HPC @ CMS

Daniele Spiga INFN

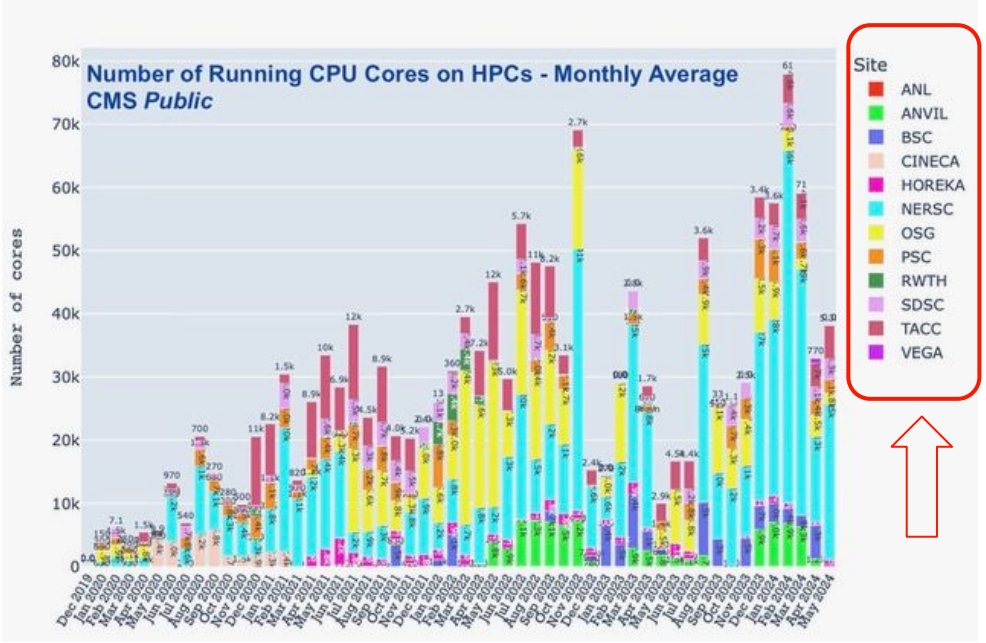
Credits for material and actual activities to several CMS and INFN colleagues

In a nutshell

After several investments by the collaboration over the past few years a number of HPC machines has been integrated and continuously been used in production mode throughout the year 2020.

CMS has developed and deployed distinct strategies

1. **Overlay batch model**
   a. Currently two incarnations (see later)
2. **Site Extension**
   a. Mainly relying on two pillars
      i. Storage on the “main site”
      ii. Site level defined matchmaking rules to allows to cherry picking only suitable workflow.
3. **HTCondor split starter mechanisms for filesystem based communication (BSC)**



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
- 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 docum

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]

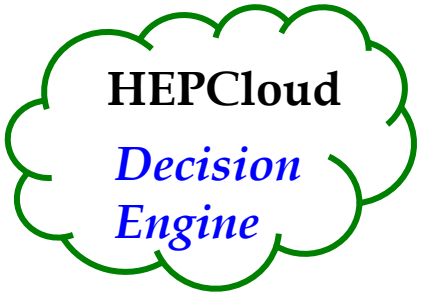
A synopsis for handshaking with sites

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		

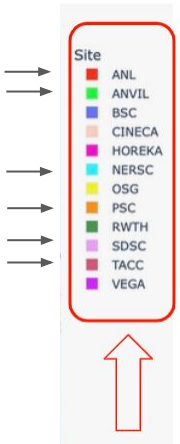
CMS Tech Document (co

	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

US HPCs via HEPCloud



- Portal to an ecosystem of diverse computing resources (academic/commercial)
- Used for integration of CMS computing infrastructure with US HPC resources
- Routes jobs to local or remote resources based on CMS workflow requirements, cost, and efficiency



- HPC runtime environment config through singularity or shifter container
- Access to CVMFS, and outbound network connectivity from worker nodes
- Remote read using AAA federation and remote stageout to FNAL storage

German HPC Resource Integration using COBaLD/TARDIS

Layers of Abstraction:

