Use of HPC in CMS : CVMFS and other needs

Daniele Spiga -INFN
Credits to: Christoph Wissing - DESY, Dirk Hufnagel- FNAL,
A. Pérez-Calero Yzquierdo - CIEMAT,PIC

TGCC LCG-FR NUMPEX
8th July. 2026

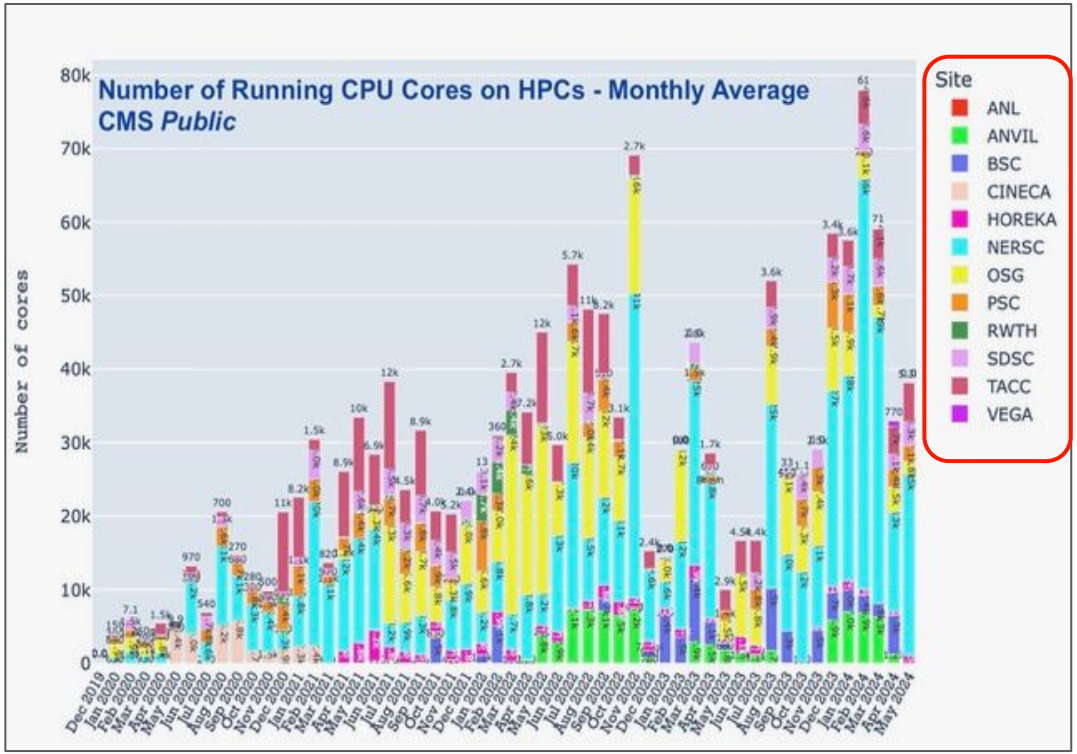
HPC @ CMS

Continuous effort is spent to integrate HPC resources to increase HPC contributions

- CMS wants to further increase the HPC exploitation particularly in the EU zone

CMS Computing / local teams have to provide interfaces between experiment and HPC

- No one-fits-all solution for HPC integration.
- Most HPCs in CMS function as WLCG site extensions



Parts of the pledges are contributed by fully transparently integrated HPCs - i.e. CSCS, CINECA,

The (known) challenges

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)
- 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 because of lack of **outbound networking**
- 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 technical document

Table from an **CMS internal technical document (2019)**, which identifies minimal set of requirements on HPC based resources in order to run CMS workflows.

