



Two decades of HTCondor and CMS success story

A. Pérez-Calero Yzquierdo, M. Mascheroni, F. Von Cube, V. Zokaite, H. Kim on
behalf of the CMS Collaboration

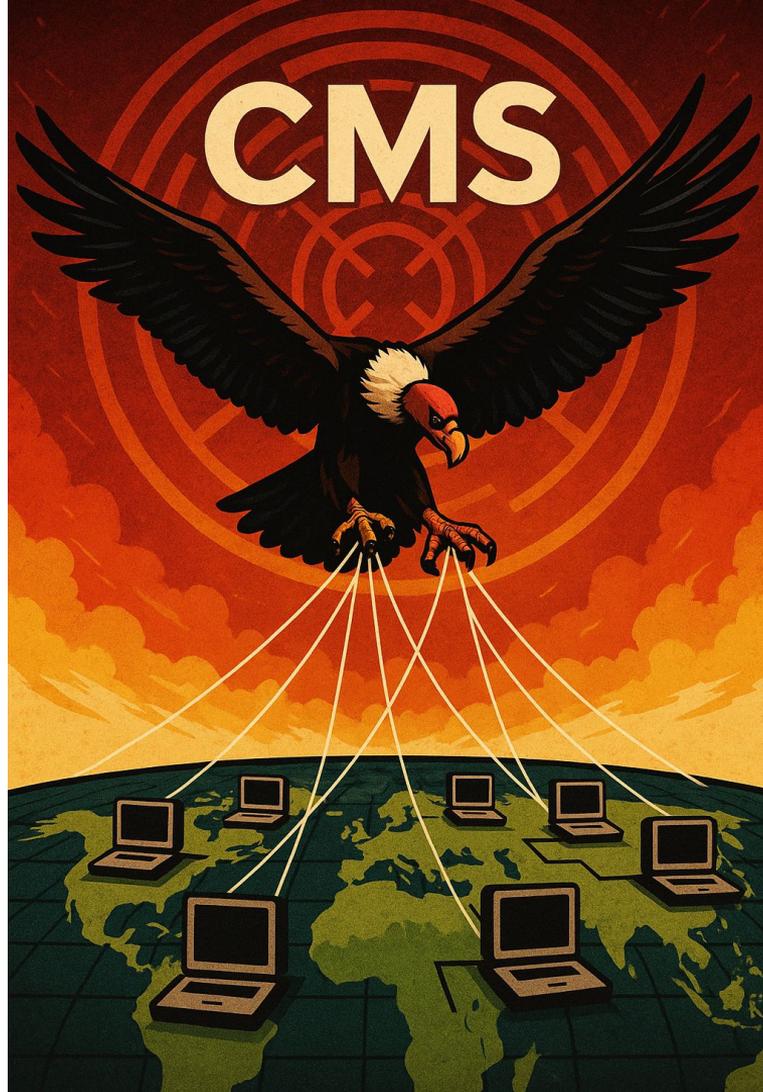
HTCondor Workshop Autumn 2025
2025/09/16, Prague



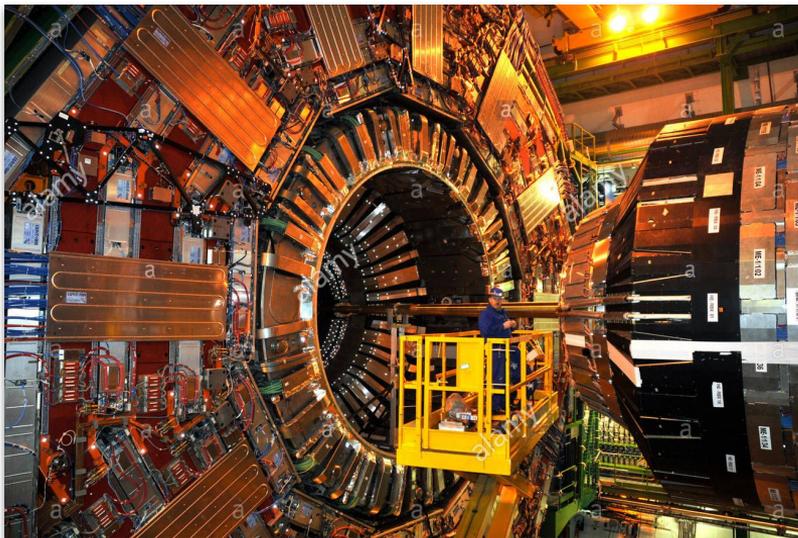
UC San Diego





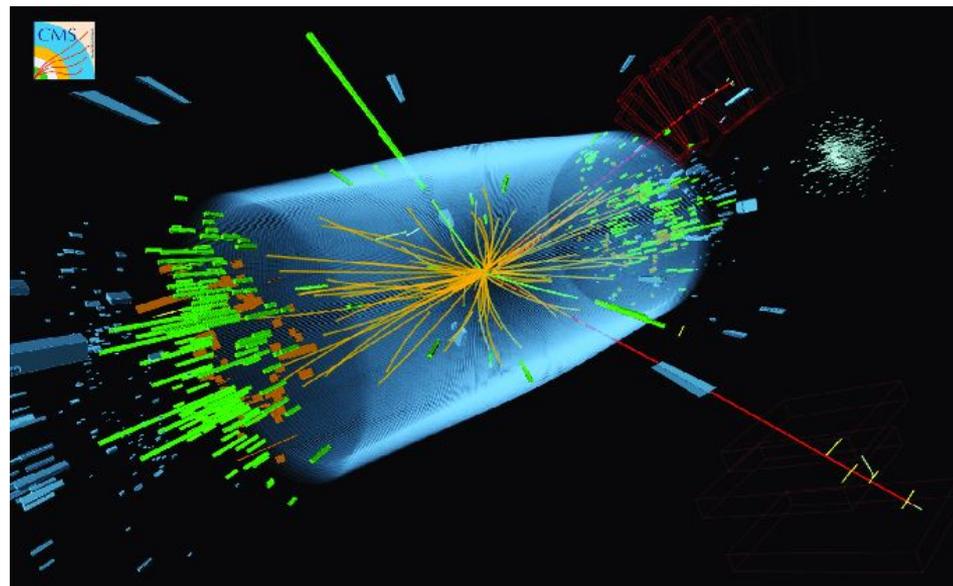


The CMS experiment at CERN



- **High Energy Physics general-purpose experiment recording proton-proton collisions at the LHC at CERN**

- Experimental data is stored, distributed, reconstructed, and analyzed, comparing to simulated data (Monte-Carlo)
 - **Hundreds of PBs per year**



The computing landscape - the WLCG

- Data traditionally analyzed using **Worldwide LHC Computing Grid (WLCG) resources**
 - Global collaboration of around 170 computing centers
 - Access based on dedicated resources (**pledges**)
 - **Over 1M CPU cores and 2 EB in data storage**



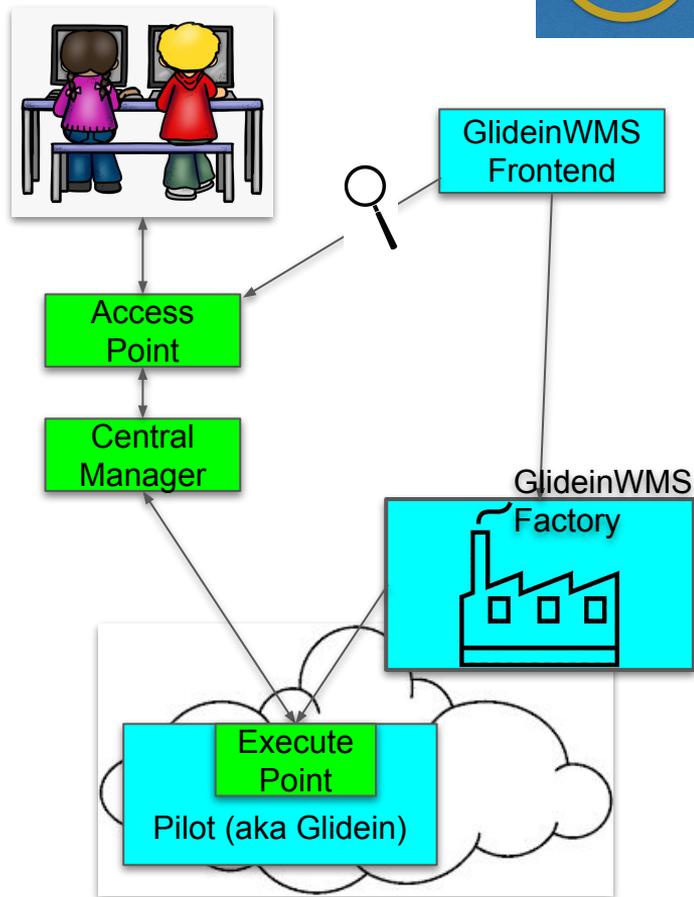


The CMS Submission Infrastructure Group



- Part of CMS Offline and Computing in **charge** of:
 - Organizing **HTCSS** and **GlideinWMS** operations in CMS, in particular of the **Global Pool**, an infrastructure where reconstruction, simulation, and analysis of physics data takes place
 - Communicate CMS **priorities** to the development teams of **GlideinWMS** and **Condor**
- In practice:
 - We operate a set of federated pools of resources **distributed over 70 Grid sites, plus non-Grid resources**, managed by HTCondor

CMS relies on a flexible and scalable pilot-based late-binding model for resource acquisition and workflow execution! Let's review this successful story!



