CMS Strategy for HPC resource exploitation

Antonio Pérez-Calero Yzquierdo
on behalf of the CMS Experiment

CHEP 2019 Adelaide, Australia, November 4th 2019

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
Ciemat Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas
EXCELENCIA MARÍA DE MAEZTU
cfp CIEMAT física de partículas
PIC port d'informació científica

Outline

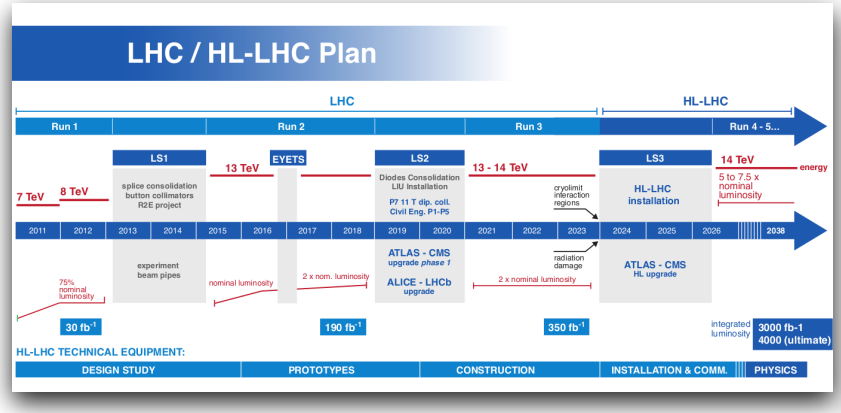
- Motivation for the use of HPC resources by CMS
- Integration effort in CMS Computing
- Challenges towards HPC resource integration
- HEPCloud in the US
- BSC, a hard case example
- Conclusions

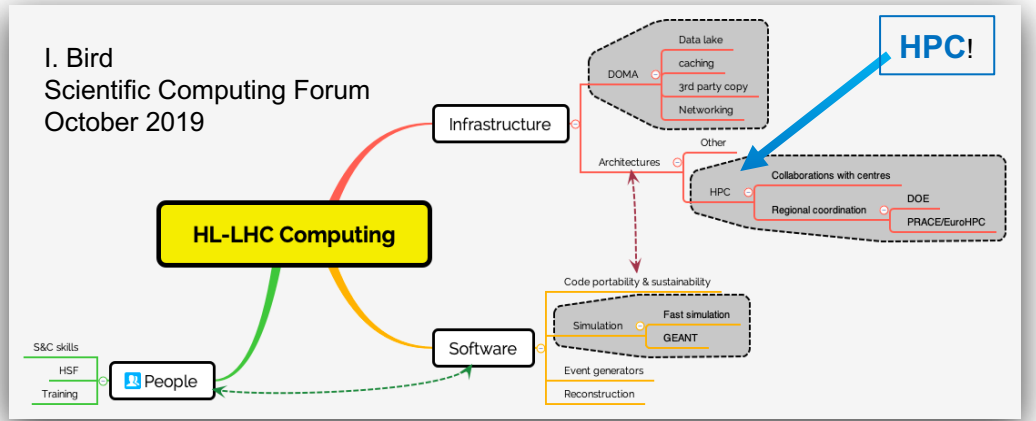
Disclaimer: large subject, will present mainly the headlines, lots of details in the backup slides

Motivation for use of HPC resources

CMS aims towards increasing the usage of HPC resources in the mid to long term future (Run3 & HL-LHC):

- **Growing funding** in HPC infrastructures looking onwards to **deploying Exascale machines**
- **Countries/Funding agencies pushing** HEP communities to **make use of these resources**
  - Opportunity to get HPC managers experienced in providing support in the near future for other data-intensive scientific studies
- Interest in HEP experiments to **access best technologies available**, usually employed at HPC sites
- HPC future contribution regarded as integral part of **WLCG strategy towards HL-LHC**





Integration efforts and access to HPC resources by CMS

- **CMS strategy to approach HPC centers**: **local CMS people handshaking with HPC representatives**
  - To support this approach, a technical document on the CMS requirements to successfully exploit HPC systems has been produced, along with an executive summary addressed at policymakers/FAs

- **Each HPC is different**: Technical and policy questions to be dealt with in each case
  - **Big integration efforts**
  - Technically:
    - Negotiate **deployment of required services** (singularity, CVMFS, data cache services, job gateway (CE), storage access for input/output data files
    - Prepare appropriate application images when no CVMFS or run-time access to conditions data

