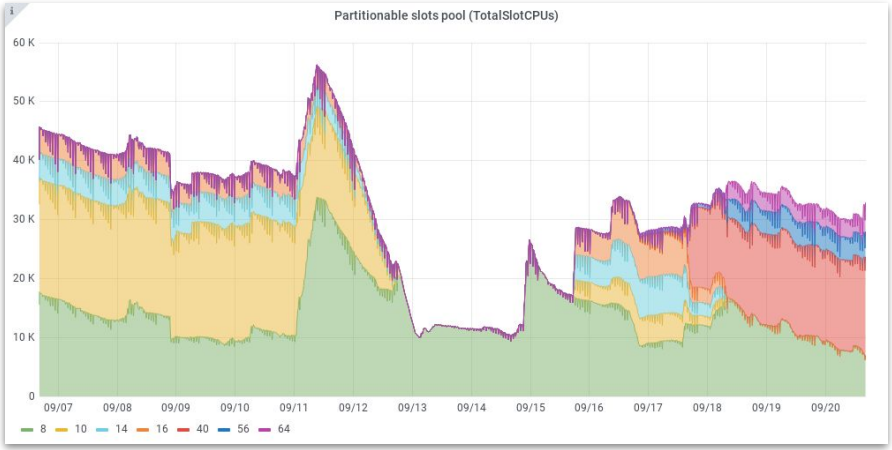
Tier-0 tasks during CMS data taking:

- RAW data repacking
- Prompt detector calibration
- Prompt data reconstruction

The CERN pool (II)

- All CERN resources dedicated to CMS aggregated into the CERN pool
  - Old T0 and T2 CERN partitions combined for improved overall utilization
  - Includes BEER (Batch on EOS Extra Resources) and CERN computing extension into Cloud
- From CERN side, ongoing transition from VMs to bare metal resources
  - For CMS, going from 8, 10, 14, 16 cores fixed-size to **whole-node auto-configurable pilots** (e.g. 64 CPU cores)





Additional pools

- The **HEPCloud** pool
  - Managed by FNAL, designed to aggregate compute resources from FNAL T1 cluster, Cloud (e.g. AWS) or HPC (e.g. NERSC), based on workload demands, local availability, spot price, etc.
  - CMS Workloads can then be flocked from the FNAL-managed CMS WMAgent schedds into HEPCloud pool execute nodes
- The **Volunteer** pool, operating with volunteer computing resources, based on independently launched BOINC glideins and supporting the CMS@Home project
- Other **external** pools: flocking from CMS schedds can be enabled in order to opportunistic use of externally managed HTCondor pool (e.g. tested as initial prototype for the BSC worker nodes integration to PIC)
- **Integration and testbed** (ITB, ITB-dev) pools, employed also for scalability testing

Alternative submit nodes

In addition to CRAB schedds supporting user analysis tasks, our Global Pool enables workload submission from:

- **CMS Connect**: A project providing a single sign-on service for CMS users access to the Global Pool
  - host `login.uscms.org` provides a HTCondor job submission service (schedd)

- **Institutional schedds**: E.g. **MIT** schedds submitting CMS jobs to local computing clusters are also allowed to overflow into unused CMS Global pool slots

Advantages:

- Both facilitate running **arbitrary analysis code** on CMS resources (i.e. not necessarily based on the main CMSRun executable), hence a **complement to CRAB** in support of physicists distributed analysis tasks
- In general, they provide single-core jobs **backfilling** into slot leftovers, therefore improving the overall **utilization efficiency** of the pool

IPv6 readiness

- IPv6 tested in the ITB pool successfully
  - Using HC jobs running on IPv6-only WN at CERN

- CERN Pool is already running on Dual Stack (with IPv4 preference).
- All SI nodes at CERN (Schedds, collectors, CCBs and GlideinWMS services) support dual stack.
- Full migration of CMS Global Pool postponed due to ongoing issue with HA daemon when prefered IPv6 (i.e.PREFER_IPV4 = False) is enabled.
  - Will resume the activity once fix will be available in following releases