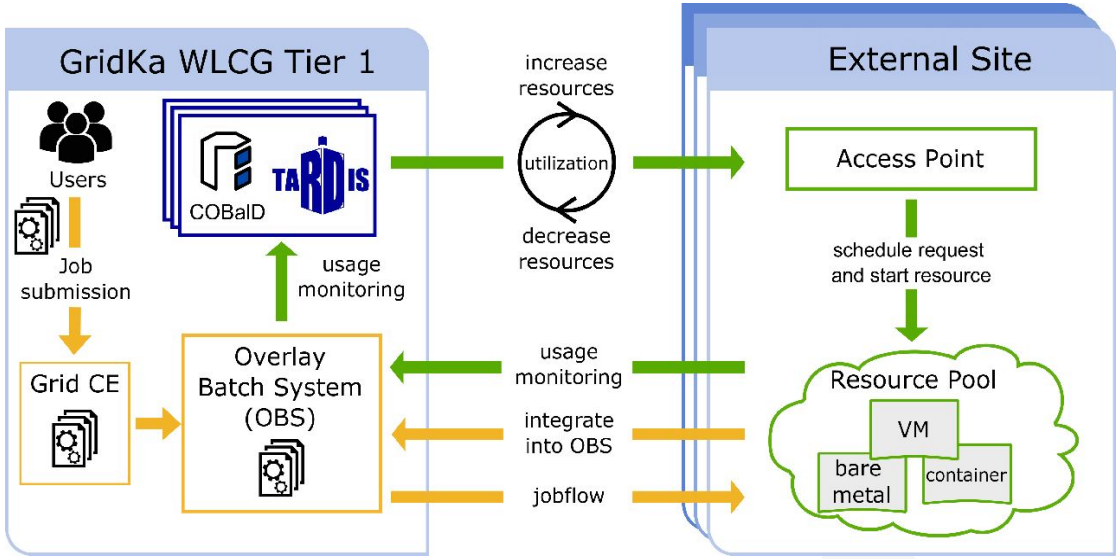
- Grid Compute Elements act as well established single point of entry
- Overlay Batch System provides a single pool of resources hiding complexity

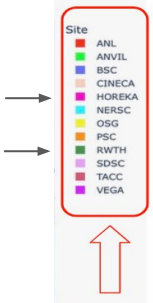
⇒ Transparent & hassle-free access to HPC resources

Resource Integration:

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

⇒ Efficient and dynamic access to HPC resources





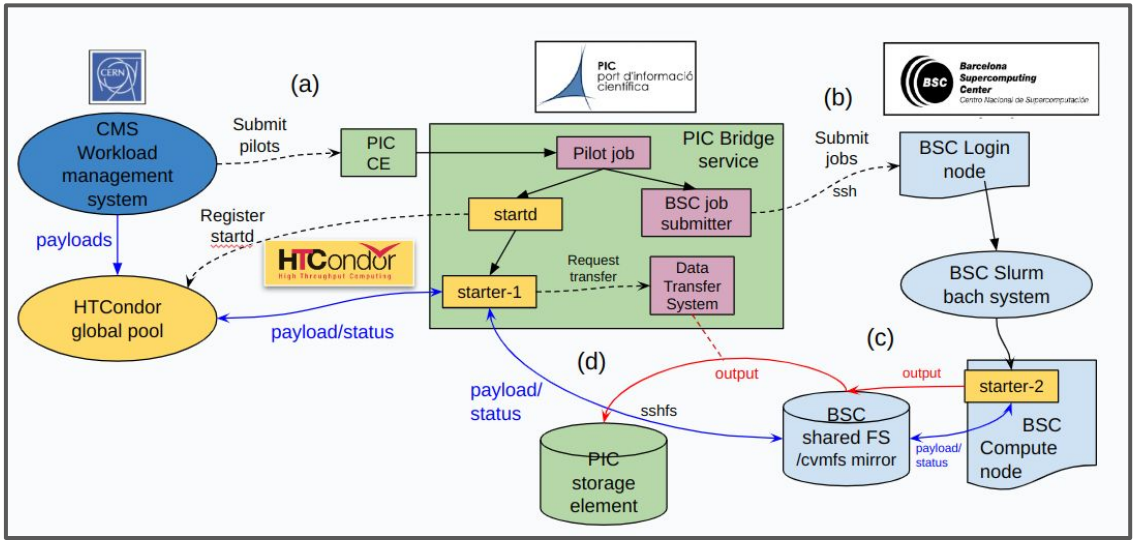
Overlay batch && site extension

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-place them at BSC
- Develop a service for output data transfer



Integration of CINECA into CMS Computing

Activity started under Prace grant back in to 2019

CMS and CINECA were able to agree 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;
- HTCondor-CE allowed as edge node.