- **Resource availability**: procedure of calls for resource **allocation** at HPC centers is in general **not suitable** for HEP experiments, including CMS:
  - Often, allocations are granted targeting solving **specific physics studies**, not generic for all experiment's needs
  - Instead, a **stable share of resources at HPCs** is needed to properly plan CMS computing operations
  - Care should be put in **protecting the traditional pledges on Grid/HTC resources,** when discussing use of HPC as pledged resources with FAs

Software development

- **Computing power from HPCs will probably not be provided by CPUs exclusively**
  - CMS **software written for CPUs (Intel x86_64 architecture)**
    - best choice for **affordable computing** in the last decade
  - **Computing power not accessible to CMS**

- Effort within CMS being organized to address this challenge
  - **Performance portability,** re-thinking of reconstruction algorithms and data structures aiming at allowing to run single implementations of algorithms on the largest number types of accelerators as well as on CPUs
    - Investigating a number of performance portability libraries
  - **Support for heterogeneity in the CMSSW framework**

- Testbed available in Run3? CMS aiming at deploying heterogeneous **CPU+GPU HLT farm**

Operational challenges

- Global CMS **operations model** towards stable exploitation of HPC: **transparent and agnostic** with respect to the **type (HTC or HPC) of resources** being used

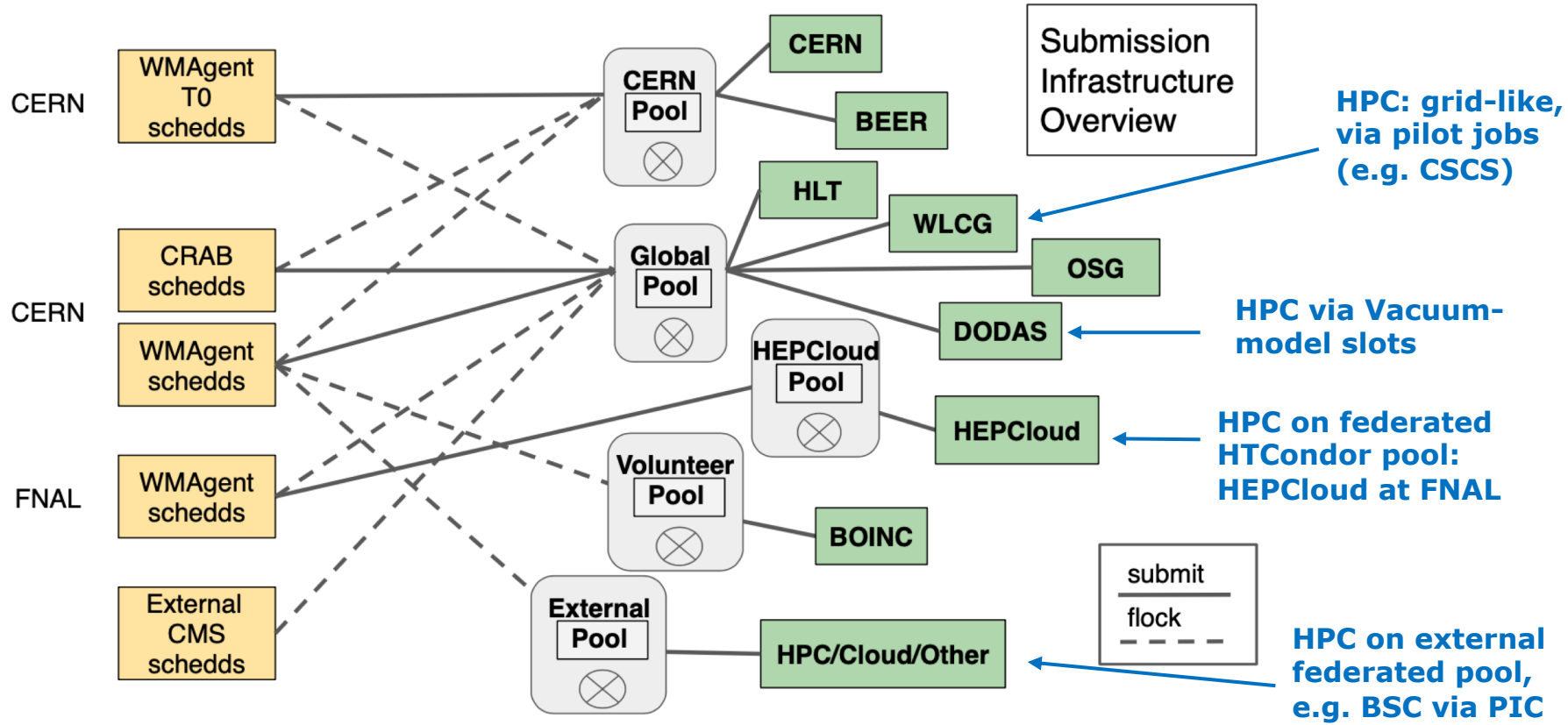
- **Automatically select suitable workflows to execute in each HPC resource**
  - **Characterization of both resources and workflows becomes crucial** to schedule workflows on such special resources (see backup slides for details)
  - Need to describe the capabilities of the resources in our WMS
    - E.g. execution node network connectivity, local pile-up sample availability, internal and external I/O capabilities
  - Define workflows in a accordance to resource capabilities
    - Simulation jobs: easiest workflow would be GEN-SIM, but we are in fact targeting all simulation stages be done in a single workflow (GEN-SIM followed by DIGI-RECO)
    - Job splitting to satisfy certain HPC requirements (e.g. job length)?