Near future goal: provide support for IPv6-only CPUs

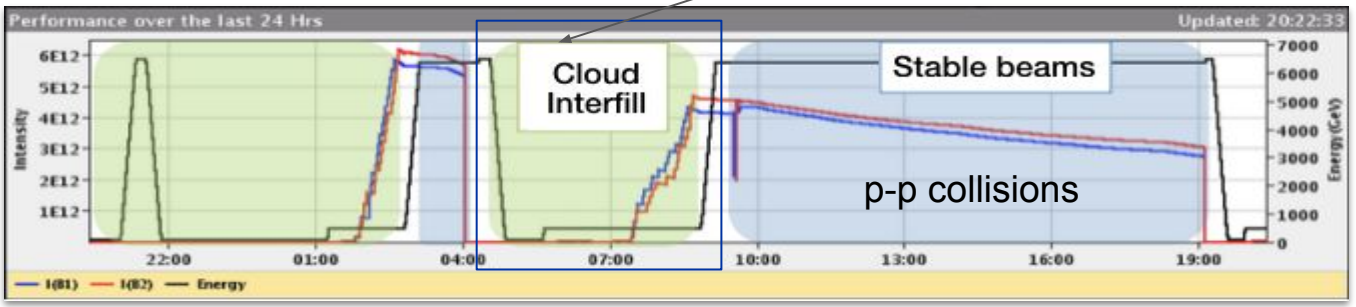
However, full WLCG IPv6-only still far away into the future (LHC LS3?)

Some non-standard resources

The CMS Online Cloud (HLT)

- The **CMS Online Cloud** (deployed on the High Level Trigger farm resources) was commissioned during Run2 to dedicate HLT **resources for offline data processing** when not needed to support data taking (ref)
- Launch Openstack VMs running the startds when CPU is available (**Interfill + Fill modes**)
  - VMs suspended when HLT needs resources back
- A **`MaxHibernationTime`** delay (24h) is applied at both CCB and WMAgents schedds in order to ensure job execution can be resumed when the VM is back into the Global Pool

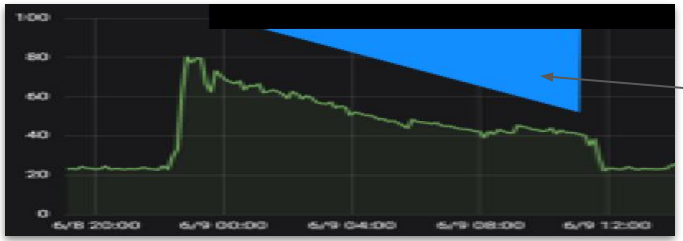
INTERFILL



LHC operation cycles

DURING LHC FILLS

HLT resources % required for DAQ



HLT resources available for Global Pool

Opportunistic resources integration

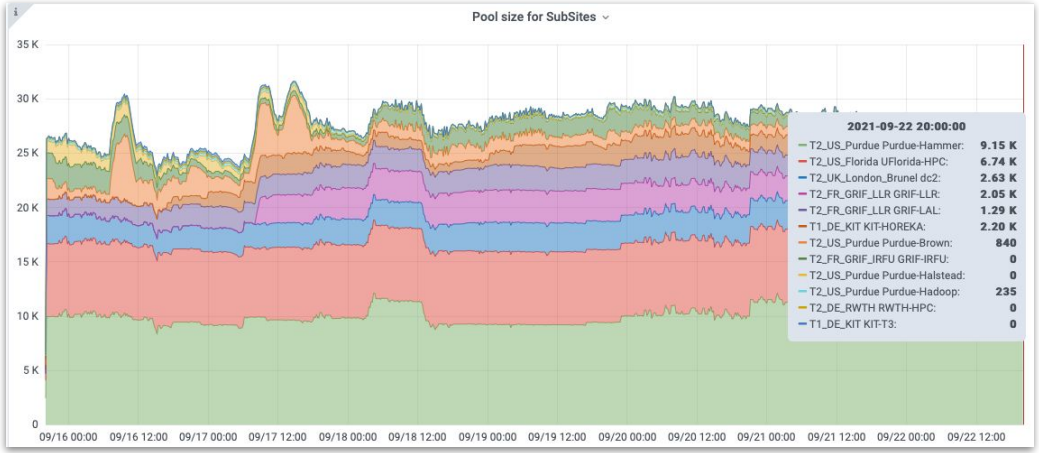
Ongoing efforts to continue integrating new resources into the CMS Global Pool

Frequently managed as “extension” (or SubSites) of WLCG T1 or T2 centers with pledged resources

- **HPC**: CINECA (T1_IT_CNAF), BSC (T1_ES_PIC), Jülich and HOREKA (T1_DE_KIT)
- **Cloud**: CERN_Azure, PIC_AWS
- **Opportunistic** use of local clusters: CERN_BEER, KIT_T3, Purdue...

