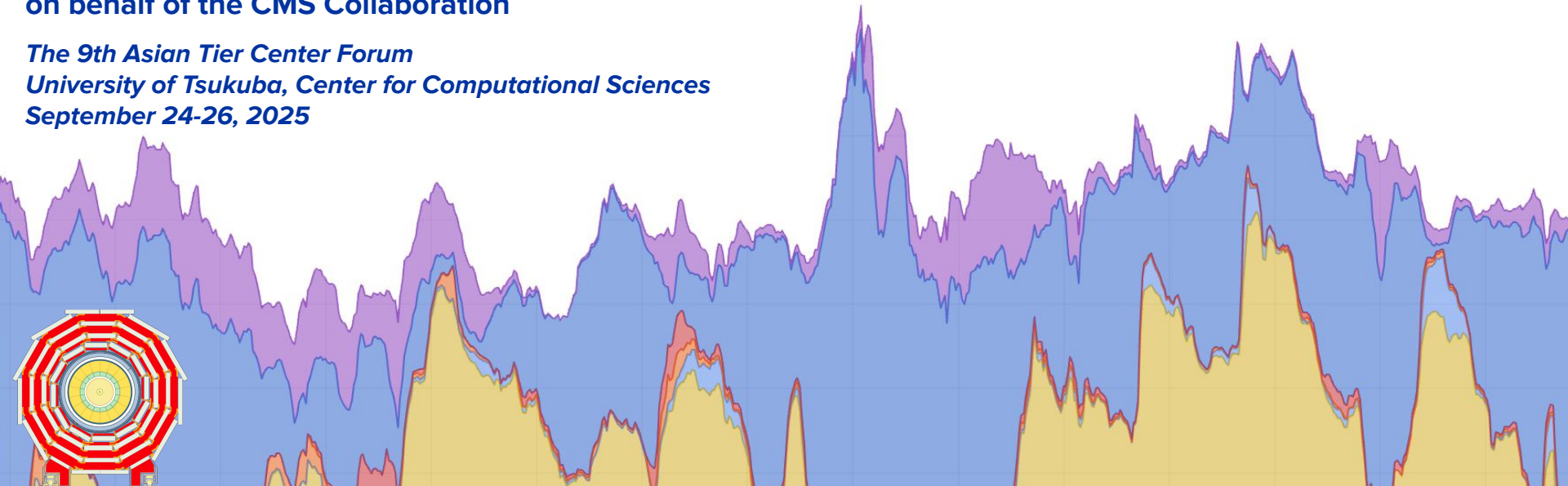


# HPC @ CMS

**Phat SRIMANOBHAS (Chulalongkorn U., CERN)**  
**Credit: Daniele Spiga (INFN)**  
on behalf of the CMS Collaboration

*The 9th Asian Tier Center Forum*  
*University of Tsukuba, Center for Computational Sciences*  
*September 24-26, 2025*



# Outline of Topics

- CMS computing activities
- Opportunistic resources
- What we run: CMSSW application
- HPC@CMS
  - Challenges
  - Distinct strategies for HPC
- Summary

# CMS computing activities

CMS computing activities are aligned with the LHC schedule. Monte Carlo sample production is continuous and driven by analysis needs.

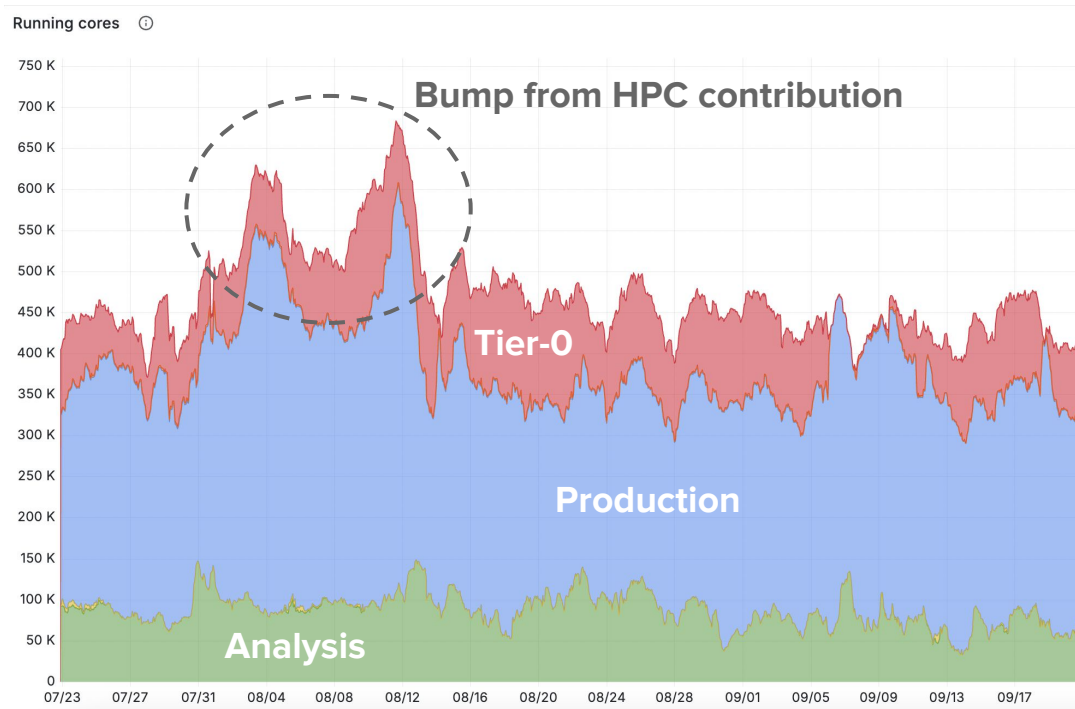
The main activities are:

- Collision data processing and reprocessing
- Monte Carlo sample production
- Analysis (about 30% of WLCG pledged resource)

With additional resources, CMS can:

- Generate larger Monte Carlo samples
- Process parked data earlier

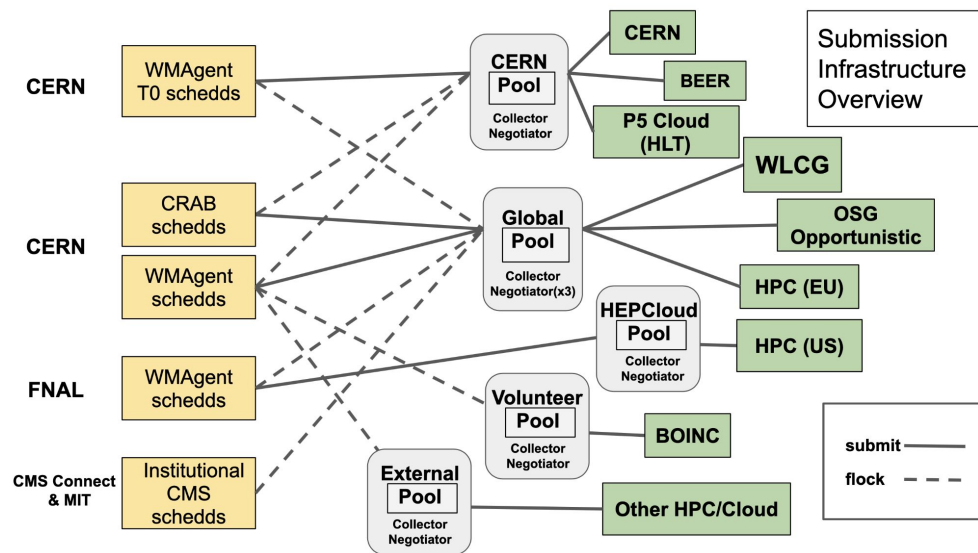
***How does CMS find additional resources?***



# Opportunistic resources

- In addition to pledged resources, CMS makes use of multiple opportunistic CPU sources to support central production. In 2024, these resources was about 36% of total resource.

- Cloud - HLT farm(s):** Currently, Run 2 HLT farm is used for prompt processing and MC production.
- HPCs:** *What we will focus today.*
- Grid:** Usage corresponds to WLCG Tier-1/Tier-2 sites where CMS runs beyond their pledged resources.
- BEER:** Batch on EOS Extra Resources - opportunistic use of EOS storage nodes for batch processing
- Volunteer:** CMS@Home – based on the BOINC middleware framework for volunteer/distributed computing, harnessing CPUs donated worldwide.



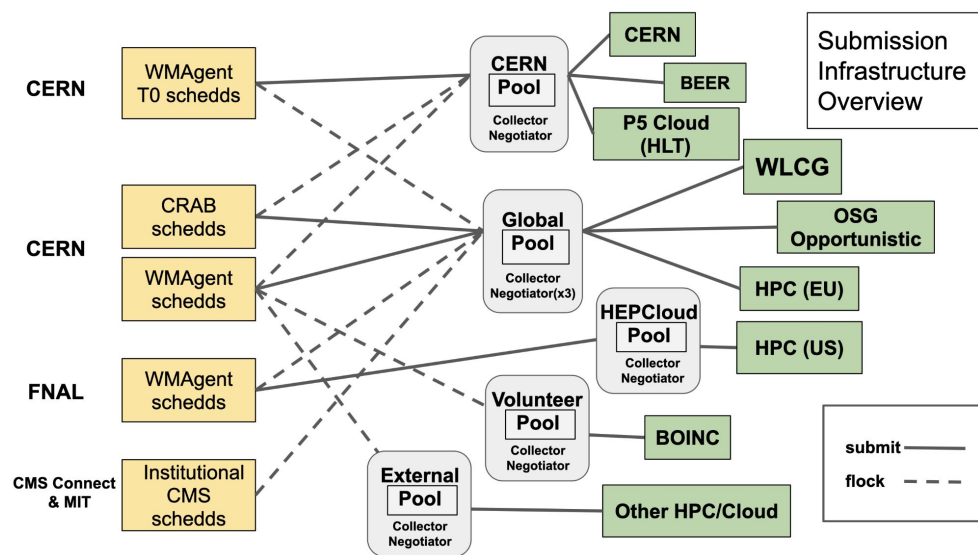
# Opportunistic resources: CMS GlobalPool

- All our integrations have peculiarity in the resource provisioning approach (aka how they start glideins). The common trait is that all them bring resources (slots) into the CMS GlobalPool

- Via regular GlideinWMS push model (i.e via CE)
- Via schedd (Access Point) flocking
- Via manual started glideins at site
- ...