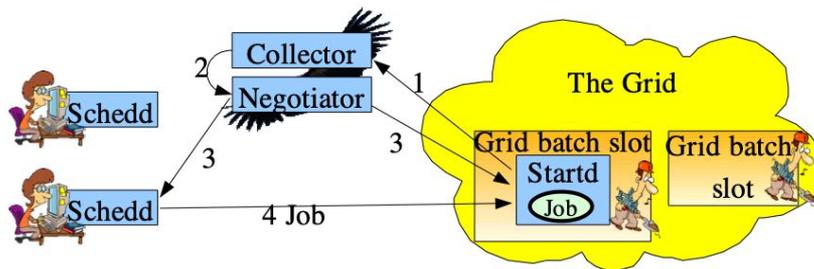


How did we arrive here...?

Around 2007, the idea of the “**pilot job**” coupled with **condor** was introduced for Grid resources. This represented the birth of **glideinWMS**, started to be used by diverse experiments in the US, including CMS.

- Facing the **Grid heterogeneity**, “*software development effort (would be required) to obtain a **Grid-wide WMS**”*. But then: “***alternative approach** to the problem is to create a **homogeneous virtual private pool of compute resources**, and use a standard batch system to manage them*” (I. Sfiligoi, CHEP07, [ref](#)).
- Moreover: “*Having started as a **scavenger batch system**, Condor is an **excellent choice for a pilot WMS**”*
 - “***all a pilot job needs to do is configure and start a startd***”. (I. Sfiligoi et al, [ref](#)).

A Condor glidein is simply a startd started as a Grid job. Once the startd registers with the collector, a schedd can send a job as it would do in a local pool, as shown in figure 3.

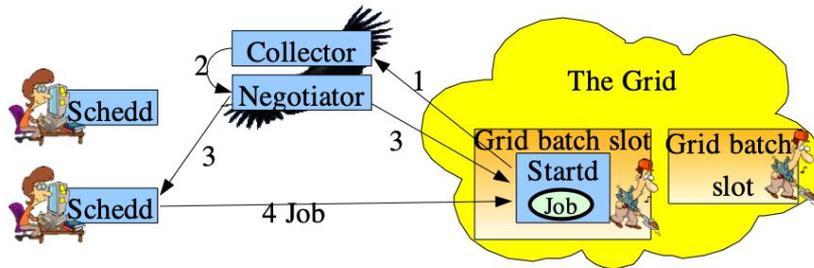


Early concepts for glideinWMS with HTCondor

Around 2007, the idea of the “**pilot job**” coupled with **condor** was introduced for Grid resources. This represented the birth of **glideinWMS**, started to be used by diverse experiments in the US, including CMS.

- **Advantages** already identified: “**Late binding, Reliability, Grid-wide fair share**”
- Logic based on “*Maximize the amount of user jobs, while **minimizing the amount of wasted resources***”
- *The system has been designed to be **highly scalable***
 - “*Only two pieces that cannot be replicated; the **Condor central manager** and the **WMS collector**, and both are relatively **lightly loaded**. Any other daemon (...) can be split (...) ts load distributed.*”(I. Sfiligoi, CHEP07, [ref](#)).

A Condor glidein is simply a startd started as a Grid job. Once the startd registers with the collector, a schedd can send a job as it would do in a local pool, as shown in figure 3.

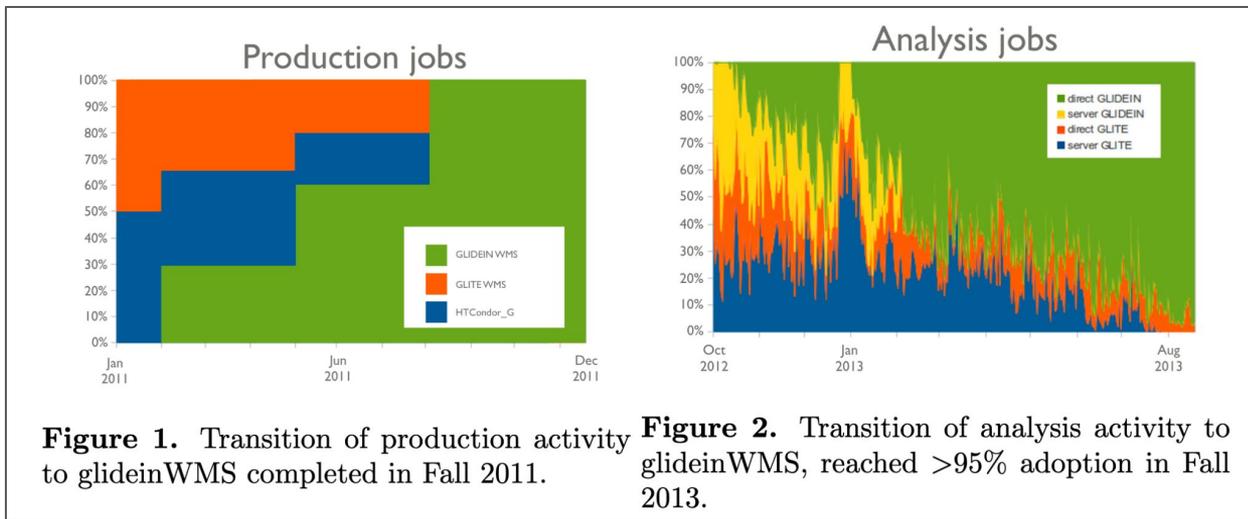




Pilot-based WMS wide adoption by CMS



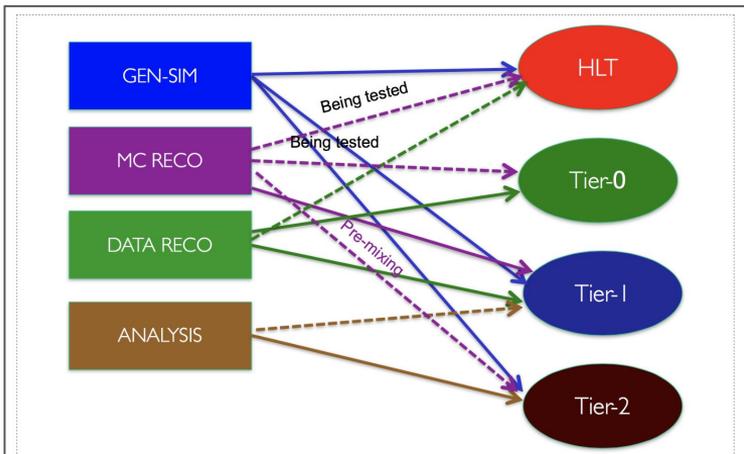
- CMS initially used multiple different job submission solutions:
 - “predominantly in Europe, the *gLite* WMS system, in the USA, the *HTCondor G* system”
 - Both “operating in direct submission mode”.
- However, “during LHC Run 1 (2010-2013), inefficiencies were observed”:
 - Strong dependencies with network and sites status
- Decision was made to move to **unified mode of operations based on pilot jobs: GlideinWMS**
- By **2013** the transition to **model based on pilot jobs** was concluded