Most include **custom start** expressions, with **additional filters** to tune matchmaking of jobs to **non-standard resources**

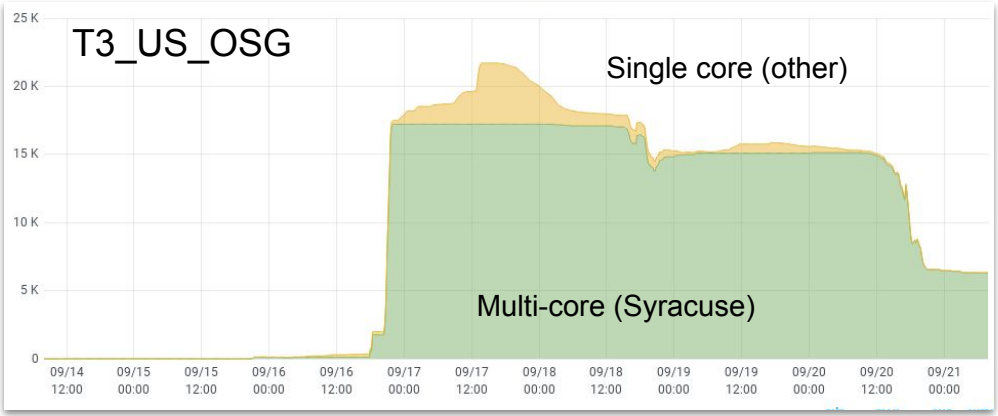
- E.g. **avoid analysis jobs** on HPCs, or just run **GEN-SIM jobs** (no remote storage access required) on opportunistic resources



OSG Pool resources into the Global Pool

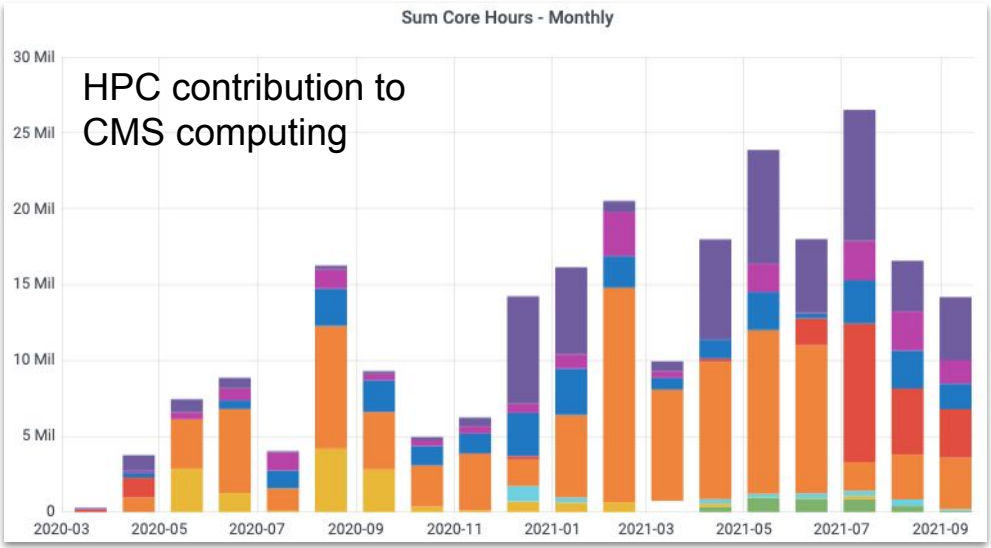
- In addition to resources pledged to CMS by USCMS sites, we are kindly allowed to submit **CMS pilots on to OSG pool for opportunistic usage**