- Regarding the Job slots there is no enforcement on a particular size from the CMS side. We can configure depending on site specific needs/constraints

- Wholenode and then glidein-htcondor does the partitioning based on cores/mem/disc
- Multicore (8 cores) slots



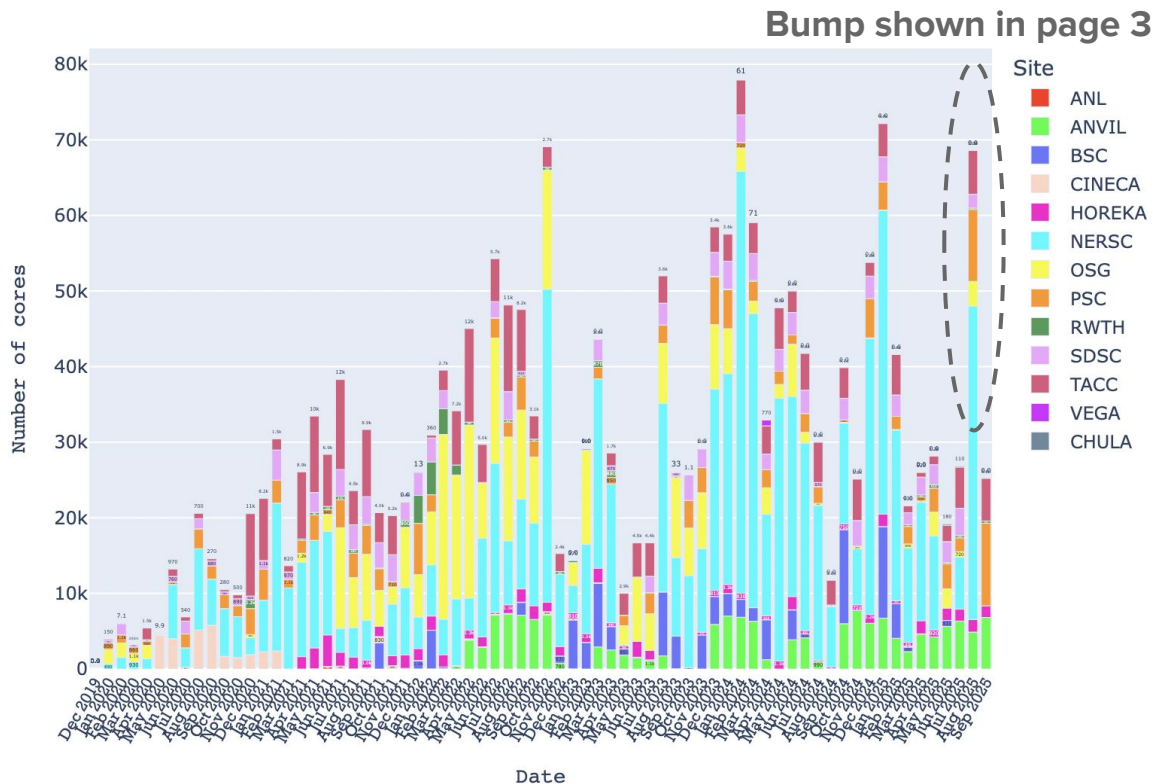
# What we run: CMSSW application

- CMSSW is the CMS data processing software, built around four key components: the framework, Event Data Model, conditions system, and core services (e.g. simulation, reconstruction).
- Over the last decade, CMSSW has evolved from a single-threaded to a fully multithreaded framework. Multi-threaded framework for production started in 2015 as part of the start of the LHC Run 2.
- The main motivation was to reduce per-core memory usage by allowing threads to share common data.
- CMSSW uses Intel's Thread Building Blocks (TBB) for task scheduling and a custom serial task queue to handle shared resources efficiently, avoiding blocking.
- Concurrency now exists at multiple levels: events, luminosity blocks, runs, and algorithms.
- This design enables CMS to exploit modern many-core CPUs (64-128+ cores) effectively, maintaining good scaling while lowering memory requirements.
- ARM is planned to enter production in 2026. In parallel, we expect increasing offload to GPUs in offline reconstruction, following the ongoing developments already deployed at the HLT level.