Data Management challenge

- Data Management issues when running on HPC systems
  - Generally **not meant for accessing data residing outside the facility**
  - Not designed with **bulk size** and **WAN connectivity** for overall processing throughput in mind

- Managing input/output files: twofold default solution conceived:
  - **Remote I/O via xrootd** with "nearby" WLCG site when suitable WAN available
  - **Prestaging data** via file management using PhEDEx/Rucio when HPC site provides http/gridFTP/xrootd door and enough local storage

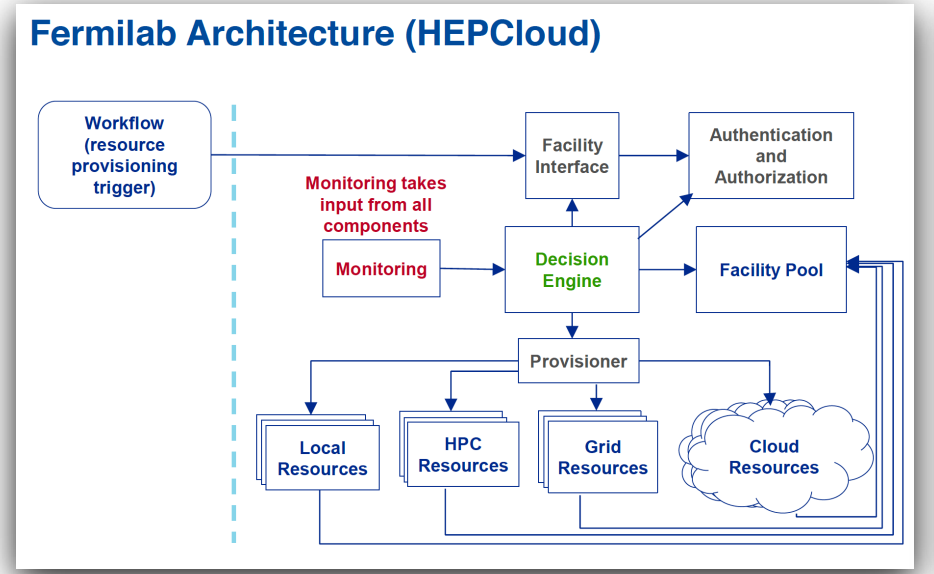
Provisioning HPC resources into CMS infrastructure

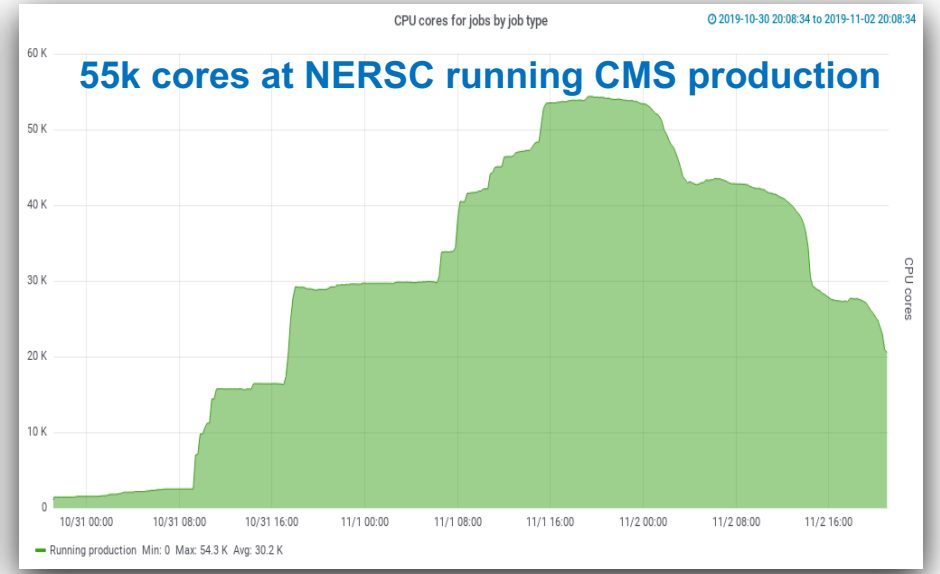


CMS at HPC via HEPCloud

Resource provisioning with HEPCloud: **Decision Engine** provisions resources based on queued job properties, state of the HEPCloud pool and resource manager set of policies

Fresh from the press!!