S. Belforte et al,
CHEP13 [ref](#)

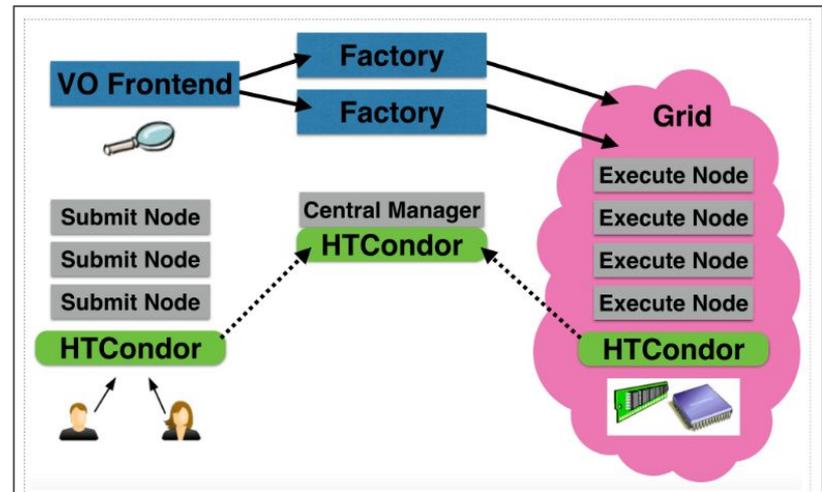
Building the CMS Global Pool

- Even if **CMS** had moved to the pilot job paradigm, a design based on **distinctive pilots per task type** chosen
 - **Separated condor pools** for resources dedicated to production or analysis jobs exclusively,.
 - This followed the then **static** and **hierarchical** distribution of tasks to grid sites (Tier-1s vs Tier-2s), inherited from the old **MONARC** model
- **Before the start of the LHC Run 2 (2015)**, categories started to fade and **sites became multi-role**.
 - After this, *“a unified submission infrastructure was urgently needed”*
 - That would allow *“CMS centrally prioritize between different tasks to make most efficient use of the resources”*
 - **A single Global Pool was adopted.**



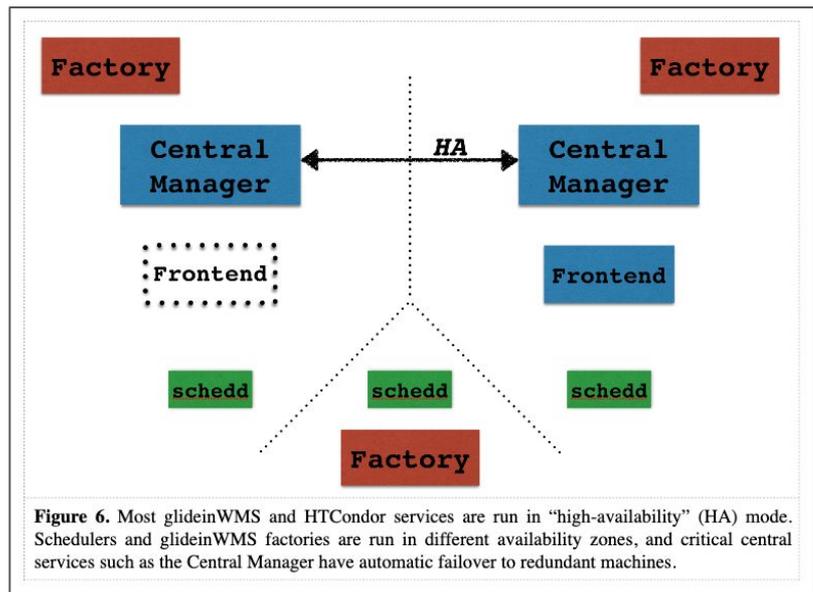
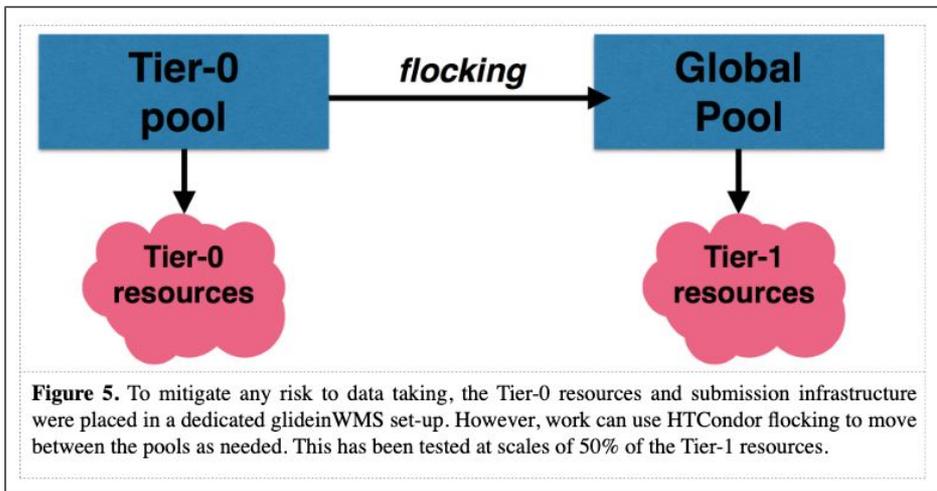
J Balcas et al
CHEP15 [ref](#)

Figure 1. The evolution of the computing model such that different types of workflows can be directed to different sets of resource types, including cloud and opportunistic resources.



Building the CMS Global Pool (II)

- Moreover, CMS model for a **global pool** was conceived with **HA setup** from the start, using the availability zones at **CERN** and **FNAL**
- Given the transition of the Tier-0 to also use Cloud-like resources, and in order to increase robustness of the critical system, **flocking** between Tier-0 into Global pool was introduced

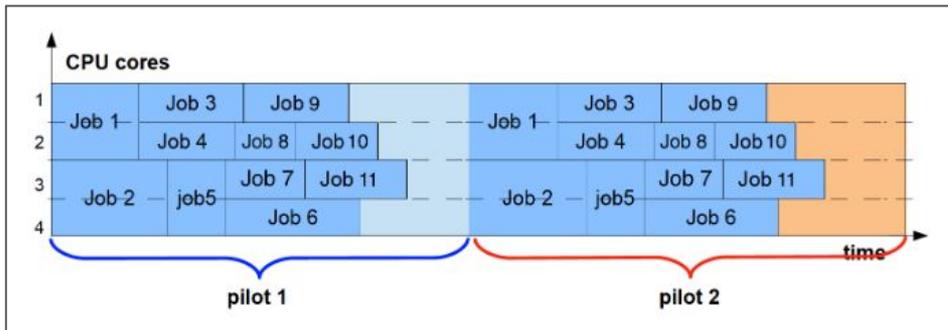




Early multi-core concepts, tests and deployment



- Early work on the use of **multicore slots and jobs** started in preparation for the **LHC Run 2** (CHEP12 [ref](#), CHEP13 [ref](#)).
- Adoption of the **partitionable slot startd** as basis for the new glideins



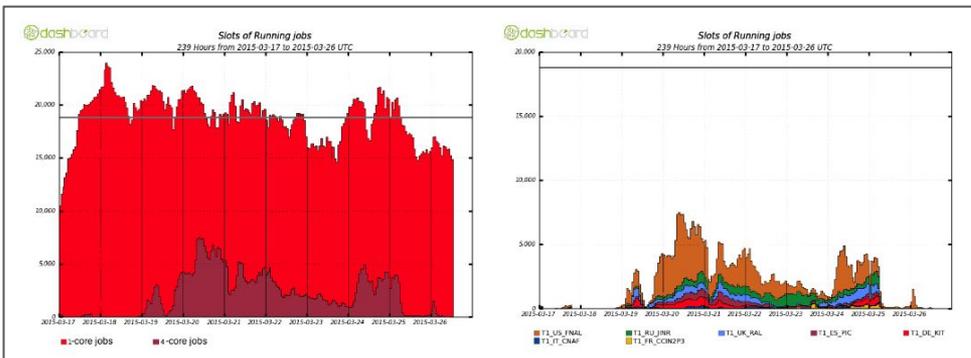
```
- /bin/bash ./glidein_startup.sh -v std -cluster 26592 -name v2_3 -entr
└─ /bin/bash /home/cmprd007/home_cream_114180671/CREAM114180671/glide
└─ /home/cmprd007/home_cream_114180671/CREAM114180671/glide_i1tE5V
└─ condor_startd -f
└─ condor_starter -f -a slot1_3 vocms224.cern.ch
└─ /bin/bash /home/cmprd007/home_cream_114180671/CREAM114
└─ stress --cpu 2 --timeout 200
└─ stress --cpu 2 --timeout 200
└─ stress --cpu 2 --timeout 200
└─ condor_starter -f -a slot1_2 vocms224.cern.ch
└─ /bin/bash /home/cmprd007/home_cream_114180671/CREAM114
└─ stress --cpu 1 --timeout 200
└─ stress --cpu 1 --timeout 200
└─ condor_starter -f -a slot1_1 vocms224.cern.ch
└─ /bin/bash /home/cmprd007/home_cream_114180671/CREAM114
└─ stress --cpu 1 --timeout 200
└─ stress --cpu 1 --timeout 200
└─ condor_procd -A /home/cmprd007/home_cream_114180671/CREAM
/home/cmprd007/home_cream_114180671/CREAM114180671/glide_i1tE5V
└─ condor_startd -f
```