# HPC@CMS: Overall

- After several years of investment by the collaboration, a number of HPC machines were integrated and continuously used in production since 2020.
- CMS needs to develop distinct strategies to comply with the policies of each HPC system.

Number of running cores - Monthly average [Source]



# HPC@CMS: The (known) challenge

- Integrating HPC resources into the highly automatized processing setups of the CMS experiment requires a number of challenges to be addressed in order to bypass the main HPC design features such as:
  - Relying on accelerators to boost performance (today mostly Nvidia GPUs, tomorrow FPGA, Intel Xe, AMD GPUs)
  - Strict user access policy (i.e. based on ID card)
  - Storage systems optimized for latency and speed, and less for volume
  - Performant node-to-node interconnection
  - Limited capability for data access outside the facility, if not absent
  - Scarce / absent local scratch disk
- Moreover HPC centers differ on a variety of specialized hardware setups, and policy wise strict usage rules can apply mostly for security reasons.
- [CMS-NOTE-2020-002](#) identifies minimal set of requirements on HPC based resources in order to run CMS workflows

Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS devals to solve the no-go
<b>Architecture</b>	Base system architecture	x86_64	x86_64	x86_64 + accelerators (with partial utilization)		Currently, <del>OpenPower</del> , ARM, ... they could be used but at the price of physics validation	QEMU? Recompiling + physics validation?
				<i>ARM is planned for 2026 production</i>			
<b>Memory per Thread/core</b>	Memory available to each thread / process	2 GB/Thread	2 GB/Thread	Down to 0.5 GB/thread needs heavy multithreading, at the expenses of CPU efficiency	GEN and SIM workflows need less than 2 GB/Thread (0.5GB/Thread would be a limit) in order to run efficiently	Less than 0.5 GB/thread	

Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS devals to solve the no-go
<b>I/O</b>	I/O demand per process	5 MB/s/core	5MB/s/core		GEN and SIM workflows are mostly CPU bound still ok with 0.1 MB/s/core	Less than 0.1 MB/s/core	
<b>Local Scratch space</b>	Local space per production job	20 GB/Thread	20 GB/thread local	Less than 20 GB/thread ok if a shared high performance FS is available on all the machines Large multithreading lowers 20 GB/thread requirement to ~ 10	Some CMS workflows run for hours without creating huge local disk areas (GEN, SIM)	No sizeable local space and no shared usable FS	

# HPC@CMS: The (known) challenge

Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS devals to solve the no-go
<b>Outgoing networking</b>	Needed on WNs in order to access remote data, conditions, and to speak to the CMS Global Pool	Full outgoing connectivity	Full outgoing connectivity	Connectivity to only a subset of the IP ranges (for example, to CERN, and to a close xrootd proxy cache) And to everywhere we have condor services?	NAT with a very limited bandwidth via an edge service	No outgoing connectivity from the compute nodes and no NAT available	Edge service running Harvester or HTCondor? Prepare a single container to be deployed at the edge and doing: NAT for Condor, Squid, Xroot proxy cache, ...?
<b>Computing Element</b>	Launch local batch pilots jobs under GWMS factory control	Present locally	Present locally	It can be not local, if a proper network DMZ is set; A remote example is BOSCO. E.g. OSG hosted CE solution.	It can be substituted by VACUUM pilot creations, which is problematically not aware of CMS job pressure		

Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS delevs to solve the no-go
<b>Local batch queue</b>	Distributes the pilots to the local compute nodes, upon submission via the CE	Any supported (HTCondor, LSF, SGE, Slurm, ...)	Any supported (HTCondor, LSF, SGE, Slurm, ...)	Autostart from /etc/rc.local is always possible. It gets very close to the vacuum model	Start via MPI not yet present but could be a possibility		Harvester can help packing jobs into multinode jobs
<b>Operating System</b>	The base Linux system	A Linux derivative of CentOS7	A Linux derivative of CentOS7	<p>If <del>singularity</del> is present, "almost anything" new enough (64 bit)</p> <p><i>apptainer</i></p>	If no singularity/virtu alization AND a strange / not standard FS, the feasibility depends on the quantity of resources (since porting / recompiling is not effortless)	Anything not Linux x86_64 based	Virtualization is an umbrella solution