Link to the document

A synopsis for handshaking with sites

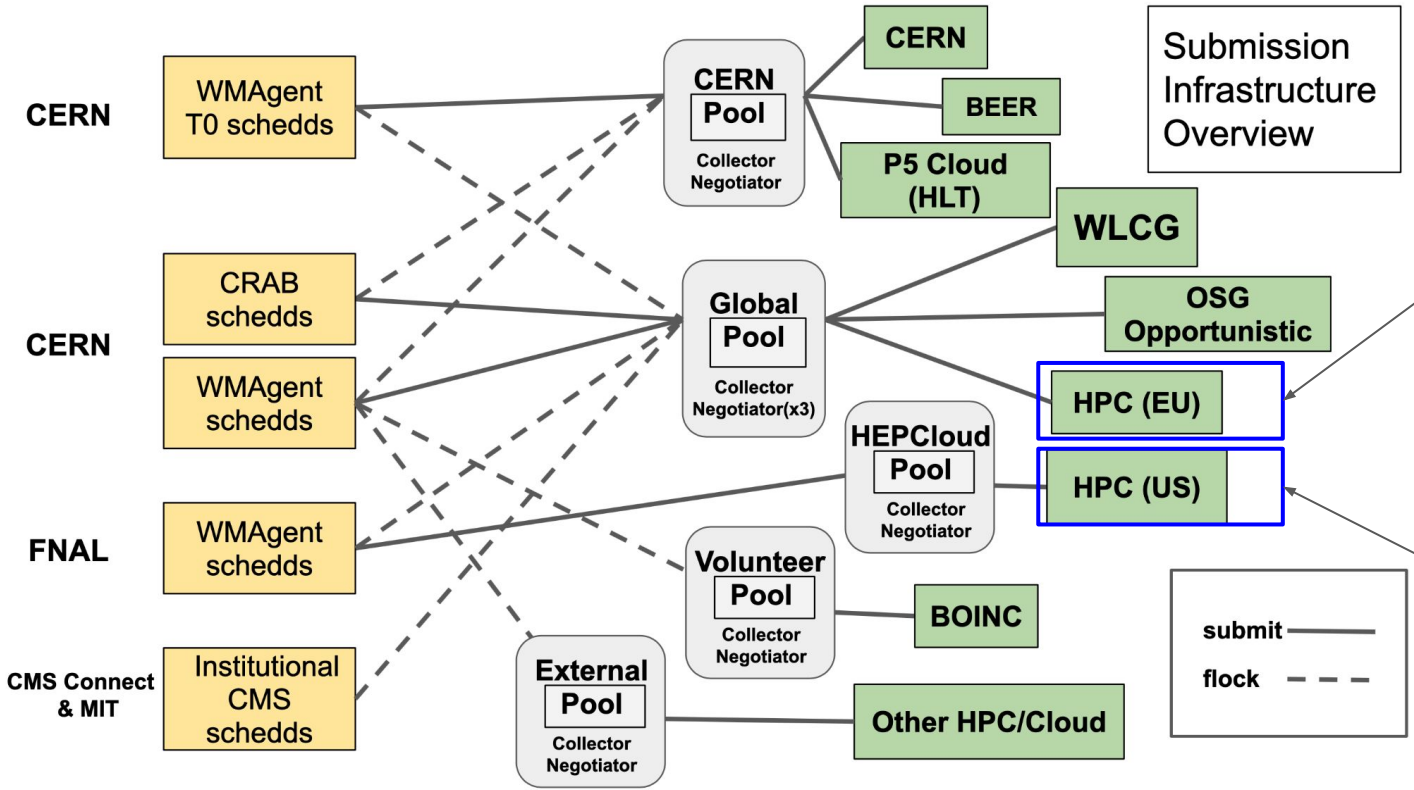
Category	Explanation	CMS standard solution	CMS preferred solution for HPC	CMS fallback workable solution (full utilizability)	CMS fallback solution (for a fraction of workflows)	CMS no-go scenario	Possible CMS devels to solve the no-go
Architecture	Base system architecture	x86_64	x86_64	x86_64 + accelerators (with partial utilization)		Currently, OpenPower, ARM, ... they could be used but at the price of physics validation	QEMU? Recompiling + physics validation?
Memory per Thread/core	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	
I/O	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	
Local Scratch space	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	
Outgoing networking	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, ...?
Computing Element	Launch local batch pilots jobs	Present locally	Present locally	It can be not local, if a proper network	It can be substituted by		

CMS Tech Document
Minimum requirements

- CVMFS installed on the system
- Network routing to CERN and FNAL ***- at least;***
- Singularity/Apptainer deployed;
- ***Highly desirable:*** Edge node i.e. Computing Element (AARC, HTCondor)

	under GWMS factory control			DMZ is set; A remote example is BOSCO. E.g. OSG hosted CE solution.	VACUUM pilot creations, which is problematically not aware of CMS job pressure		
Local batch queue	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
Operating System	The base Linux system	A Linux derivative of CentOS7	A Linux derivative of CentOS7	If singularity is present, "almost anything" new enough (64 bit)	If no singularity/virtualization 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
Virtualization	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	
CVMFS	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	
Length of jobs (on the local batch system)	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	If the slots need to be very short (say 1 h), we can send specific workflows, but the overall	Slots available for less than 1 hour	Event service