CMS jobs network needs

Pilot needs:

- CMS submit pilot jobs to sites. These interact with the CMS Global Pool to:
  - **Retrieve configuration and validation scripts** to create and run the HTCondor startd for CMS
  - **Late binding**: connect to the collector/negotiator to get payloads to execute

Payload job needs:

- **Software distribution**: CVMFS is needed for CMSSW versions and Singularity images
- **Conditions Database** access: Payloads get conditions from Squid Caches (port 80)
- **Data input and output:**
  - Data processing tasks: input data streamed to payload
  - Simulation tasks: still need to read multiple input data files, streamed via XRootD (to add pile-up)
  - Output insertion on CMS data catalogues

A Hard case: CMS at BSC

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

The **Barcelona Supercomputing Center** is the **main HPC center in Spain**

- Their biggest general-purpose cluster is called MareNostrum4 (**>150.000 processors**, peak power is **11.15 Petaflops),**
- Selected for deployment of a new **near-Exascale cluster, MN5 (200 Petaflops by 2021)**

BSC hard environment for CMS processing:

- Compute nodes do **not have open ports**
- The login machines **only allow ssh incoming connection**
- **No support for edge services** (Squids, CVMFS, …)
- **Disk is however accessible**: shared GPFS area mounted on all of the BSC **compute nodes**. Also on the **login machines** (UI) and externally by **sshfs**

PIC-HTCondor project to enable BSC network isolated nodes for CMS use

07 June 2019

EuroHPC selects Barcelona Supercomputing Center as entity to host one of the largest European supercomputers