Multi-core tests and deployment



- By **early 2015**, CMS started using **multicore pilots at Tier-1s** (CHEP15 [ref](#))
 - Multicore resources **expansion** in CMS Tier-1s in 2015 and 2016 (CHEP16 [ref](#))
- Given the **difficulty to operate in a mixed single-core + multicore pilots** model at the site level, CMS opted for **multicore pilots exclusively**



4-core job scalability test (March'15) involving all CMS T1 sites, reaching about 40% of the total T1 resources (~20k CPU cores)

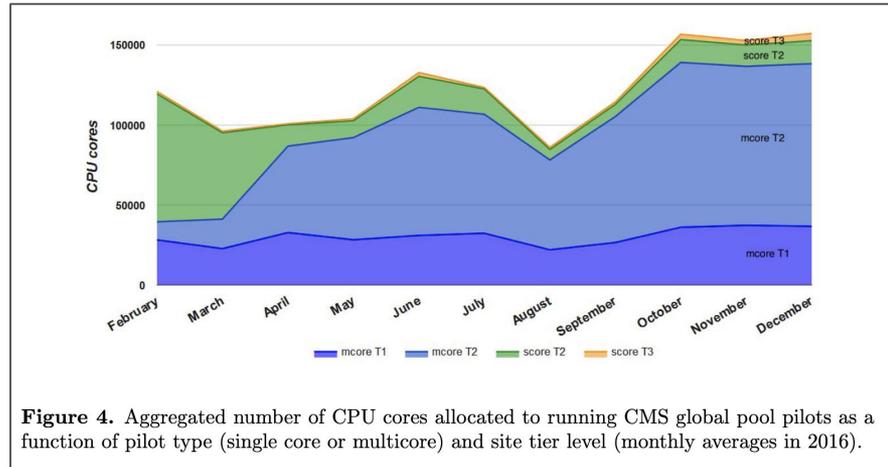


Figure 4. Aggregated number of CPU cores allocated to running CMS global pool pilots as a function of pilot type (single core or multicore) and site tier level (monthly averages in 2016).

Multi-core slot utilization efficiency

- With the wide adoption of the multicore pilot for CMS model, **new sources of inefficiency** were **identified**
- Immediately apparent **need for a more sophisticated monitoring system**, that enabled global evaluation of wasted resources
- Followed by studies on how to **optimize scheduling efficiency** (CHEP18 [ref](#))

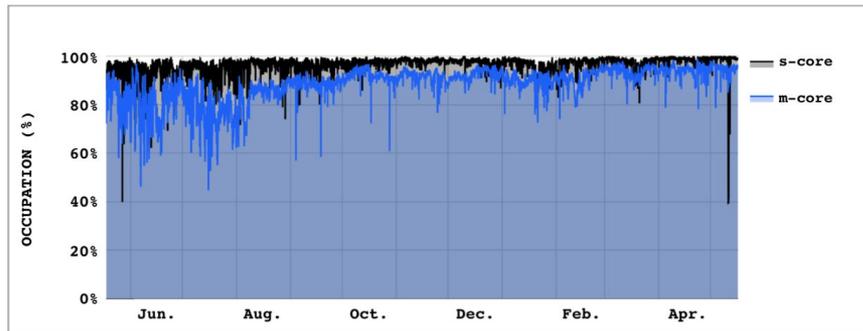
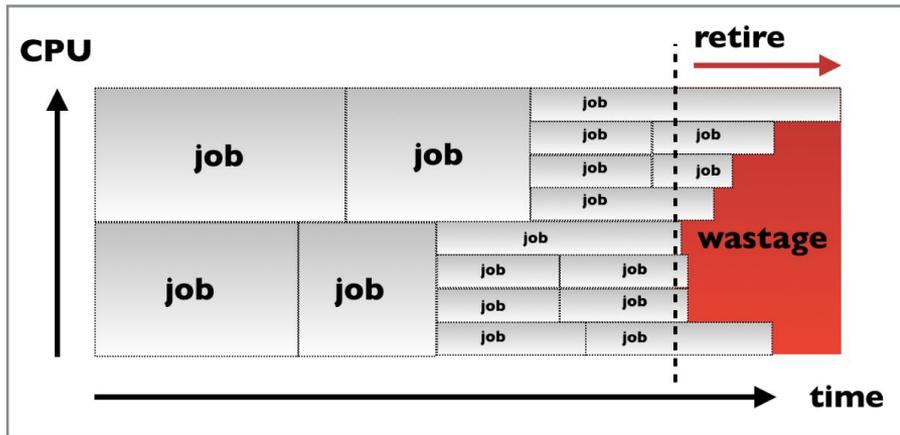
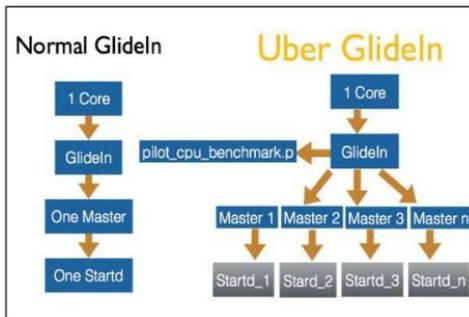


Fig. 3. Improvement of the multi-core CPU scheduling efficiency over time from mid-2017 to mid-2018. Single-core glidein scheduling efficiency is shown as the black, upper line while for multi-core glideins it is the generally lower, blue line.



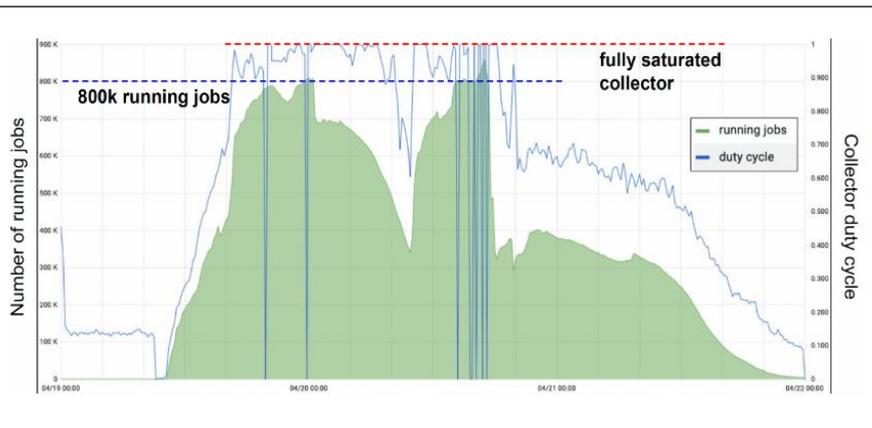
Scalability studies of the CMS Global Pool



Since the start of the Run 2, we have been continuously **detecting and solving bottlenecks to our Global Pool**, with the support of the **HTCondor** and **GlideInWMS** developers teams over the year ([ref](#), [ref](#), [ref](#), [ref](#))

Multiple customized settings:

- 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 (presently disabled)
- Hierarchy of **secondary collectors** connected to the main top collector process
- Optimized **slot update conditions** (filter on update triggers, use UDP vs 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 backup secondary collector (HA infrastructure at FNAL)



Last iteration of the test in **2023** at **800k** running jobs

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



CMS SI deployment with multiple federated pools (~2018)