General overview of HPC resources integration in CMS



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

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

Glideins and GlobalPool: the common trait

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 HTCondor based GlobalPool**

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

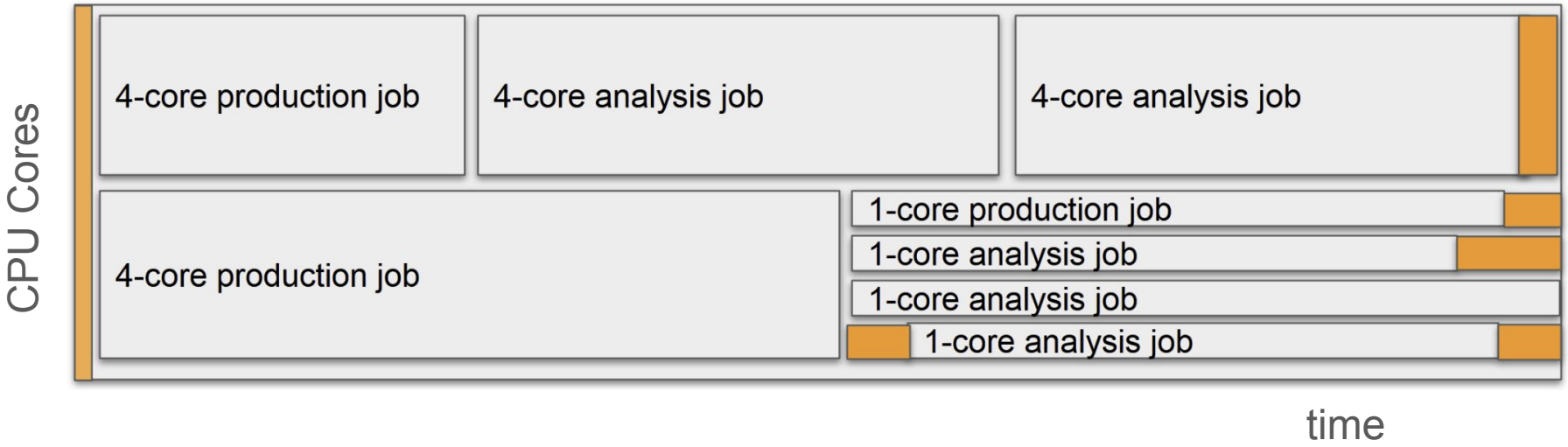
Regarding the Job slots there is no enforcement on a particular size from the CMS side. We can configure depending on site specific needs/constraints

- **Wholenode** and then glidein-htcondor does the partitioning based on cores/mem/disc
- **Multicore** (8 cores) slots, commonly used as WLCG standard.

Multicore pilot model in CMS SI

HTCondor partitionable slots allow CMS to execute multiple payload jobs concurrently and consecutively for the duration of the pilot lifetime.

Scheduling of individual payload jobs into the resource slots is managed by CMS (**late binding**).



Workflow Setup

CMS targets to make the HPC i**ntegration as transparent as possible** and tries to run almost **all production workflows at HPC**, but often we cherry-pick the "easiest" types, i.e. the least demanding in terms of I/O

- (Too much) cherry-picking has a cost on central ops at CMS

As a matter of fact

- We focus on MC workflows (vast majority of our work, StepChains well suited for HPC)
- More relaxed where the HPC integration is implemented as site extension
  - Site extension based integration also means “direct link” with storage at reference CMS Grid Site
  - Custom matching rules (defined at site) to filter out non suitable workflows (“merge” etc)
- For HPC as a separate site (BSC, HEPCloud HPC) CMS controls assignments of workflows
  - BSC limited in what workflows can run (due to the integration constraints)
    - Longterm goal to be able to run ‘everything’ and switch to T1 site extension
  - HEPCloud HPC can run almost all MC workflows (and is normally auto-assigned for them)