PIC port d'informació científica

HTCondor High Throughput Computing

A Hard case: CMS at BSC

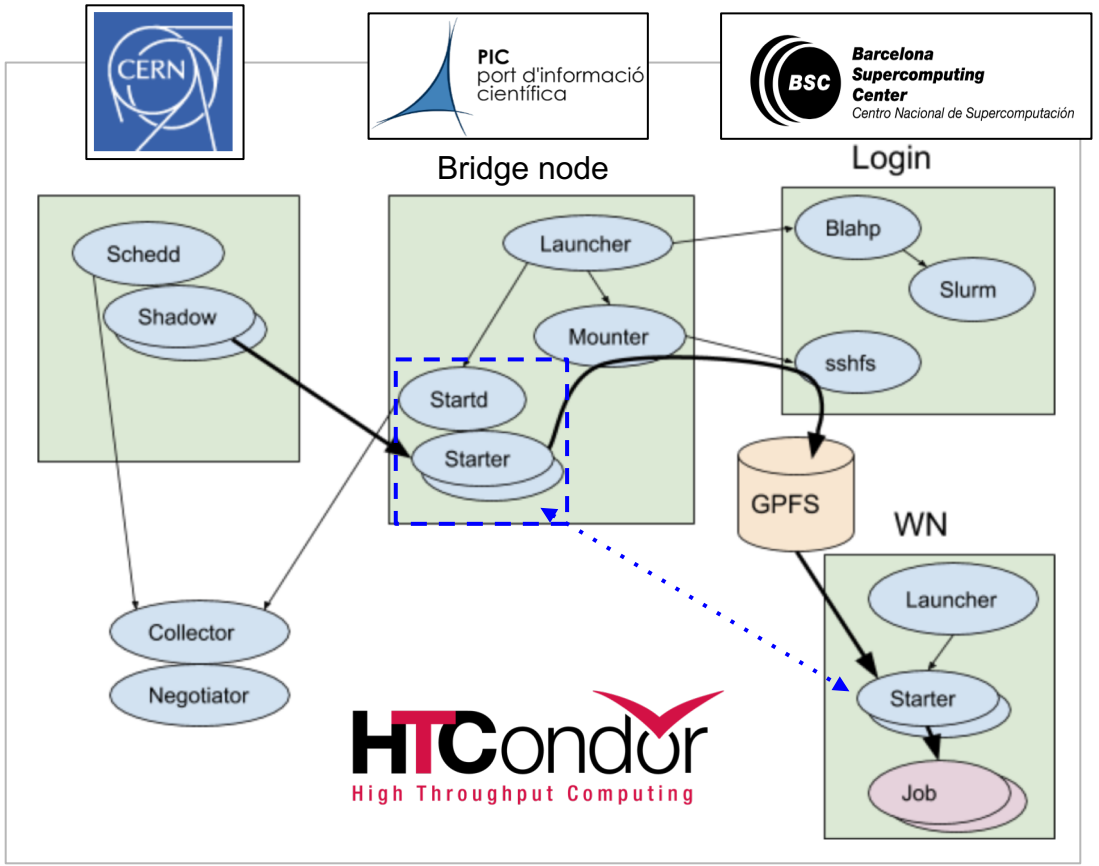
Use communication via FS to replace WAN

A **bridge node** installed at PIC submits **slurm jobs to BSC** that instantiate **HTCondor starters** on the BSC nodes, which connect back to **startd deamons** running at PIC

Job input sandbox, status, etc, **passed as .tar files via gpfs from bridge to WN**

The **startd** process at PIC opens to **matchmaking** by **CMS jiob schedds at CERN**

From a functional perspective, **the node at BSC has joined the HTCondor pool at PIC, federated to CMS Global Pool**



Conclusions

- CMS is dedicating **efforts to integration of HPC resources**, which will be needed by Run 3 and beyond
- Rely on **local** CMS teams for the **handshaking with each HPC center**
- Software development needed to **exploit GPUs**, as future HPCs going in that direction
- Ensure **operations to be transparent** with enhanced **description** of resources and workflows for proper automatic matchmaking
- Pre-place **input datasets**, or ensure WAN connectivity for streaming input data
- HPC resource allocation into CMS infrastructure can be achieved via a number of technical routes
  - **Some already working (NERSC)**
  - Others in testing/prototyping phase
- Even with **network restrictions, technical solutions can be found** (at additional costs)

BACKUP SLIDES

Abstract

High Energy Physics (HEP) experiments will enter a new era with the start of the HL-LHC program, where computing needs required will surpass by large factors the current capacities. Looking forward to this scenario, funding agencies from participating countries are encouraging the HEP collaborations to consider the rapidly developing High Performance Computing (HPC) international infrastructures as a mean to satisfy at least a fraction of the foreseen HEP processing demands. Moreover, considering that HEP needs have been usually covered by facilities cost-optimized rather than performance-optimized, employing HPC centers would also allow access to more advanced resources. HPC systems are highly non-standard facilities, custom-built for use cases largely different from CMS demands, namely the processing of real and simulated particle collisions which can be analyzed individually without any correlation. The utilization of these systems by HEP experiments would not trivial, as each HPC center is different, increasing the level of complexity from the CMS integration and operations perspectives. Additionally, while CMS data is residing on a distributed highly-interconnected storage infrastructure, HPC systems are in general not meant for accessing large data volumes residing outside the facility. Finally, the allocation policies to these resources is quite different from the current usage of pledged resources deployed at CMS supporting Grid sites. This contribution will report on the CMS strategy developed to make effective use of HPC resources, involving a closer collaboration between CMS and HPC centers in order to further understand and subsequently overcome the present obstacles. Progress in the necessary technical and operational adaptations being made in CMS computing will be described.

Related talks at CHEP2019:

- **Modeling of the CMS HL-LHC computing system**
- **Using OpenMP for HEP Framework Algorithm Scheduling**
- **Bringing heterogeneity to the CMS software framework**
- **Evolution of the CMS Global Submission Infrastructure for the HL-LHC Era**

Detailed job description for custom-built scheduling decisions

Jobs attributes to be used for resource allocation and workflow scheduling:

- **Site name**: used nowadays as a proxy for input data needs, location and availability, as this is known to CMS computing Operations
  - do we want to introduce additional _HPC (e.g. T1_ES_PIC_HPC) site names in order to assign jobs specifically for those resources? Prefer to hide resource complexity from Ops team
- **RequestCPUs, RequestMemory, Execution time:** clear requisites, but also:
  - **resizable or not**: would the job fit if resized into the non-standard slots?
- **Job type identification**: as HPC resources are not generic, need to tag jobs appropriately so that certain matchmaking policies can be implemented, e.g.:
  - Only GEN jobs to HPC (needs **changes in WMAgent code?**)
    - **CMS_JobType, WMAgent_RequestName, WMAgent_SubTaskName**
  - Exclude (or not) analysis (**AccountingGroup**)

