Heterogeneous resources integration to CMS WM and SI

HSF WLCG Virtual Workshop on New Architectures, Portability, and Sustainability

A. Pérez-Calero Yzquierdo,
May 12th 2020

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

Motivation (I)

CMS needs to be **ready to utilize heterogeneous resources: an increasingly larger and more relevant part of the available computing power to LHC VOs**

- **Grid**: WLCG sites supporting CMS already providing some GPUs
- **HPCs accessible to LHC VOs**: built on heterogeneous architectures CPU+GPU (e.g. BSC's MN5)

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

The European Union would contribute around 100 million euros to MareNostrum 5, the highest EU investment in a research infrastructure in Spain.

An heterogeneous supercomputer to meet today's needs

Another of the remarkable aspects of the future MareNostrum 5, which will have a peak performance of 200 Petaflops (200×10^{15} of operations per second), is that it will be an heterogenous supercomputer.

Taking advantage of non-CPU computing power requires evolution of the CMSSW code but also **WM tools:**
=> this talk aims to be **a basis for the discussion in the CMS context**

Motivation (II)

Use of GPUs is important for CMS not only for **offline** but also **online processing**:

- **CMS is investigating the opportunity to test a heterogeneous CPUs + GPUs solution at the HLT farm for Run3**

Code and framework being re-written to take advantage of accelerators (NVidia GPUs now, more likely to come in the future):

- enabling **support for heterogeneity** in the CMSSW framework
- revision of the reconstruction algorithms and data structures, use of **performance portability** libraries
- need to **validate** the heterogeneous code for HLT

Regarding CMS offline processing, no impact yet from the limited number of available GPUs

- Re-written code is still not used, or takes very little time wrt whole task execution
- This might change soon!

GPU Model for CMS Offline production

A model for Workload Management that includes GPUs (and other non-standard resources), first needs to **consider how these resources will be allocated**:

- Employ **dedicated GPU** resources (like a PS4 farm)?
- Or GPUs will always be alocated **associated** to >=1 CPUs?

An how we want to schedule work on such resources:

- Will workflows be **specifically targeting** resources that **must also provide GPUs**?
- Or will jobs match to **whatever resource** slots and, if available, **use GPUs in a transparent way**?

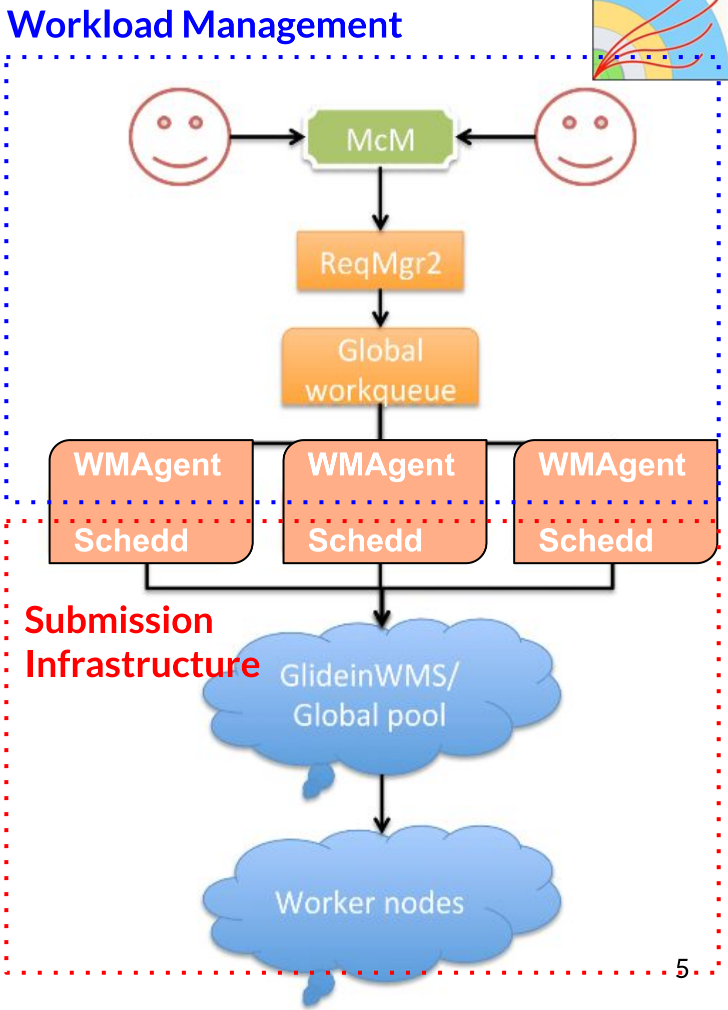
A concern for the second option from **Operations** pov: **job splitting when running on heterogeneous resources** resulting in **execution times** distributions spanning **an order of magnitude**.