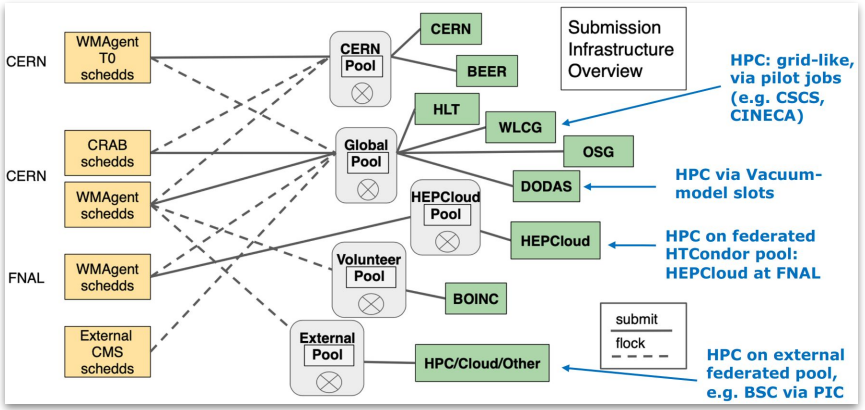
- A number of sites and CEs are now configured in our pilot factories under a common T3_US_OSG name
  - Main contributor being Syracuse University which kindly recently agreed to support our standard 8-core pilots



HPC resources

- CMS exploitation of HPC resources has **intensified in the last year**
  - Cori, Stampede, Bridges, Theta, Expanse Frontera, Marconi, Jülich, BSC
- Multiple integration strategies are being employed





Future prospects and Conclusions

Future Prospects

- Further **expansion** into more **diverse** resource types (HPCs and Cloud)
- Full support for heterogeneous resources and workflows (**GPUs**)
- Keep up **scalability** of our infrastructure despite growing CMS demands for resources (**HL-LHC**)
- Full adoption of **IPv6** (dual stack readiness, prefer IPv6)
- Transition to **token-based** authorization

Conclusions

- The Submission Infrastructure is a **stable** and **performant** piece of CMS Computing, continuously being reviewed, upgraded and expanded in **support of CMS scientific goals.**
- Our infrastructure has evolved to satisfy the requirements of **capacity, stability, flexibility and efficiency**.
- New challenges ahead, as the **landscape** of resources for CMS is continuously **evolving** towards higher scales, and now also heterogeneity.

We have achieved a lot thanks to HTCondor technologies and close collaboration over many years!

Extra Slides

Abstract

The CMS experiment at CERN requires vast amounts of computational power in order to process, simulate and analyze the high energy particle collisions data that enables the CMS collaboration to fulfill its research program in Fundamental Physics. A worldwide-distributed infrastructure, the LHC Computing Grid (WLCG), provides the majority of these resources, along with a growing participation from international High Performance Computing facilities. The combined processing power is harnessed for CMS use by means of a number of HTCondor pools operated by the CMS Submission Infrastructure team. This contribution will present a detailed view of our infrastructure, encompassing multiple HTCondor pools running in federation, aggregating hundreds of thousands of CPU cores from all over the world. Additionally, we will describe our High Availability setup, based on distributed (and in some cases replicated) infrastructure, deployed between the CERN and Fermilab centres, to ensure that the infrastructure can support critical CMS operations, such as experimental data taking. Finally, the present composition of this combined set of resources (WLCG, CERN, OSG and HPC) and their roles will be explained.

HTCondor features in use by CMS

- **Flocking** used to migrate jobs from the pool where they naturally belong to other pools where they can run
- **Dagman** used by CRAB to manage jobs **resubmission and transfer** of output files
- **JobRouter** used to adjust on the fly user job requirements
  - *Overflow*: jobs idle for a long time on a site
  - Tuning of jobs requirements (i.e.: walltime)
- **Custom start** expressions
  - Site admins can specify in their local resource configuration extra requirements for their jobs; useful for expansion of the pool to **non-standard resources**
  - E.g.: just run GEN-SIM jobs on opportunistic resources
- **Resizable jobs** have been introduced to improve **resource usage efficiency**
  - Jobs will match a range of resources (CPUs/Memory)
  - A job can reconfigure itself at runtime based on the resources allocated