Detailed job description for custom-built scheduling decisions

- **Input data:** does the job takes input data, which dataset, where it is located: (**DESIRED_CMSDataset, DESIRED_CMSPileups, DESIRED_CMSDataLocations**)
  - don't match jobs that require input data at all
  - limit based on some rate: IO needs in terms of bandwidth (**EstimatedInputRateKBs**)
  - also input/output volumes
  - exclude those not reading from the closer/host glide site (CINECA from CNAF, or BSC from PIC)
- **Priority of the request:**
  - backfill HPC with CMS low-prio jobs, while high-prio runs on WLCG?
  - Or rather consider both types of resources on equal terms
- **Special runtime environment:**
  - OS, singularity images, etc (**REQUIRED_OS, DESIRED_Archs)**
  - CMSSW when full cvmfs is not available (**CMSSW_Versions**)

Provisioning HPC resources for CMS

Mechanisms to join and access HPC resources: some are already being employed by CMS, other, more complex, are in prototype or test stages

- **Grid-like**, directly via GlideinWMS pilot submission to a CE, then **generic slots** join the CMS Global Pool: e.g. **CSCS HPC**
- **Grid-like**, but with **customized pilots**, include customized scheduling filters (e.g. input dataset)
- **Vacuum-like**: resources are acquired independently of glideinWMS, but would still join the CMS Global Pool. E.g. **DODAS** pointing to HPC instead of Cloud
- HPC slots might join a **remote HTCondor pool**,
  - with dedicated CMS WMAgents and schedds: **HEPCould**
  - rely on **flocking** from central CMS schedds: e.g. **BSC slots joining PIC pool, federated to CMS Global pool**
- Etc (condor annex?)

- Independently of the method, decision to **launch/request/instantiate resources** would happen in **reaction to demands** (workload pressure from suitable jobs)

=> Detect suitable jobs and act

Provisioning HPC resources for CMS

Each case presents **technical challenges** that need to be solved:

- Multi-node pilots
- Slot validation for resources not allocated via pilots (see backup for details)
- Remotely customized matchmaking rules
- Specialized singularity images
- Flocking to remote pools
- Access to isolated HPC nodes via FS
- etc

Slot validation

- This is covered by GlideinWMS natively in those resource allocation models where it is employed
- In other cases, CMS is preparing the use of an “emulated pilot” (*) mechanism to download CMS set of slot validation scripts in order to prepare it and ensure it can be used by CMS jobs
  - Basically, download validation scripts from the GlideinWMS factory
  - Mechanism ready, needs to be fully tested yet

- Customized validation: regular validation assumes things like access to singularity images via cvmfs. But what if:
  - No cvmfs available?
  - Need custom, not default singularity images?
  - Etc

(*) http://cms-htcondor-monitor.web.cern.ch/cms-htcondor-monitor/_book/emulated_pilot/

Scalability and efficiency of CMS Submission Infrastructure

The Submission Infrastructure picture is going to be evolving, by increasing in size and complexity

Must ensure that CMS can operate in the most complex scenario, in which resouyrces can be acquired in any and all of the mechanisms described:

- Support inflating the pool(s) to higher limits than today
- Being able to handle more schedds in parallel, maybe specialized for regions, to serve CMS's but also federated pools
- Having each of the schedds potentially talk to multiple (many more) negotiators than today
- Schedds with sufficient job submit/start/complete rates that can use pools that can grow with elasticity
- Ability to move workload between schedds for more flexible operations?
- Etc

Monitoring

- Regardless of the mechanism that defines how resources are exploited, CMS will need to have a global view of what's going on.
- We are monitoring the pools, but can't see slots not attached to our collector, or remote schedds that we can't query.
- So if CMS wants to access the global picture, we probably need to base it on job monitoring, rather than Sub. Inf. pool monitoring.
- Need to anticipate possible loose ends together with the monitoring team.