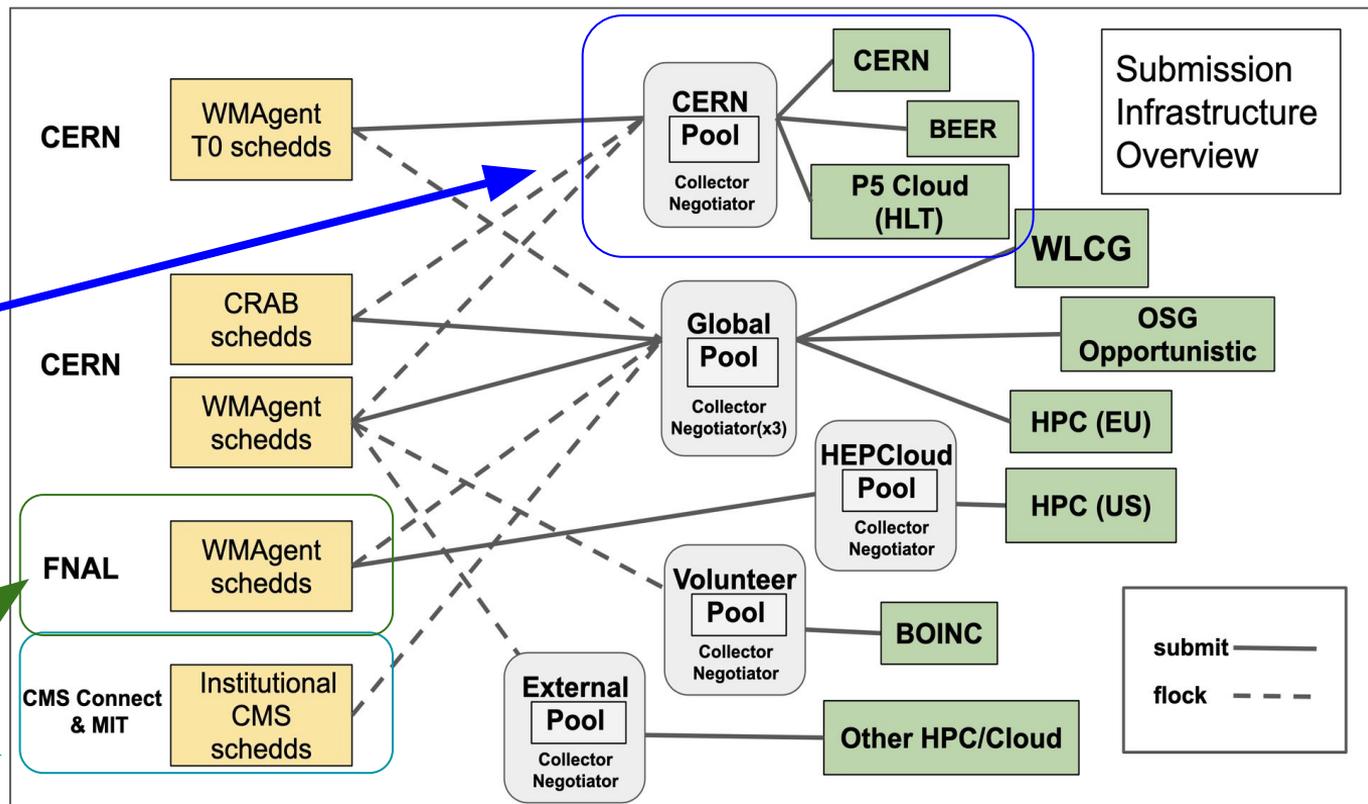


- Types of access point
- Types of execution point

Deployment of a **second "global" pool, merging all CERN resources** (up to then divided as Tier-0 and Tier-2), **federated** with Global, for increased **flexibility** and **use** of all resources

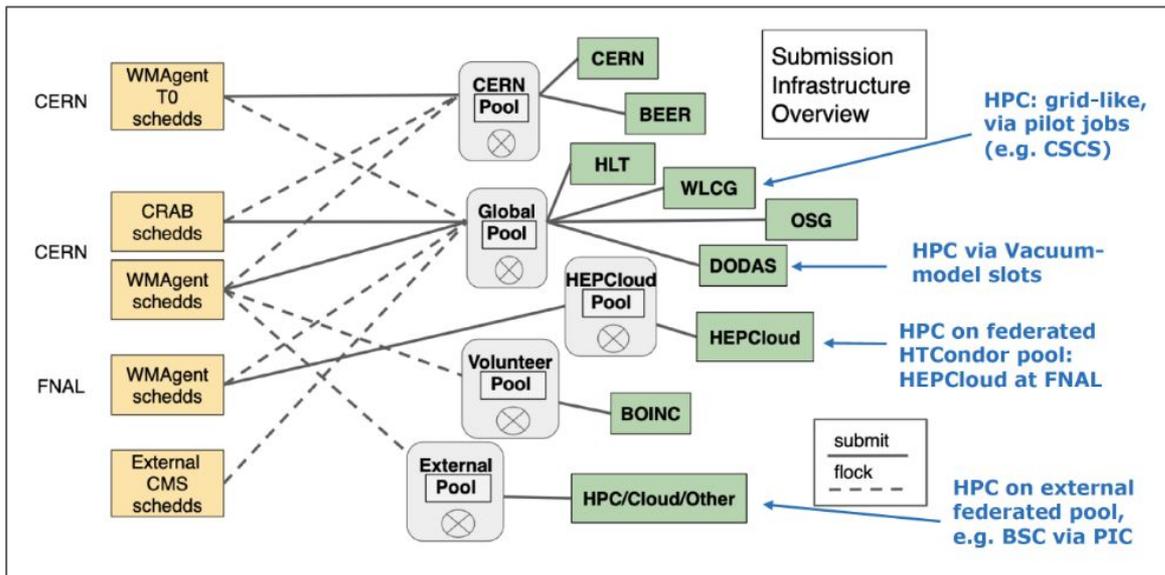
Federated FNAL pool for **HEPCloud** resources

Institutional pools APs federated in similar manner

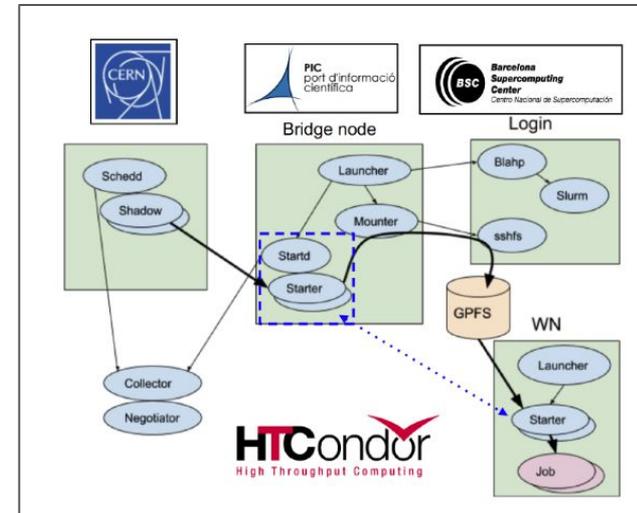


HPC resources integration into CMS model

- Integration of HPC resources into the CMS Submission Infrastructure started around 2019 (CHEP19 [ref](#))
 - Including network-segregated resources with the use of the *split starter* method



The case of BSC (see from CHEP'19 [ref](#))



- Transition to **token-based authentication** in CMS SI happened around 2023 (CHEP23 [ref](#))

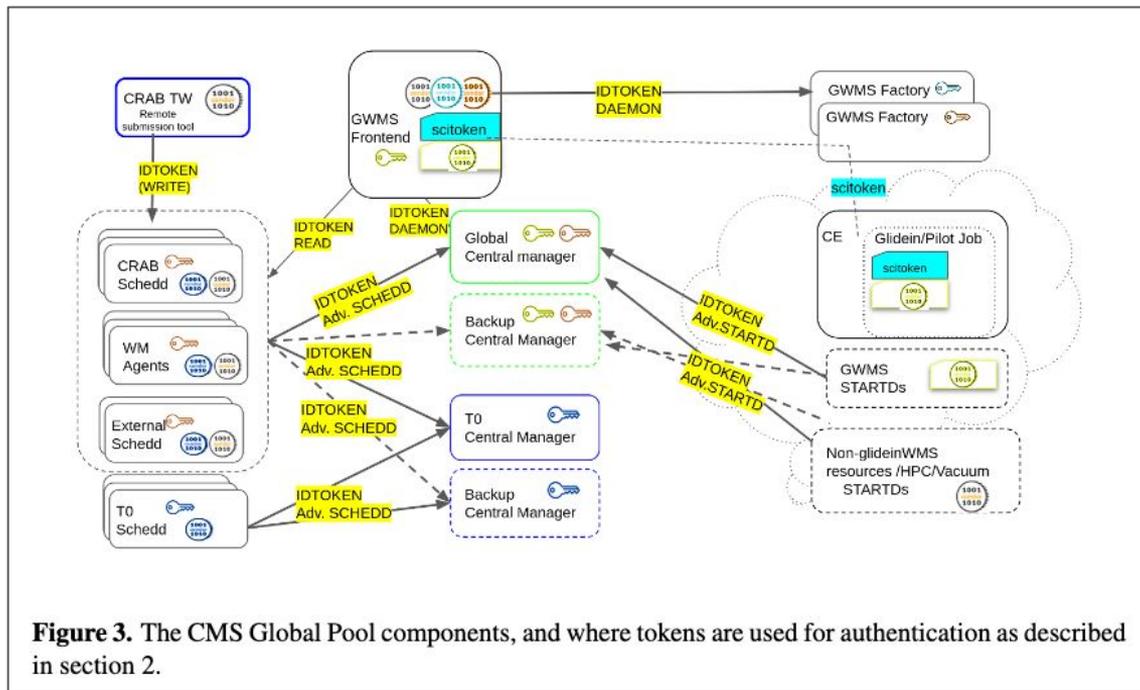


Figure 3. The CMS Global Pool components, and where tokens are used for authentication as described in section 2.