# HPC@CMS: The (known) challenge

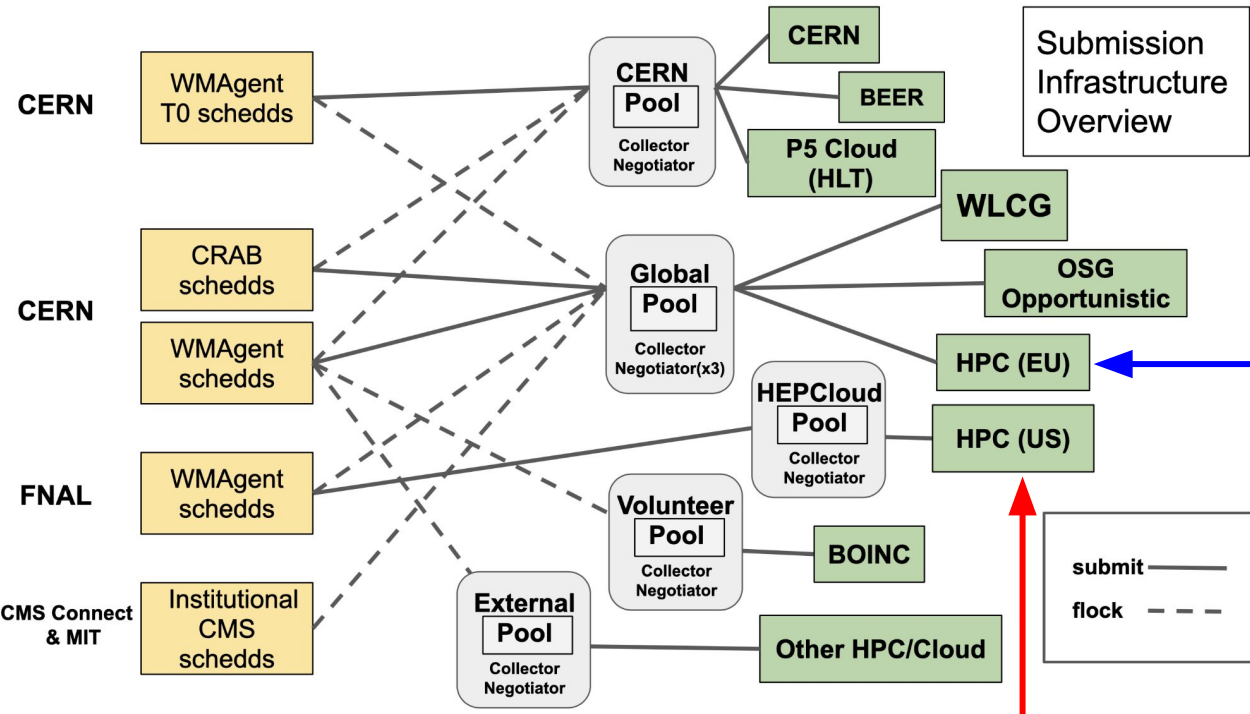
Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS devals to solve the no-go
<b>CVMFS</b>	Used to distribute SW and needed middleware (grid etc)	Available on every compute node, served via a local squid	Available on every compute node, served via a local squid	A single CVMFS server with NFS export should work. Not clear the number of Compute Nodes it can support	Having containers containing full CMSSW releases is possible, but only a few releases. It means a site could support just a specific workflow. In principle, we could support also "by hand" installation on a local shared area, but again it would support just specific workflows	No CVMFS, no CVMFS in NFS mode, no virtualization , no large local shared area	

# HPC@CMS: The (known) challenge

Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS delevs to solve the no-go
<b>Virtualization</b>	Used to ship middleware and specific packages w/o the intervention of local sysadmins	Singularity requested by CMS from the start of 2018	Singularity	We can certainly use full virtualization (OpenStack, etc), but they need someone to launch them.	Docker should also be possible; with some effort in principle also udocker	No virtualization and a strange operating system	
<b>Length of jobs (on the local batch system)</b>	Time a slot can be held by CMS processes	48 hours	48 hours (or more)	We can adapt to smaller length, but it needs specific jobs in principle. In practice if the tuning is 8 h as of now, even 24 should be good enough. Just, the CPU efficiency can go down due to the inability to fill the latest hours of a slot	If the slots need to be very short (say 1 h), we can send specific workflows, but the overall throughput will not be stellar	Slots available for less than 1 hour	Event service

Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS devals to solve the no-go
<b>Access control / local users</b>	Local users (uid/gid) which are mapped to our workflows	One for the pilot, one for singularity. Add to this 1-2 for SAM and debugging. The pilot will launch processes taking credentials from outside (proxies etc)	One for the pilot, one for singularity. Add to this 1-2 for SAM and debugging	We can live with a single user (the pilot one), most probably		Sites which do not want to accept external users at all	