Accounting

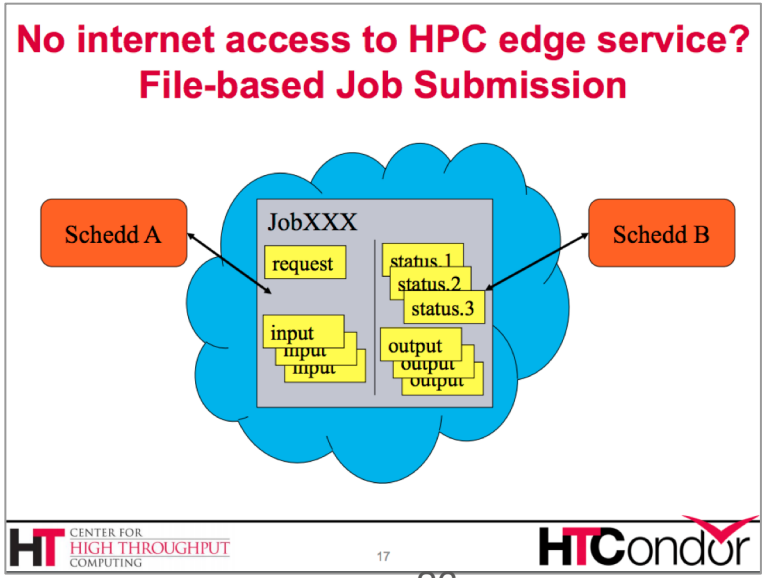
- In the case of grants for HPC usage, need to accurately measure resource utilization to demonstrate to the HPC management body that the allocation has been properly used
- Divergent criteria to the usual standards employed by CMS/HEP (e.g. accounting based on HS06*h)

Alternatives in case of limited connectivity

- Slot validation: “emulated pilot” mechanism to download CMS set of validation scripts in order to prepare the slot to be used by CMS jobs
  - Basically, download validation scripts from the glideinWMS factory
- Software: pre-placed container images locally available, including
  - Operating system plus dependencies
  - Pre-packaged, payload-dependent, CMS SW images
  - Install software in shared FS?
- Database: Extract conditions data and make it available locally as additional input files
- Input data: either
  - Limit the needs for input data by specifically choosing tasks that do not require it (pure simulation)
    - Limits flexibility and usability of these resources
  - Data pre-placement to local storage: dedicated agent to feed input files and get output files

Jumping the network barrier with HTCondor

- It's clear that most (all?) of the HPC-exploiting experiences in CMS have required **integration efforts** as well as tuning and some flexibility from HPC providers
- In order to succeed in the BSC very difficult scenario we can make use of a HTCondor developers idea: **use a shared FS as control path for HTCondor**