Integration of Heterogeneous resources



- GPU resources integration in the CMS global pool since ~2023 (CHEP23 [ref](#))

CMS SI providing Internal catalogue of GPU slots the Global Pool by scout pilots with HTCondor GPU discovery tool

Heterogeneity demands a much more fine-grained matchmaking based on device properties

Most recently: implementation in CMS WM of a free +REQUIREMENTS string to allow individual users full customization of their resource request

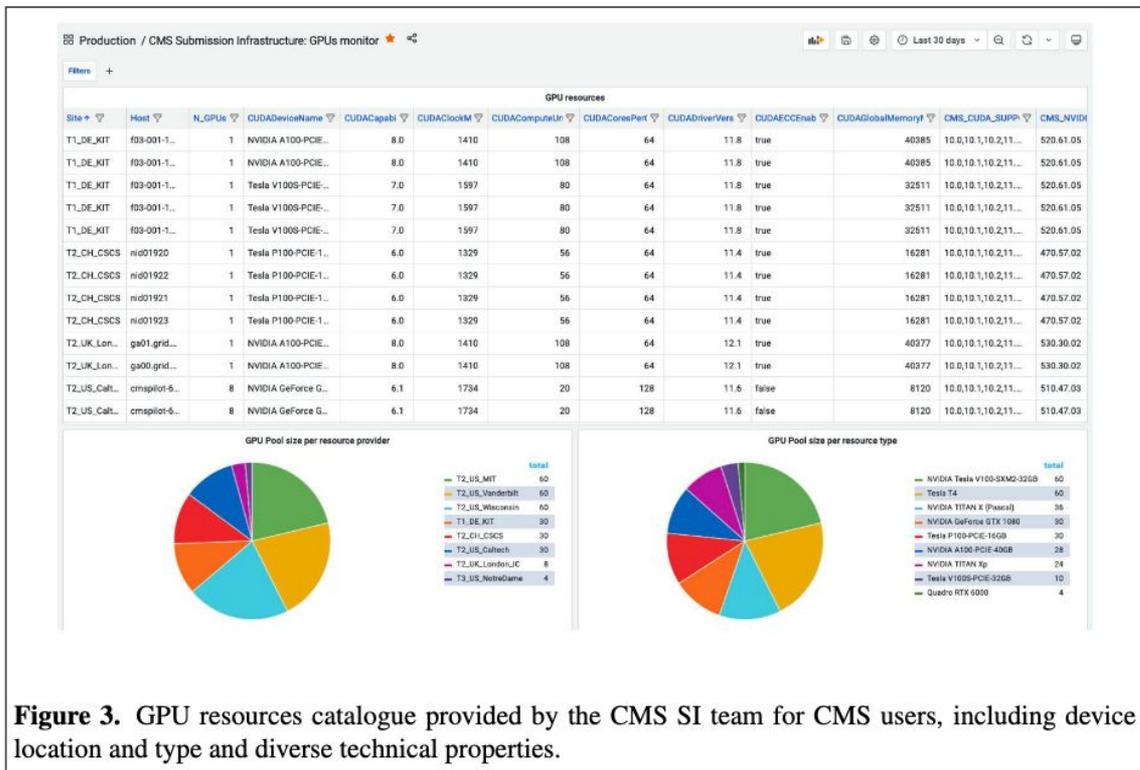
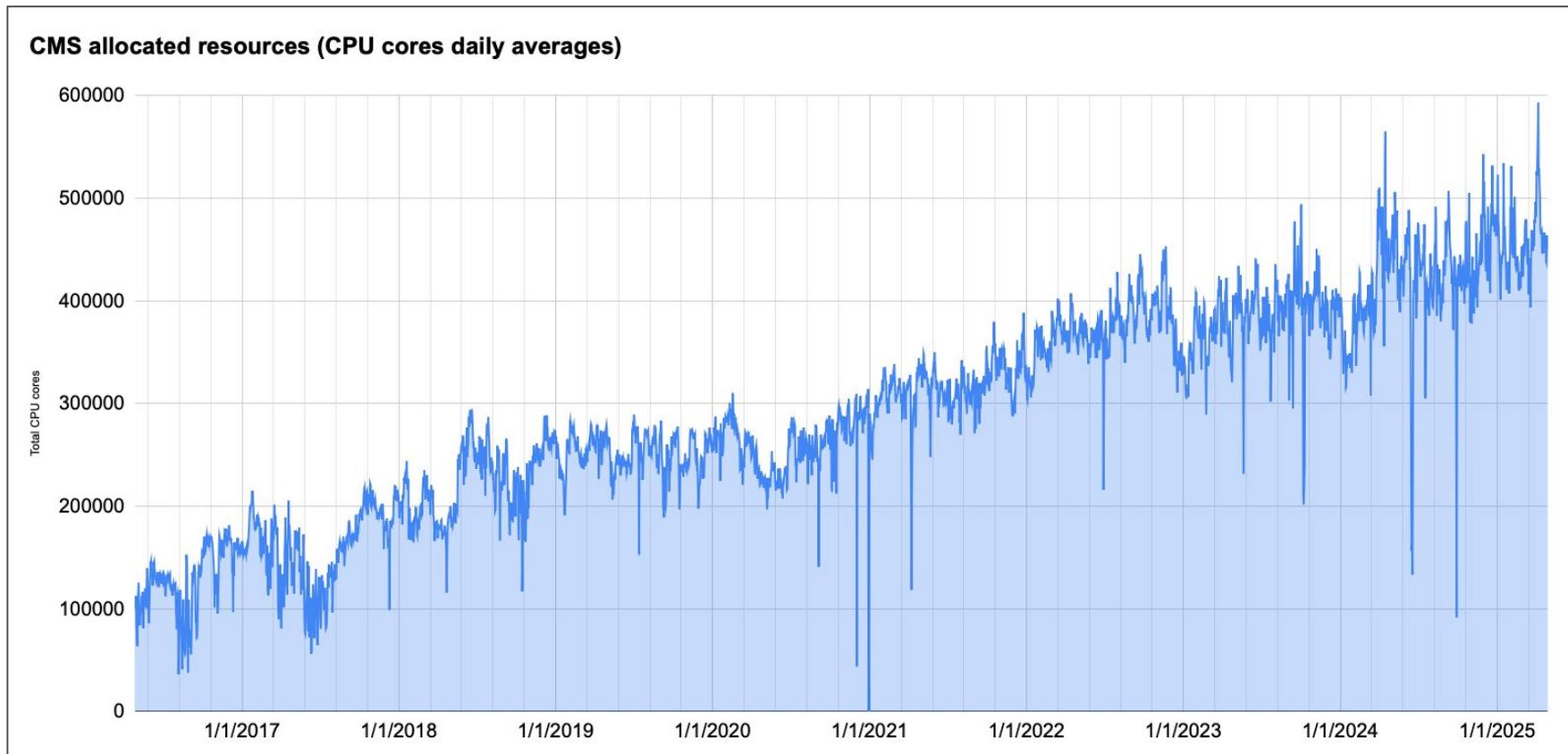


Figure 3. GPU resources catalogue provided by the CMS SI team for CMS users, including device location and type and diverse technical properties.