Data Delivery

Input data handling

- Mainly rely on AAA, the CMS xrootd federation
  - **Required data products get streamed from remote storage** systems directly to the jobs
  - StepChain (Gen to Reco) MC workflows need to read pileup from remote storage
  - In some case we use **xcache** (through proxy mode) to fan-out
- In cases of a preferred ‘close’ storage site, that is the first try and AAA is fallback
  - Obvious for site extension, but even HEPCloud HPC use this mode (FNAL)

Output data handling

- **Produced output goes to grid storage site**
  - Extended sites relies on storage of the reference site (plus fallback)
  - HEPCloud HPCs in the US all stageout to FNAL directly

Additional considerations

We see the possibility to deploy **edge services as a key element to improve**.
It seems to be more popular now at HPC sites. **These are instrumental and ease the integration with CMS infrastructure**

- (self) deployed “custom” services at the boundary of HPC simplify the integration
  - Some early experiences with edge services to provide cvmfs/frontier at LCF in US
- Allow to standardize
- Open the doors to additional workflow (ML/AI and GPU access in general)
  - Prototyping interLink, a edge service enabling payload offloading i.e. from Analysis Facility at INFN [see CHEP26]

HPC with limited or restricted network won’t support remote data read use case well / at all

- Will need storage allocation at HPC and active data management
- Could also use this for local stageout (remote stageout through edge services also possible)

BACKUP

German HPC Resource Integration using COBaLD/TARDIS

Layers of Abstraction:

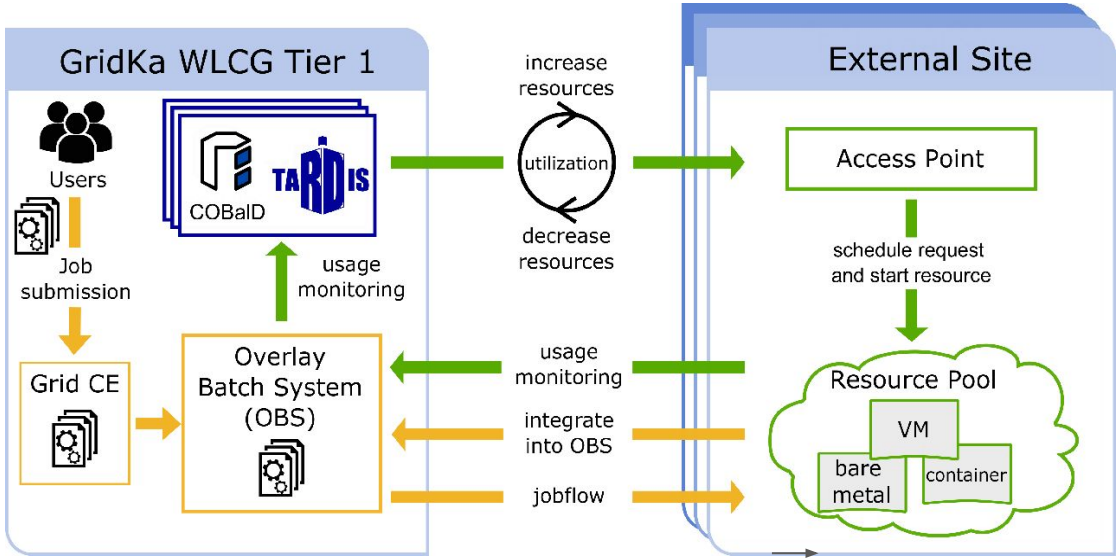
- Grid Compute Elements act as well established single point of entry
- Overlay Batch System provides a single pool of resources hiding complexity

⇒ Transparent & hassle-free access to HPC resources

Resource Integration:

- Utilizing COBaLD/TARDIS resource manager
- Simple feedback based approach: “More used, less unused resources”

⇒ Efficient and dynamic access to HPC resources



Integration of CINECA into CMS Computing

CMS and CINECA were able to agree (2019) on a minimal set of changes to allow for CMS job processing.