- Even **current levels of CPU performance spread** (along with inaccurate measurements of execution time at the workflow validation step), **already producing issues today**.

=> Depending on the answers, there might be **development needed on the CMS Workload Management and Submission Infrastructure systems (next slide)**

Workflow to job creation to execution in CMS

1. Physics groups **request workflows** providing job config. (dataset, CMSSW version, conditions, etc)
2. McM uses that information to **test and create** a workflow in ReqMgr2
3. Global WorkQueue uses the **workflow specs** to create **chunks** of work (**WQE**)
4. The WMAgents **pull down** these chunk(s) of work, then **materialize them into jobs,** populating **condor submit nodes** (**schedd**) queues, including job requirements such as Site, Mem, Cpu, Arch, Storage, etc.
5. Those jobs generate pressure on the CMS GlideinWMS FE and pilot factories, **requesting and submitting suitable pilots to resources**
6. Once a running pilot appears that can fulfill a job requirements, the **job gets matched from the schedd to the available slot (a condor startd in the CMS Global HTCondor Pool)**



Overall WM Developments Needed

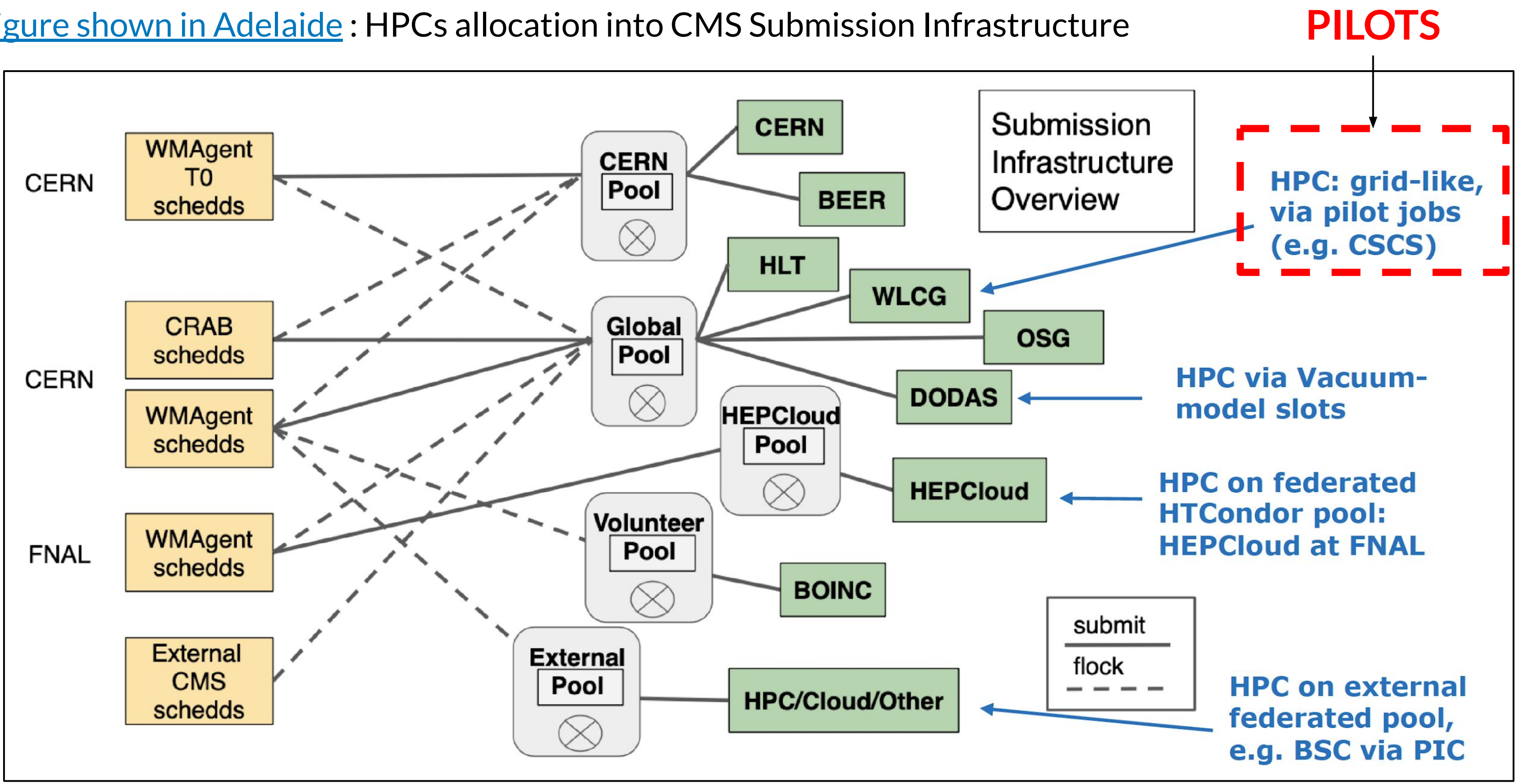
The following will evolve according to the GPU usage requirements and strategy, so take it with a grain of salt:

- CMS MC management team needs to **flag which requests would take advantage of GPU resources**
  - Is the use of GPU **optional** or **required** for this request?
  - Would it be decided at **runtime** or **predefined**?
  - If predefined, Physics users to flag and **validate** each request as GPU-like or not
  - do we need to **define a number/range/type of GPUs** to be used **per execution step**?
    - CMS using now a single Step Chain for Gen-Sim-Digi-Reco

- WMCore/ReqMgr needs to **support new request parameters** (use of GPU, type, etc)
- WMCore/WMAgent needs to **propagate such parameter** all the way down to the job classad

Allocating GPU resources with GlideinWMS (I)

Figure shown in Adelaide : HPCs allocation into CMS Submission Infrastructure



Allocating GPU resources with GlideinWMS (II)

- Each pilot should include an **additional GLIDEIN_Resource_Slots,** aka the GPU slot, along with the **CPU partitionable slot**
  - In principle, we could make use of (auto) **condor_gpu_discovery** tool to fill GPU-related slot properties...

- **GPUs auto-discovery is probably a bad idea** when running on the Grid and other **shared resources**!
  - Autodiscovery strategies might be being implemented independently at the **SW**, payload job (**WM**) and pilot (**SI**) layers!
    - redundant/conflicting information?
  - Also, auto-discovery might be tried by tasks from **multiple VOs** running on shared nodes
    - Wild competition for the same resource, leading to suboptimal utilization
  - Instead, **negotiate GPU requests with the LRMS, just like CPUs, memory, etc**
    - E.g. HTCondor CE & batch system to redirect request to suitable node

- Once allocated, need to properly **tag GPU resources**:
  - **TotalGPUs**, a partitionable quantity, just like CPUs
  - Extra tags such as **CUDACapability**, CUDARuntimeVersion, CUDAGlobalMemoryMb, etc

Allocating GPU resources with GlideinWMS (III)

Payload GPU jobs are executed in **singularity containers** (general for all CMS payloads)

Software libraries: need to be made **available** and correctly located in the slot **runtime environment**.

- Access to **specialized software** (e.g. singularity containers for Tensorflow jobs) as on standard resources via **CVMFS**

- **OSG-supported software libraries distributed via CVMFS:**
  - ready-to-use singularity images for TensorFlow workflows for both CPU and GPU jobs
  - Containers based on Ubuntu
  - OSG maintains the versions of the CUDA Deep Neural Network library (cuDNN), TensorFlow, Keras, etc.

- **Additionally**, launch singularity containers with `--nv` **option** for **host-system libraries** to be mapped into the container
  - singularity itself takes care of a lot of the CUDA/OpenCL library integration/compatibility between the host and the container.

Matchmaking GPU-suitable jobs in HTCondor (I)

- All jobs that use **GPUs** must **request** this resource in their submit file
  - for example: `request_gpus = 1`
  - along with the usual requests for CPUs, memory and disk

- In cases where code that requires a specific version of **CUDA**, a certain type of GPU, or has some other special requirements, jobs can explicitly request it via their **jdl**, making use of the corresponding **slot attributes**.
  - For example, if a job needs a certain version of the CUDA libraries or a certain class of GPU, jdl should include:

```
requirements = (CUDARuntimeVersion >= 4.0) && (CUDACapability > 4.0)
```

- Jobs then need to specify also a **singularity container** suitable to create the appropriate environment
  - `+SingularityImage = "/cvmfs/singularity.opensciencegrid.org/opensciencegrid/tensorflow-gpu:latest"`
  - **Q: how to manage images?** should we use a generic standard, but increasingly fatter image, or rather maintain a well-defined collection of specialized images?