Evolution of the number of CPU cores managed with HTCondor by CMS since late 2016

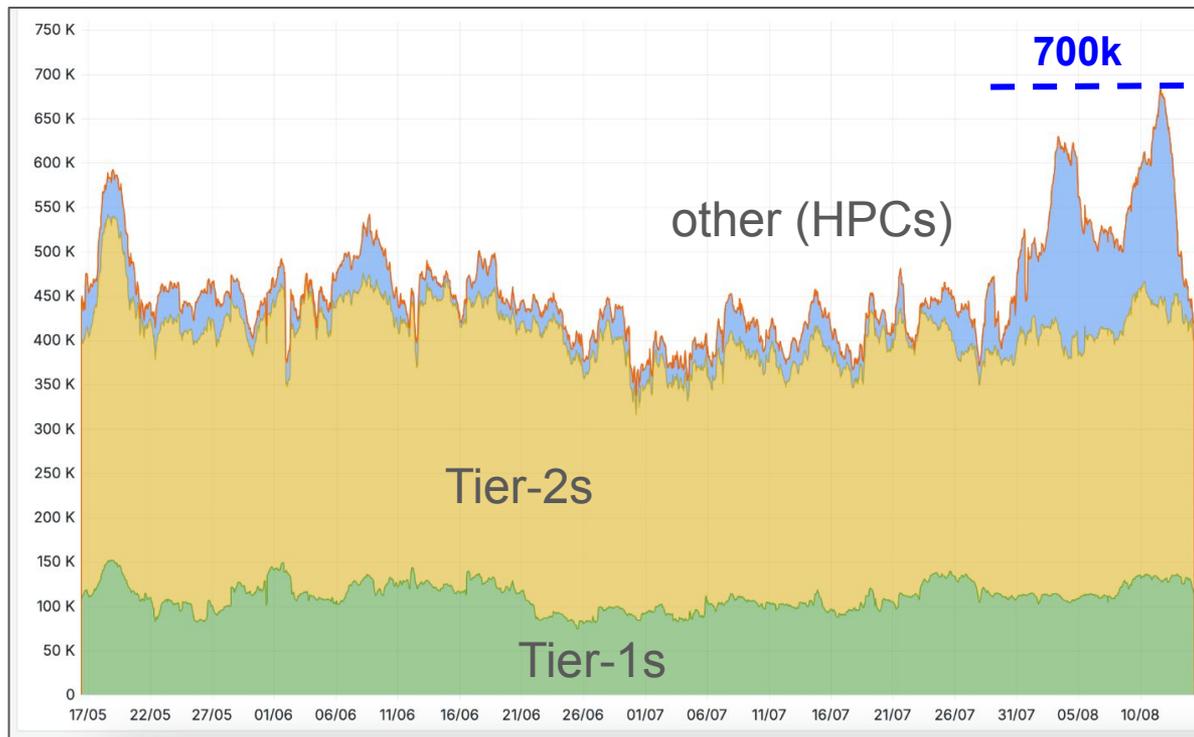




CMS running at full capacity with HPCs



CMS resource record peak managed with HTCondor infrastructure recently achieved at nearly **700k** CPU cores, including HPCs





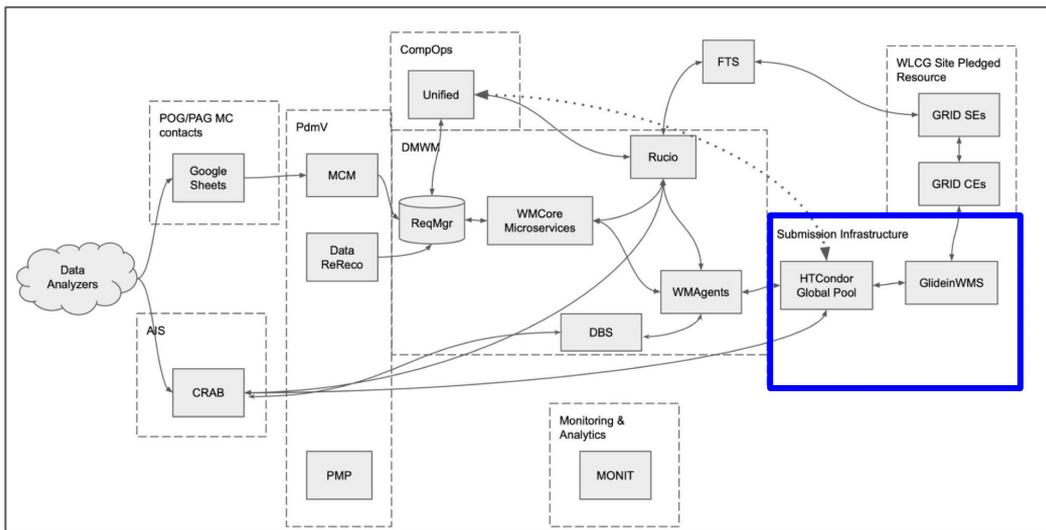
Looking into the future...



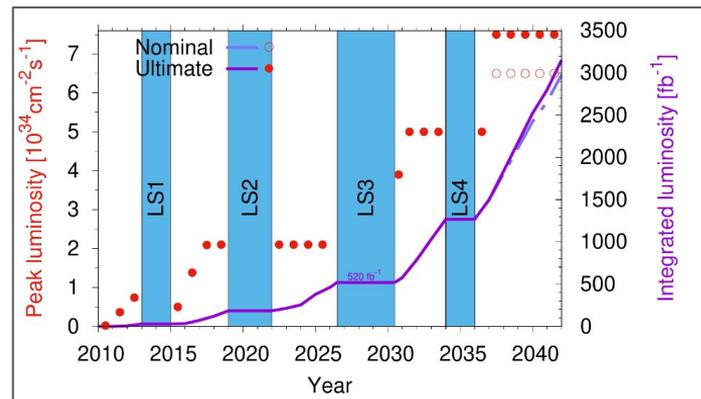
CMS WM Review for HL-LHC (2030...)



- CMS launched internal **review process** for the full WM, which includes the **SI**, thus all of our **HTCondor setup**.
- Find out level of satisfaction and potential limitations with current WM, as CMS moves into the HL-LHC.
- Review conclusions include:
 - **Critical that core strengths** of the current model are **preserved**, including **Late Binding** and **Pilot-Driven Resource Scheduling**, which has enabled CMS simplified resource discovery, whole node scheduling with improved efficiency, being **well-suited to drive CMS into the future with increasing heterogeneous environments**.

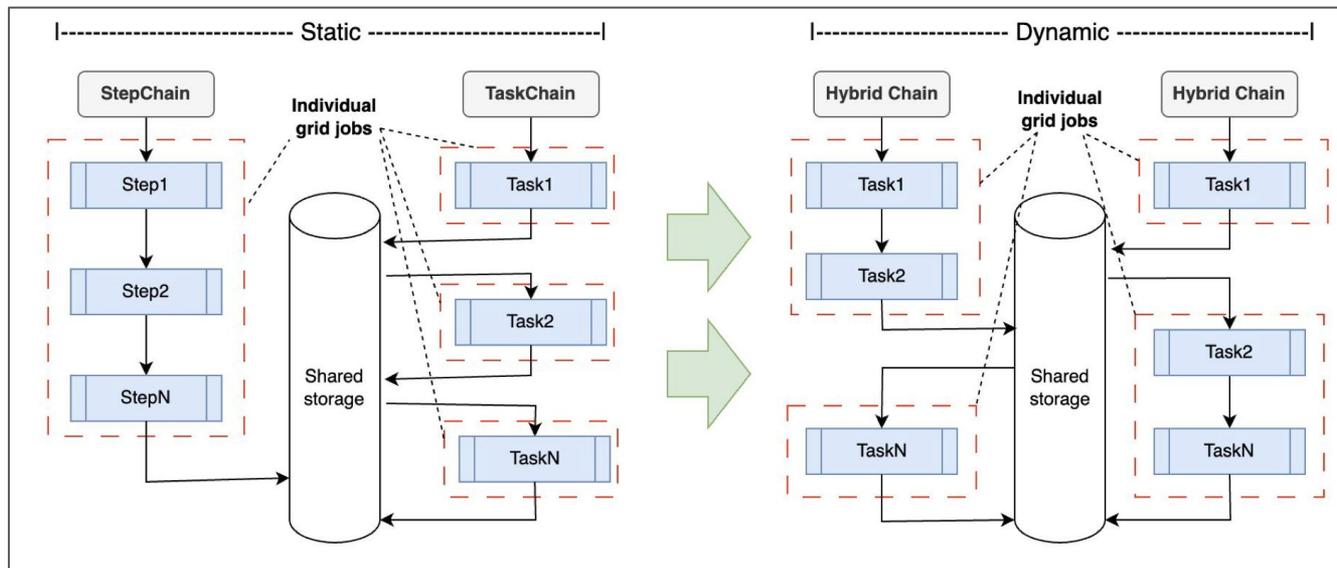


CMS and HTCondor successful story to be continued into the next phase!