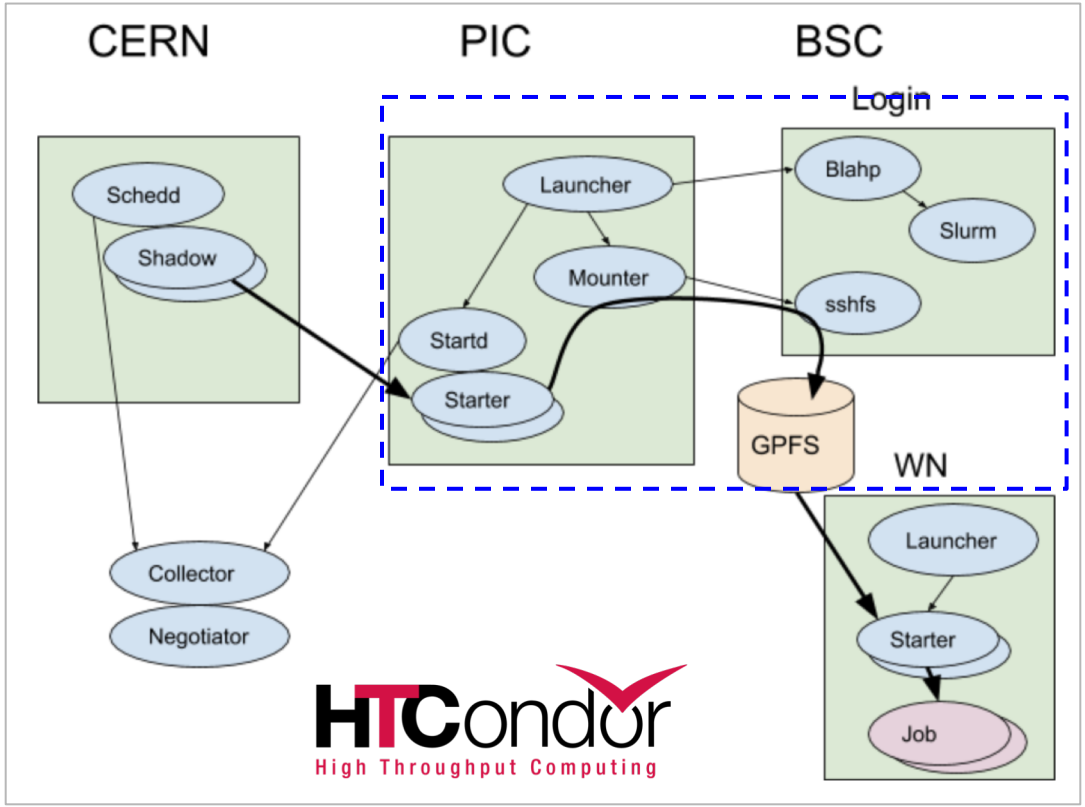


T. Tannenbaum
"What's new in HTCondor",
HTCondor Week 2018
https://agenda.hep.wisc.edu/event/1201/session/8/

Hard case in detail: CMS at BSC

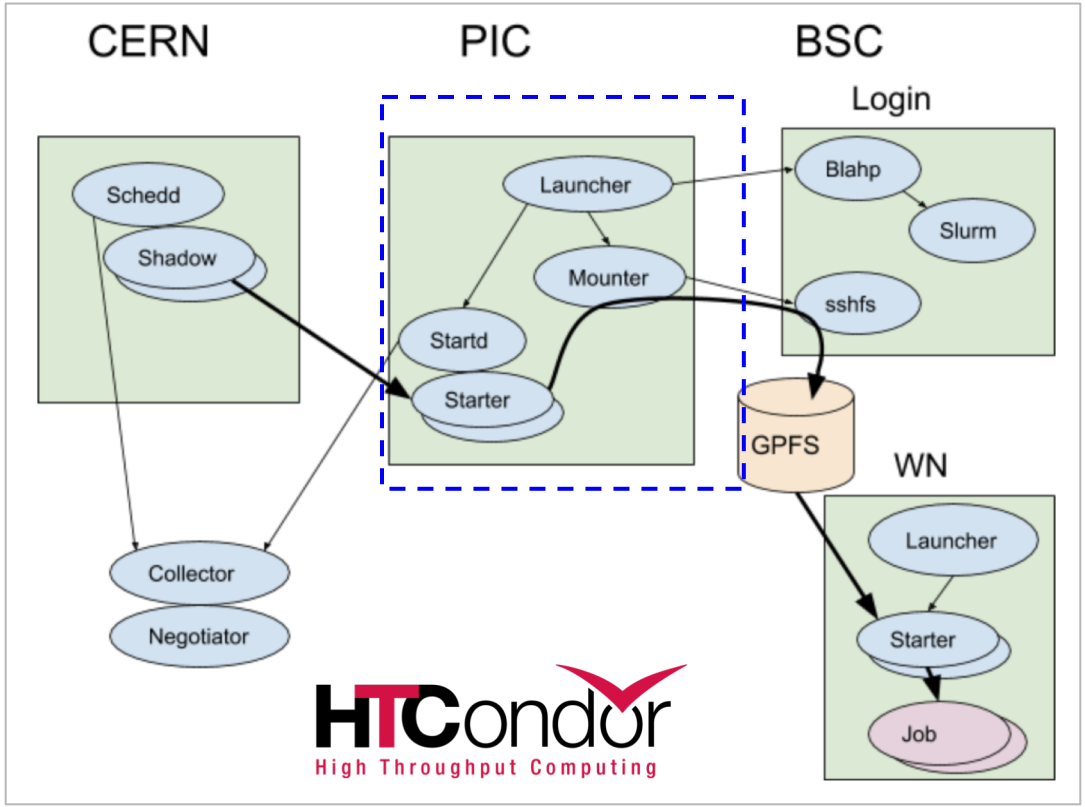
The bridge node interacts with BSC login node:

- submit jobs to BSC Slurm batch system using *blahp* over *ssh* (a setup known as Bosco)
- file transfers also over ssh



Hard case in detail: CMS at BSC

- For each node running the Slurm job, a **condor_startd** daemon is spawned on the PIC server
- Startds are configured to act like they have the hardware resources (cpus, memory, disk) of a single BSC execute node
- Each startd sets up an *sshfs* mount on the PIC server of BCS's GPFS file system (via BSC login node), creating a series of rendezvous directories in the FS, one per BSC node
- Startd's join the pool collector but advertise themselves as being unavailable for matchmaking
- Once the Slurm job starts running, *startds* re-advertise as ready to accept jobs



Hard case in detail: CMS at BSC

Once Slurm jobs start running, and startds are declared as available, the usual HTCondor machinery progresses:

- Matchmaking is done at the local negotiator
- CMS submit machines (schedds) connect to the startds via **flocking**
- **starters** are spawned at PIC and payload sandboxes are transferred from the schedd
- Then, instead of spawning the payload job directly, the starter copies the **job_ad** and **input sandbox** to the corresponding rendezvous directory of its startd in GPFS.
- PIC starters wait for the output sandbox and a job_ad summarizing the execution to appear in the rendezvous directory
- These files are copied out of FS and normal post-job-execution steps proceed (transfer the output sandbox to the submit machine, etc...)