Following the CNAF Tier-1 strategy the CINECA nodes were configured as **an elastic extension of CNAF Tier-1** receiving all the jobs targeted for the standard WLCG 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.

The First experience with CINECA: Marconi A2

- Production level activities demonstrated with the PRACE Grant #2018194658 (PI: T.Boccali) on the **Marconi A2** (KNL) and **Galileo** (Xeon) HPC systems
- Platforms are x86_64, so no real problem with sw, but a lot of work to overcome the mismatch between typical WLCG jobs and a typical HPC site

Marconi A2 Partition

- **3600 nodes with 1 Xeon Phi 2750 (KNL) at 1.4 GHz and 96 GB of RAM**
- **68 cores/node, 244800 cores**
- **Peak Performance: ~11 Pflop/s**

GALILEO

UserGuide

Model: IBM NeXtScale cluster
Architecture: Linux Infiniband cluster
Network: Intel OmniPath (100Gb/s) high-performance network

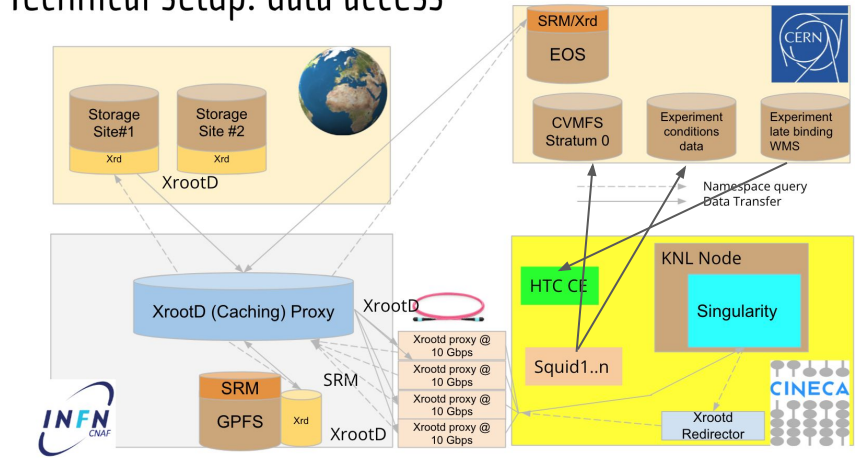
Nodes: 1022 (Intel Broadwell)
Processors: 2 x 18-cores Intel Xeon E5-2697 v4 at 2.30 GHz
Cores: 36 cores/node, 26.572 cores in total
RAM: 128 GB/node

Deploy edge services at the boundary CINECA/CNAF: HTCondorCEs for SLURM, SQUIDs for Database access, ...

A standard CINECA Marconi A2 is node configured with	A typical WLCG node has
An Intel(R) Xeon Phi(TM) CPU 7250 @ 1.40GHz: 68 or 272(HT4x) cores, x86_64, rated at ~¼ the HS06 of a typical Xeon per core	1-2 Xeon-level x86_64 CPUs: typically 32-128 cores, O(10 HS06/thread) with HT on
96 GB RAM, with ~10 to be reserved for the OS: 1.3-0.3 GB/thread	2GB/thread, even if setups with 3 or 4 are more and more typical (so a total 64-256 GB)
No outgoing connectivity from the node	Full outgoing external connectivity, with sw accessed via CVMFS mounts; additional experiment specific access needed (condition DBs, input files via remote Xrootd, ...)
No local disk (large scratch areas via GPFS/Omnipath)	O(20 GB/thread) local scratch space
Access to batch nodes via SLURM; Only Whole Nodes can be provisioned, with 24 h lease time	Access via a CE. Single thread and 8 thread slots are the most typical; 48+ hours lease time
Access granted to individuals (via passport / fiscal code identification)	Access via pilots and late binding; VOMS AAI for end-user access