# HPC@CMS: Distinct strategies for HPC

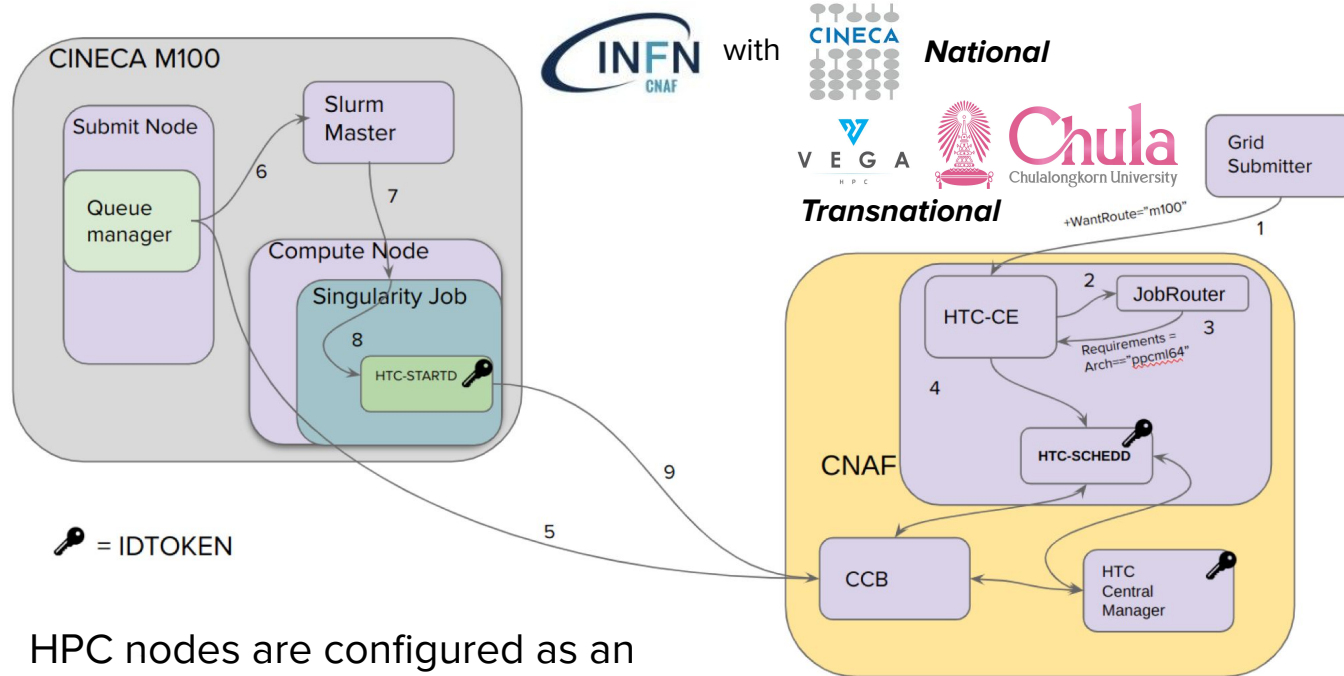


EU HPCs are integrated via the main CMS (Global) HTCondor pool, making them extensions of existing sites at DE, IT, ES.

- Site extension
- HTCondor split-starter
- Overlay batch model

US HPCs are managed from FNAL, via a dedicated HTCondor pool (HEPCloud) which is federated with the main CMS Global pool.

# Site extension + Overlay batch model



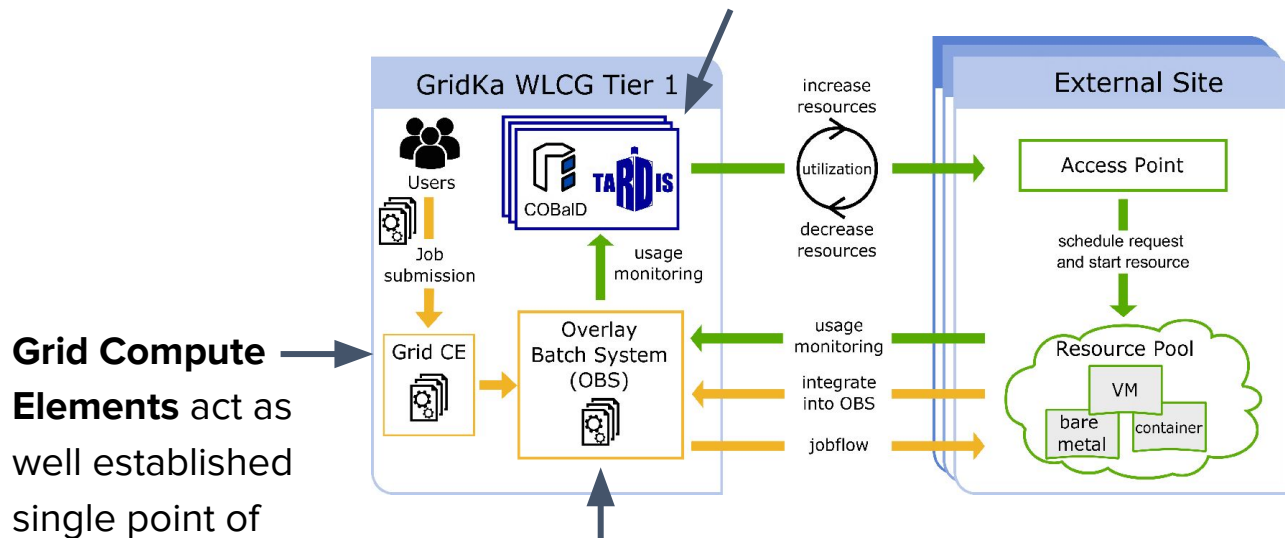
- From HPC Login Node, submit a Slurm job.
- At start, it launches a Singularity container which
  - activates an HTCondor STARTD, which
  - authenticates to the Central Manager and join the pool
  - becomes available to execute jobs submitted to HTCondor-CE at CNAF. StartJobs expression to only accept proper jobs

HPC nodes are configured as an extension of WLCG receiving all the jobs targeted for the standard WLCG site.