Matchmaking GPU-suitable jobs in HTCondor (II)

While no **`SLOT_WEIGHT`** associated to GPUs is defined, users are going to basically get the GPUs "for free" (=no impact on their quota), just “billed” on the basis of their requested CPUs. This is the situation today in CMS small GPU pool

However, once the number of **users and resources grow**, the consumption of the resources must be managed:

- Define proper **`SLOT_WEIGHT`** expression for HTCondor to calculate resource usage metric
  - Q: different weights for diverse types of GPU resources?
  - Also GPU+CPU combinations? GPU memory?

- Properly managing **priorities, shares and quotas** between groups and users probably best implemented by means of **dedicated HTCondor negotiator**
  - Allows to define **accounting groups** which can be assigned diverse GPU pool shares,
  - Manage matchmaking of GPU resources and requests, to apply policies such as fair-share on GPUs
- **Q: GPU accounting:** do we have any properly weighted “GPU-hours”?

Currently available GPUs in CMS

CMS already has a number of GWMS pilot factory entries for **Global Pool pilots that include GPUs**.

- These are **available at several US sites** (Nebraska, ND, Vanderbilt, Syracuse and UCSD).
- In the GWMS FrontEnd match rules: such pilots are **not requested for regular CPU-only jobs**

CMS Connect: a lightweight WM service providing an entry point for analysis jobs into the Global Pool, independent of WMAgents and CRAB

- **Already supporting user jobs requesting GPUs**
  - As described before, additional WM developments are needed for WMAgents (production) and CRAB (user analysis).

Interactive use of remote GPU resources:

- GPU resources can be in principle accessed **interactively** via **`condor_ssh_to_job`**
- Compatibility issue on **condor_ssh_to_job** and **singularity** blocking this feature up until recently:
  - Fixed by HTCondor 8.8.8 (8.9.7) and GWMS pilots with native Singularity support
  - However, conflict of environments between **pilot-launched GPU startd running inside the singularity container** and basic **environment derived from startd** when doing condor_ssh_to_job
    - Still a bit of tweaking needed in the pilot configuration

Multi-node slots at HPCs with GlideinWMS

- No real need for CMS workflows, of course.
    - Usual level of parallelism in CMS multithreaded tasks is of 8-16 CPU cores
    - Already using whole-node pilots with several such jobs in parallel

- However, **if required**, due to HPC scheduling policies for example, multi-node GlideinWMS pilots are available as an option.
    - Just used to launch **N single-node startd**, one per HPC node, which in practice will operate independently

Conclusions

Challenges in CMS WM+SI in order to access and use GPUs:

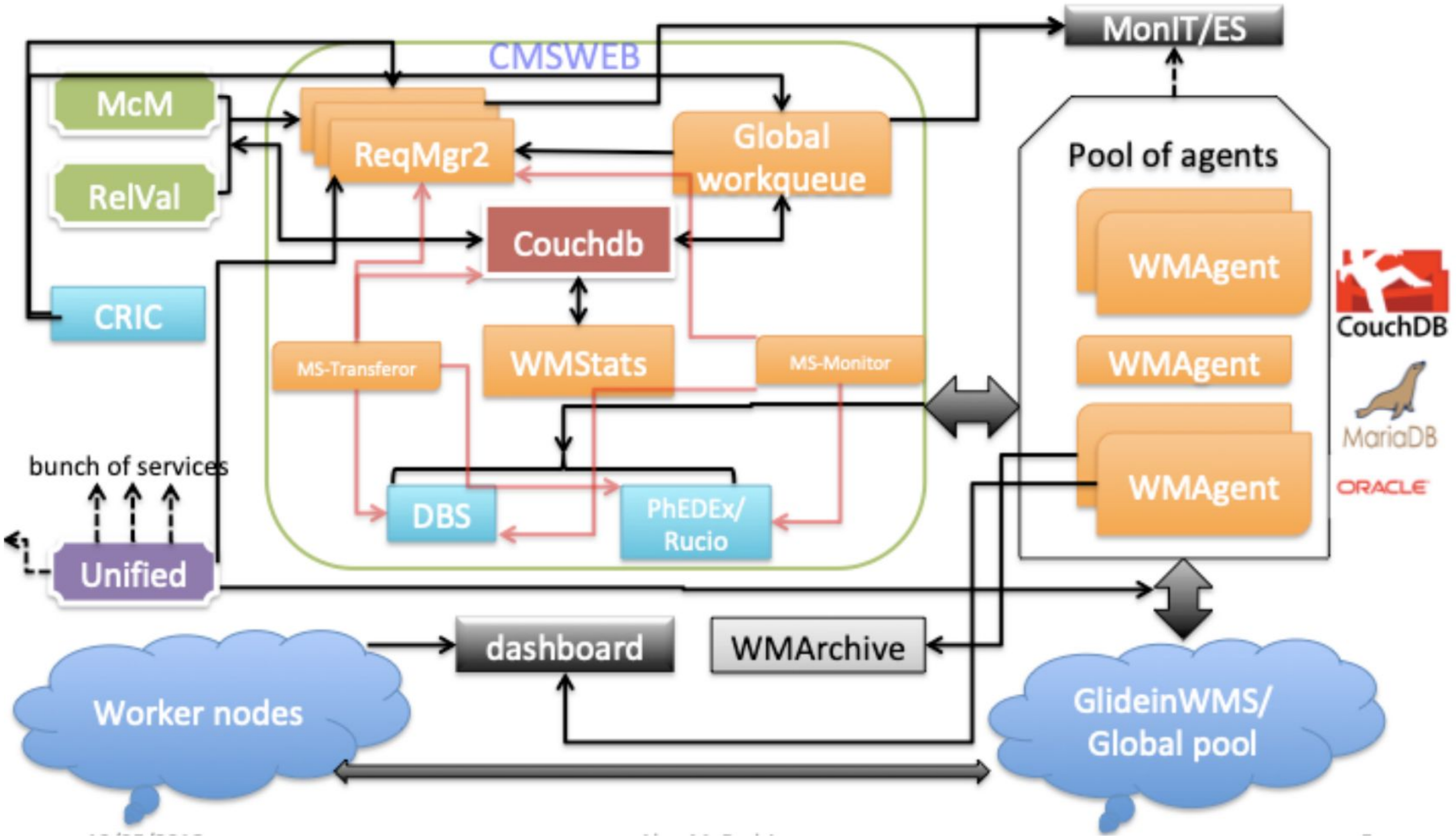
- CMS model of use for GPUs still under discussion, implications on the required evolution of CMS WM
  - Use exclusive or hybrid nodes, submit specialized or generic workflows
  - Implications for Computing Operations already recognized(e.g. job splitting)
- Specifically for WM, need to properly **tag** and **validate** requests that **can/must use GPUs**
  - Propagate requirement attributes to the job classad
- GlideinWMS already **supports the creation of pilots for GPU-like slots** and the use of **singularity images with specific software libraries**
  - Avoid GPU auto-discovery in all layers! Negotiate with LRMS
- **HTCondor supports matchmaking of jobs to GPUs**, with additional request tags;
  - Key point is to have workflows and resources properly **tagged**
  - Accounting, shares & quotas being worked on

CMS current capabilities and accomplishments:

- A small number of GPU resources already available in the Global Pool
- A submission tool for GPU user analysis jobs already in place (**CMS Connect**)

BACKUP SLIDES

CMS WM: All the Details



Some GPU-related references

- SI doc about GPUs: https://docs.google.com/document/d/1Ocf5yoxvb5gUgcFdCV7TBeP-t1QKHTHUYmKpwwaX1PQ/edit#heading=h.vxmd7td3nljv
- https://indico.cern.ch/event/739897/contributions/3559134/attachments/1922034/3179876/20191008-preGDB-bmk-AV-gpus-v001.pdf
- https://indico.cern.ch/event/739897/contributions/3559135/attachments/1922040/3179886/GPU_benchmarking_with_Patatrack.pdf
- https://indico.cern.ch/event/739897/contributions/3581585/attachments/1922378/3180454/20191008_Felice_Benchmark_739897.pdf
- https://docs.google.com/document/d/1Ocf5yoxvb5gUgcFdCV7TBeP-t1QKHTHUYmKpwwaX1PQ/edit#heading=h.vxmd7td3nljv