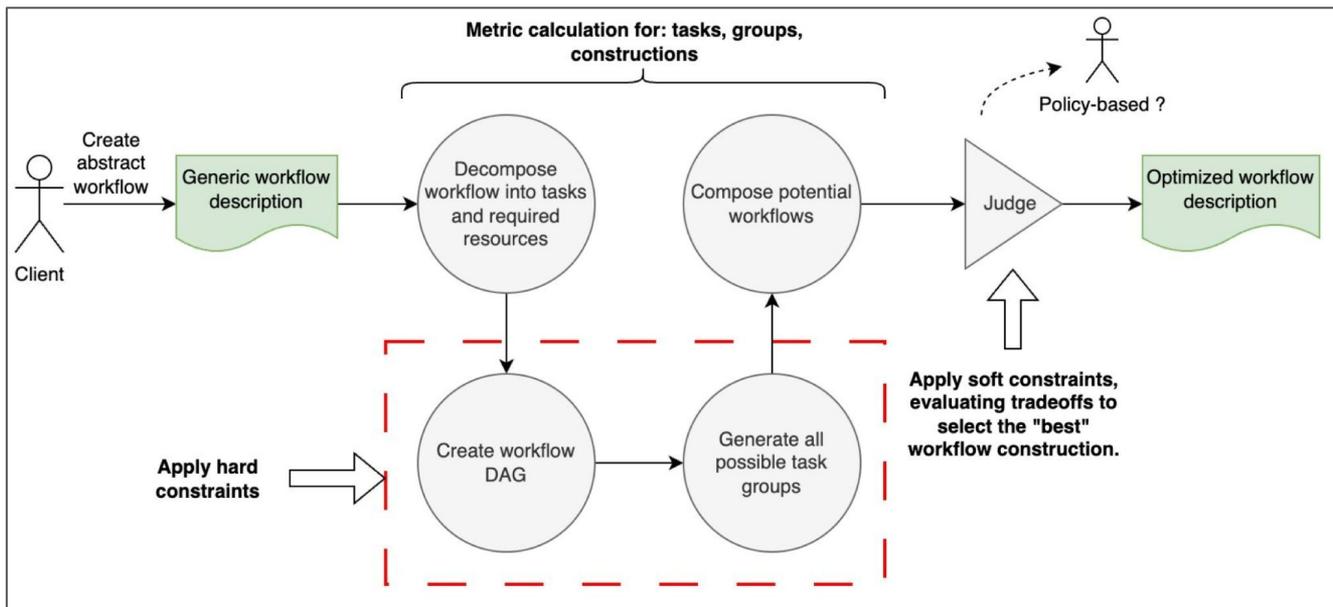
- **Challenge:** how to **group interdependent tasks into jobs** that respect diverse constraints (e.g. CPU and GPU architecture) while **optimizing throughput** and overall resource utilization?
- **Critical** for the success of very large scale workloads in heterogeneous and limited resources

Static to Dynamic Workflow Composition



- Strategy:** start from a **flexible workflow representation**, **grouping** similar tasks together, **evaluate** compatibility with constraints and trade-offs, then **suggest optimized workflow composition**

Workflow Composer Architecture





Conclusions



Conclusions



- CMS relies on a large-scale infrastructure of federated HTCondor pools to execute data processing, simulation, and analysis across 70+ grid sites.
 - **HTCondor** has been for nearly two decades an **extremely reliable suite of technology** that has enabled CMS computing to **successfully support the full physics program of the experiment**, from the preparation phase through Run 3
 - **Key strengths** that our current model provides **must be retained** as we move into the future LHC high luminosity phase
- **Congratulations** from the CMS experiment on the 40 Years of the Condor Project Commemoration!
... And our **deepest gratitude** for the many years of successful collaboration!

Looking forward to many more years of the HTCondor technology in support of CMS scientific programme!

Thanks also to FZU and beautiful Prague!





Thanks!

Project co-funded by the Recovery and Resilience Facility (Spain), and the European Union – NextGenerationEU



Research projects PID2019-110942RB-C21,
PID2020-113807RA-I00, and
PID2022-142604OB-C21 funded by:





Backup slides



Abstract

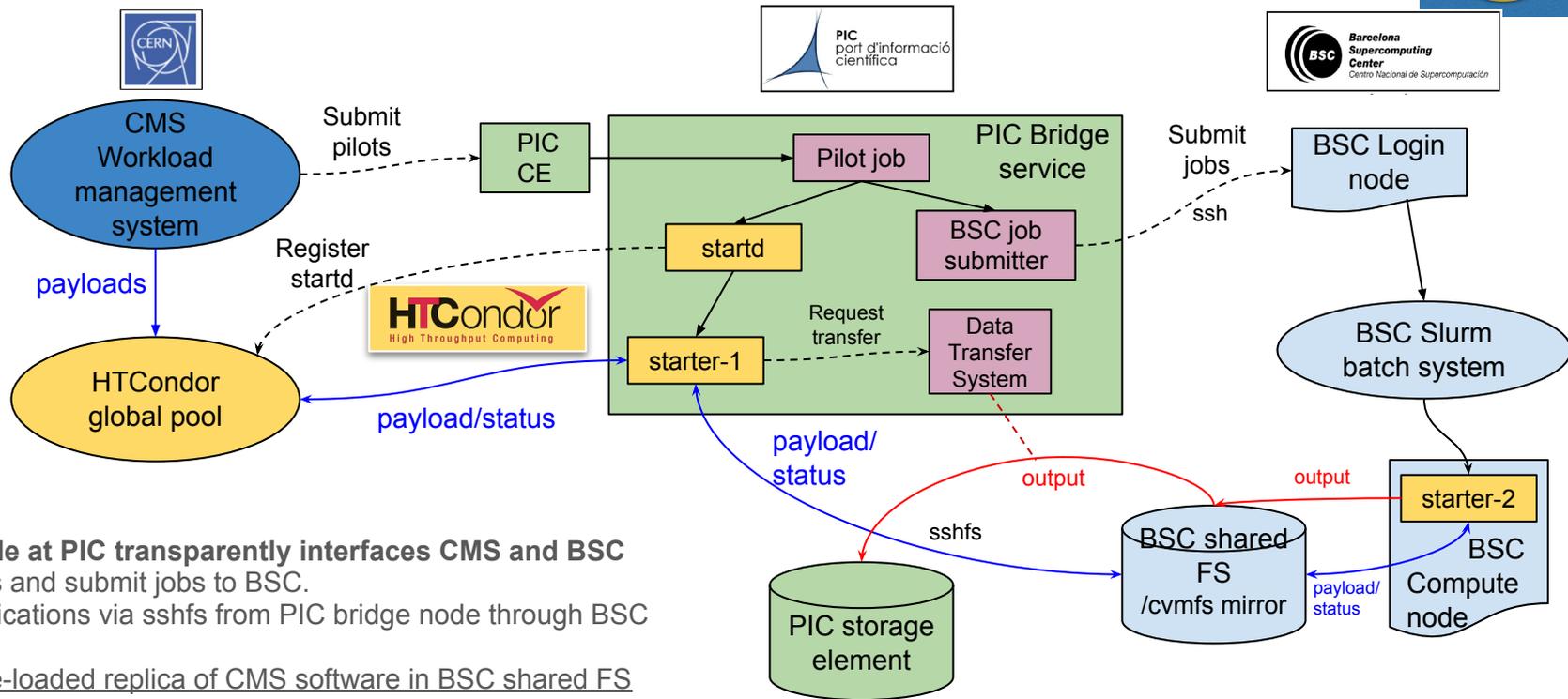


The Compact Muon Solenoid at CERN has successfully leveraged vast amounts of compute resources from a globally distributed infrastructure (primarily the Worldwide LHC Computing Grid) for over two decades, enabling the collaboration to achieve its scientific goals. During this period, a key technology has been the HTCondor Software Suite (HTCSS), which has allowed CMS to manage its High Throughput workloads, an essential capacity for the experiment in order to process, simulate and analyze the enormous datasets produced by the LHC.

In this contribution, in the year when the HTCSS turns 40, we propose an overview of the nearly 20-year shared history of HTCondor and CMS. We highlight how the CMS Submission Infrastructure team has continuously upgraded and adapted our HTCondor setup to cover the increasingly larger, more complex and diverse processing needs of CMS - thanks in no small part to the invaluable support of the HTCondor community throughout the years.



Imaginative HPC integration solutions



- > **Bridge node at PIC transparently interfaces CMS and BSC**
It runs pilots and submit jobs to BSC.
All communications via sshfs from PIC bridge node through BSC login node
- > CVMFS pre-loaded replica of CMS software in BSC shared FS
- > Conditions data accessed via double reverse ssh tunnels
- > Custom data transfer service in bridge node at PIC to move output data files to CMS storage



HTCondor at CMS AF example: CIEMAT