# Site extension + Overlay batch model

**COBaID (Cloud Bursting and Demand)**: A Python-based broker that dynamically adjusts resources based on demand.

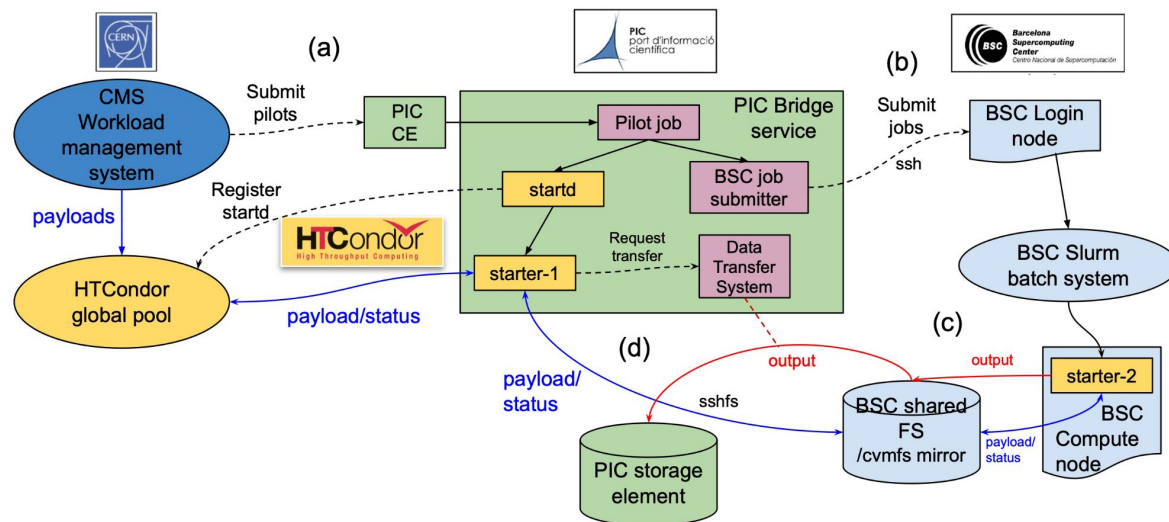
**TARDIS (Transparent Adaptive Resource Dynamic Integration System)**: A backend system that interacts with external resource providers (e.g. HPC centers, cloud sites).



**Grid Compute Elements** act as well established single point of entry.

**Overlay Batch System** provides a single pool of resources hiding complexity.

# HTCondor split-starter



## Technical solutions

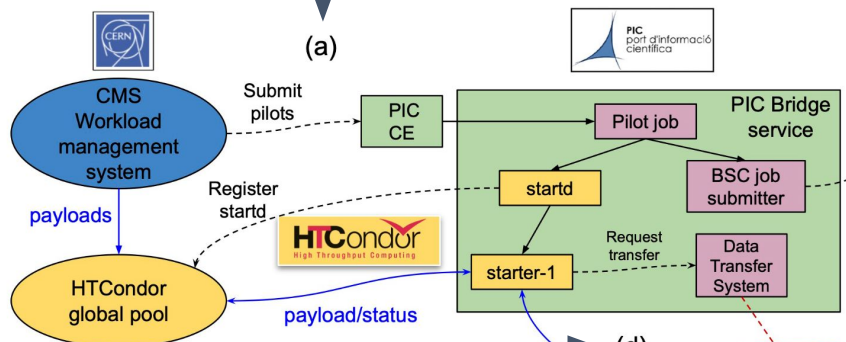
- HTCondor split-starter: use shared filesystem as communication layer for job management
- Software: replicate CVMFS repositories at BSC storage (CMSSW, singularity images, etc)
- Detector conditions data files: pre-place them at BSC
- Develop a service for output data transfer from BSC to PIC (transparent to CMS workflow management)

## Very restrictive network connectivity conditions

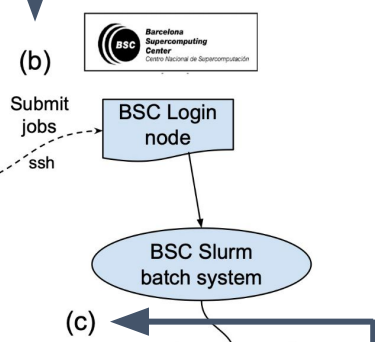
- No incoming or outgoing connectivity from compute nodes
- No services can be deployed on edge/privileged nodes
- Rely on ssh to login node and sshfs to internal shared FS