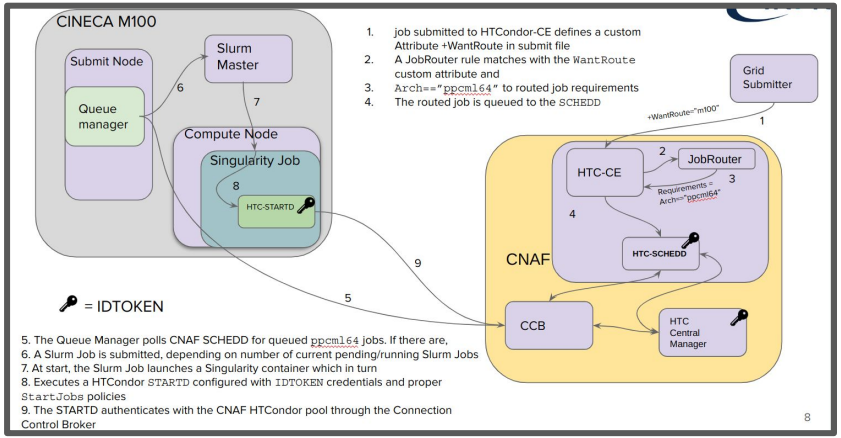
- CVMFS was installed on the systems; Network routing to CERN and CNAF IP ranges activated; Singularity audited and deployed;

CINECA nodes were configured as **an elastic extension of CNAF Tier-1** (SubSite concept) receiving all the jobs targeted for the standard WLCG site.

- Input data access to the AAA Data Federation via xrootd proxy (@CNAF)
- Worker Nodes provisioning: via site launched glidein Relying also on custom matching rules (defined at site)
- A cherry-picking method has been adopted allowing site-level specification of additional requests with respect to CNAF nodes in order to select most suitable workflows.

Same setup used also to integrate VEGA, a transnational site extension

Prototyping also a slightly evolved model transparent T1 batch extension

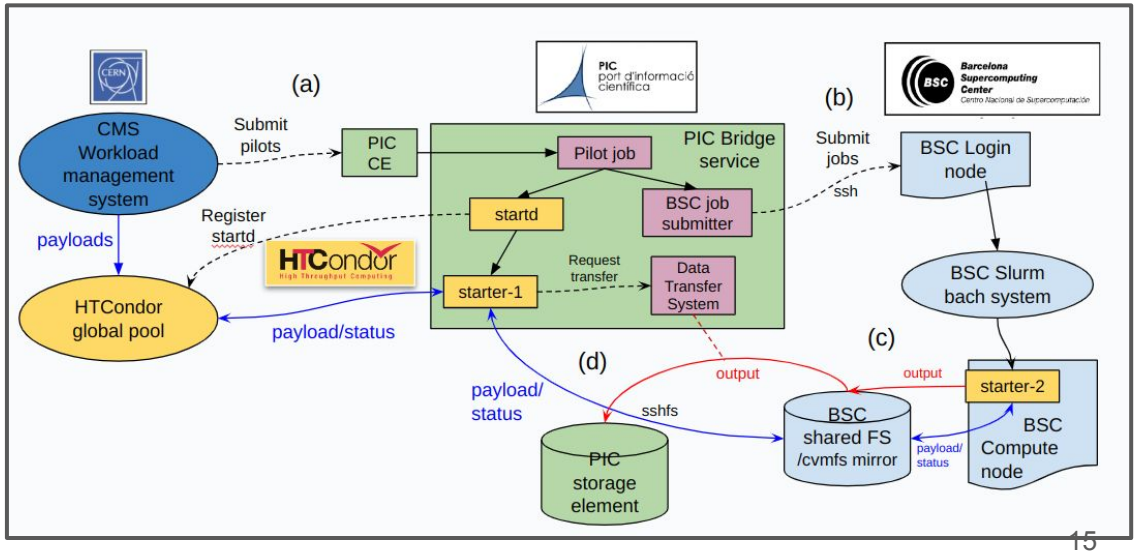


Barcelona Supercomputing Center (BSC)

- **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

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-placing them at BSC not feasible, reverse ssh-tunnels access to PIC squids
- Developed a service for in/output data transfer
- PU datasets need to be manually copied into BSC gpfs in order to run full simulation sequence



FNAL HEPCloud for US HPC

Based on GlideInWMS

(underlying that HTCondor)

Interface to multiple HPC

NERSC, TACC Frontera, ACCESS (PSC, SDSC, Purdue)

Remote ssh submission of pilots

First used with SDSC Gordon in 2014 (pre-HEPCloud)