Enable NATted connectivity to just CNAF and CERN, use proxy-caches to fan-out

Technical setup: data access



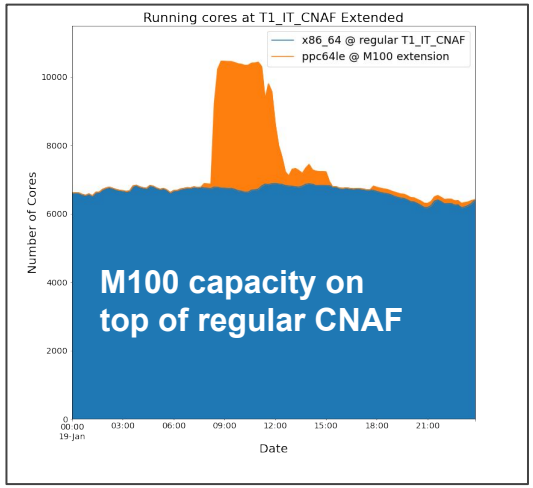
M100 integration

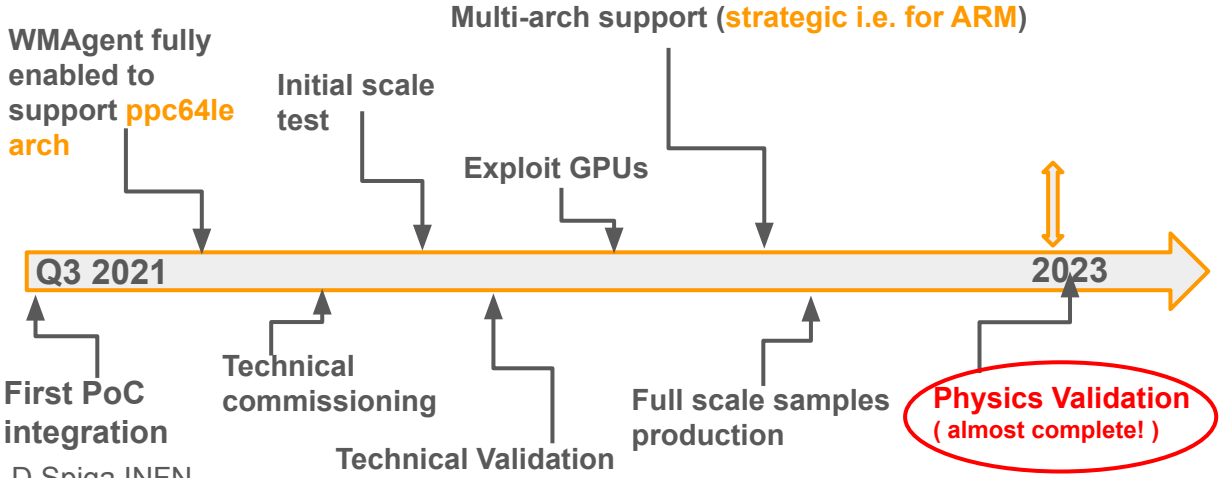
MARCONI - 100

Nodes: 980
Processors: 2x16 cores IBM POWER9 AC922 at 3.1 GHz
Accelerators: 4 x NVIDIA Volta V100 GPUs, Nvlink 2.0, 16GB
Cores: 32 cores/node
RAM: 256 GB/node
Peak Performance: ~32 PFlop/s

Site extension approach. M100 is a sub-site of Tier1 CMS computing operations deals only with CNAF

- CVMFS and squids provided by CINECA
- Outbound connectivity to CERN and CNAF (access storage @ CNAF)
- Access to the AAA Data Federation via xrootd proxy (@CNAF)
- Worker Nodes provisioning: via site launched or through regular CEs@CNAF. Relying also on custom matching rules (defined at site)