# HTCondor split-starter

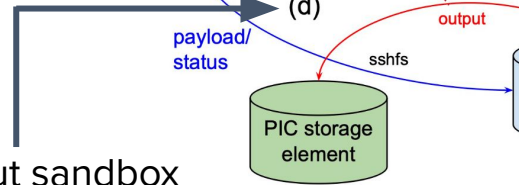
(a) CMS resource requests submission to PIC (pilot job)



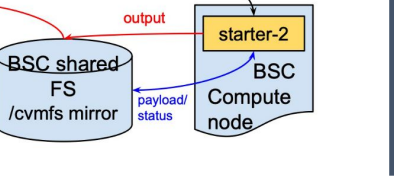
(b) Bridge node acquires resources at BSC for CMS tasks

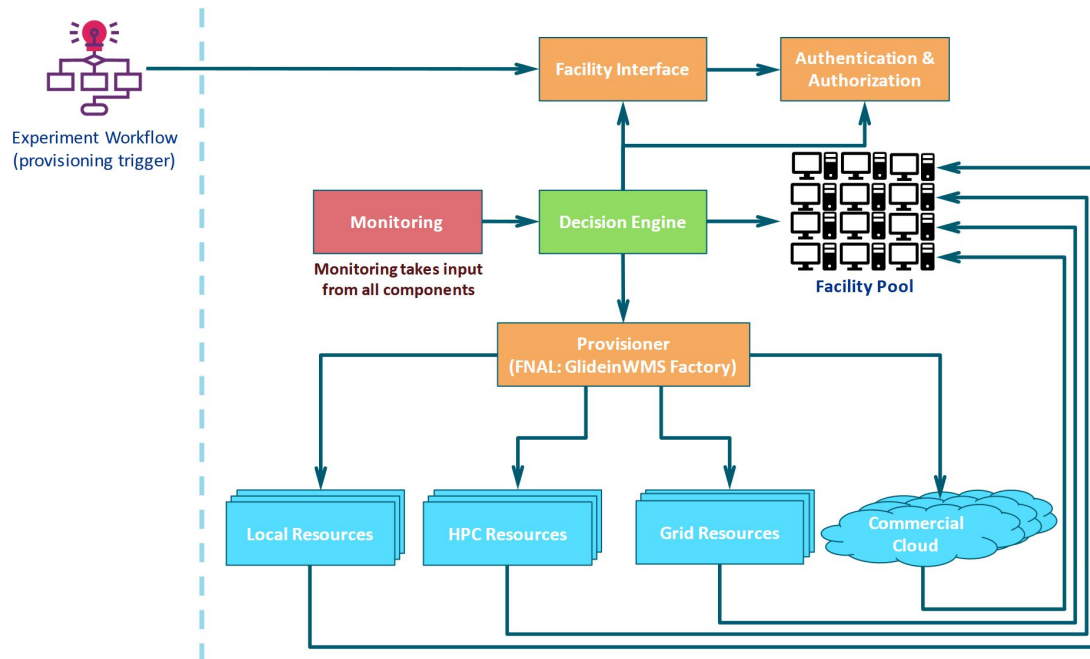


(d) Upon task completion, output sandbox back to CMS schedd, while output dataset copied to PIC storage (DTS acting as third party copy manager)



(c) Input sandbox transferred from CMS schedd to PIC startd, then to BSC starter via sshfs for job execution





- HEPCloud serves as a portal to a wide range of computing resources (both commercial and academic).
- Routes jobs to local or remote resources based on workflow needs, cost, and efficiency.
- Used for integration of CMS computing infrastructure with US HPC resources.
  - Access to CVMFS, and outbound network connectivity from worker nodes
  - Dominate job by MC
  - Read PileUp files via xrootd straight from Fermilab.
  - Remote stageout to FNAL storage

# Summary

- Continuous effort is dedicated to integrating HPC resources, increasing their contribution to CMS computing. These efforts enable CMS to reconstruct 8.8 kHz of data from prompt and parked datasets, and to generate additional Monte Carlo samples.
- Bringing HPC resources into CMS's highly automated workflows requires overcoming several challenges stemming from HPC system design.
- Together, these solutions position CMS strategically to exploit further HPC resources in the future.

# CHEP 2026

X data-intensive interdisciplinary sciences

28th Conference on Computing  
in High Energy  
and Nuclear Physics

25-29 May 2026  
Chulalongkorn University  
Bangkok, Thailand

Registration and abstract submission  
open on 1 October 2025

<https://indico.cern.ch/e/chep2026>

#### Scientific Programme

1. Data and metadata organization, management and access
2. Online and real-time computing
3. Offline data processing
4. Distributed computing
5. Event generation and simulation
6. Software environment and maintainability
7. Computing infrastructure and sustainability
8. Analysis infrastructure, outreach and education
9. Analysis software and workflows

*Other sciences making use of big data applications and large-scale data processing are also welcome.*



- <https://indico.cern.ch/e/chep2026>
- Registration and abstract submission open Oct 1, 2025