Then the logical extension to VEGA EuroHPC and now Toward Leonardo

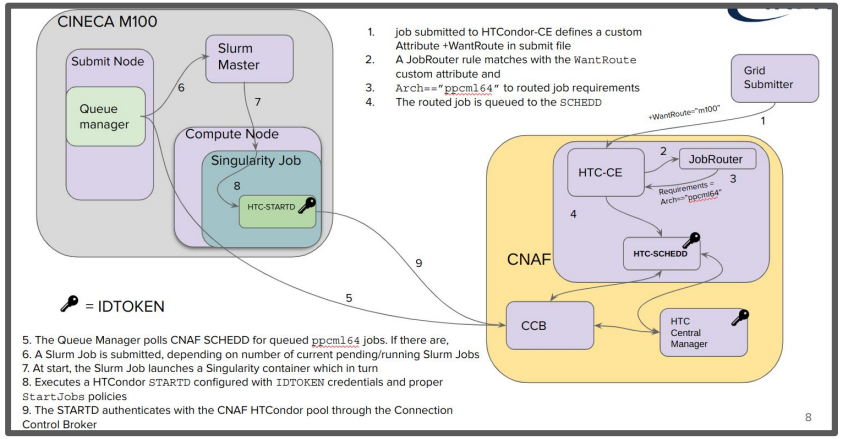
M100: From Slurm CN to CNAF WN

Overlay batch && site extension

- Two simple ideas:
  - have a HTCondor STARTD running as Slurm Job
  - Detect pending jobs for ppcml64 @CNAF and trigger resource creation

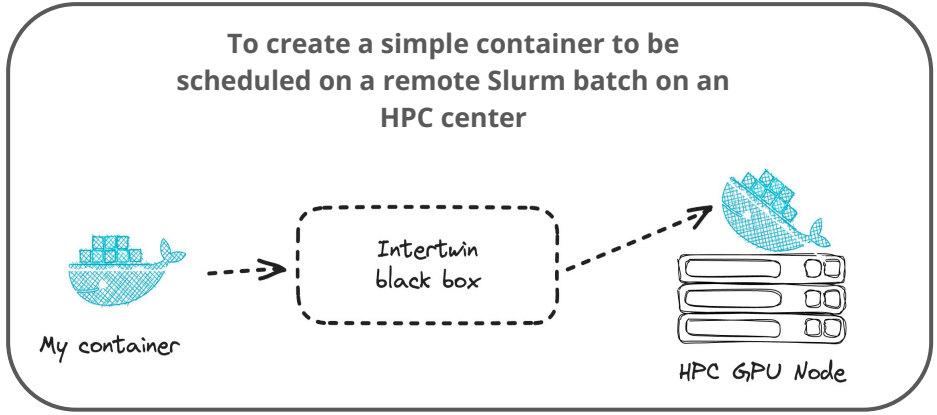
Implementation

1. From a M100 Login Node submit a Slurm job
2. At start, it launches a Singularity container which
   - activates an HTCondor STARTD, which
   - authenticates to the Central Manager and join the CNAF pool
     - CCB (Connection Control Broker 9618 port)
     - IDTOKENS for authentication
   - Becomes available to execute jobs submitted to HTCondor-CE at CNAF.
     - StartJobs expression to only accept proper jobs

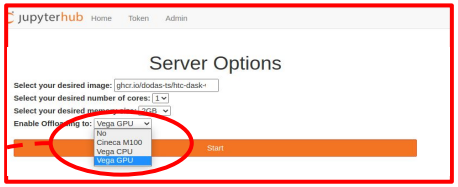


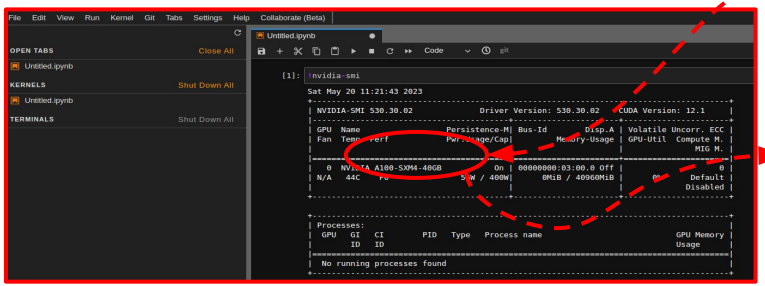
Pushing Forward

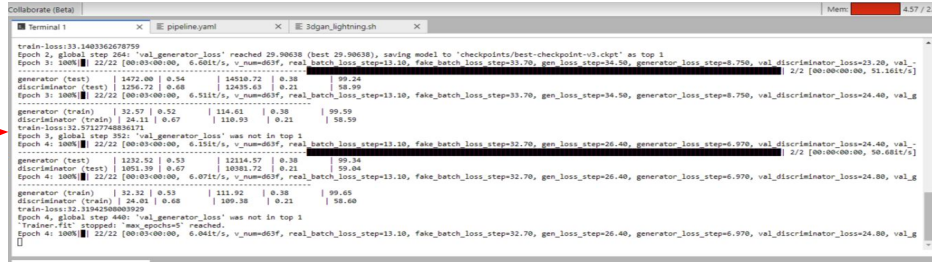
- Extend the container orchestration de-facto standard (K8s) to support offloading under the hood
  - With little to no knowledge required from the end user perspective



Spawning on-demand jupyterLab instance on an HPC along with other more “cloud-ish” instance on K8s.



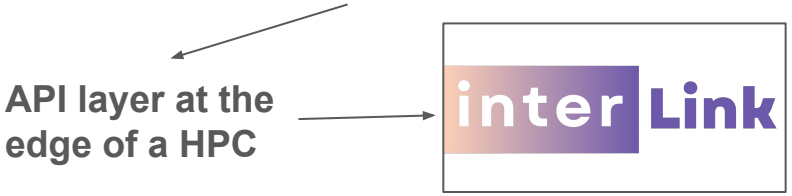


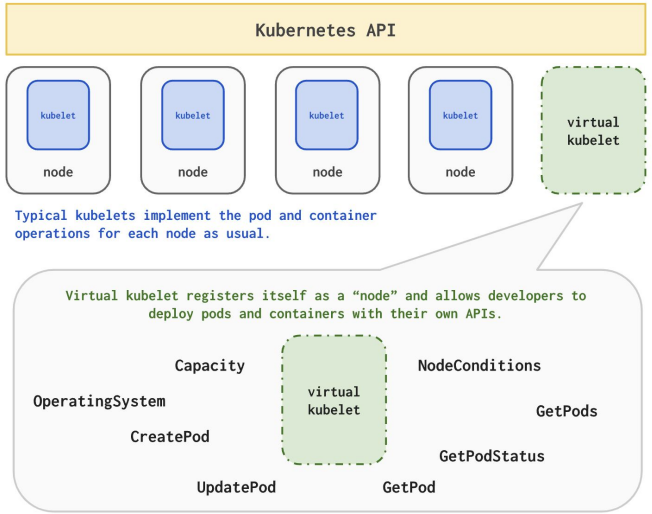


How to achieve it

Site (HPC) wise we've similar set of requirements as before: Edge node, CVFMS, Apptainer a bit of outbound… a plus: a "federable" storage (even ephemeral endpoint)

- Federated Identity? It would be a plus (a dream)





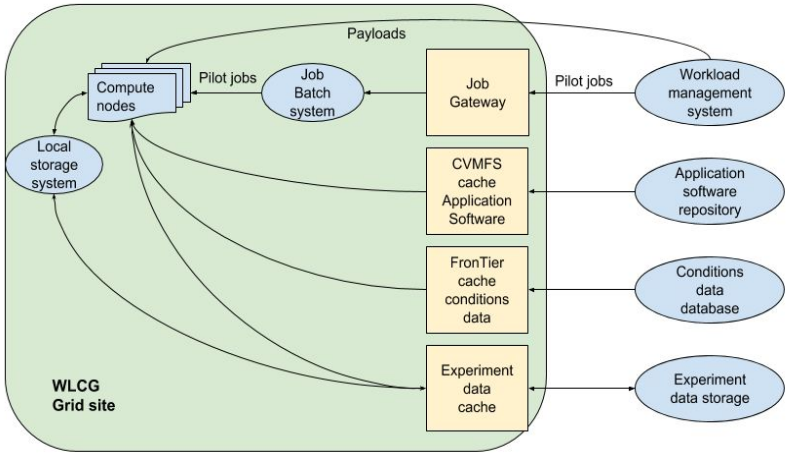
Making a Kubernetes cluster send your containers/pod to a "virtual" node that seamlessly takes care of your application lifecycle on a remote server/hpc batch queue?

- **While exposing the very SAME experience of running a pod on the cloud resources**
  - Meaning that the API layer exposed is the "normal" set of K8s apis

Backup

CMS Activities on Network Segregated centers

- **tsocks** is a networking tool to trap system calls and allow re-routing TCP+UDP connections. CMS developed a solution using tsocks in combination with a SOCKS5 tunnel to overcome networking restrictions at HPC sites.
- **Dante proxy** is an other powerful and flexible proxy solution allowing various types of tunnels. Network masks and traffic limitations are handy features.
- **proxychain-ng** allows the configuration of complex routes including several proxies. It can also be used a pre-load library to trap the network calls.



Typical external network connections by a CMS payload