

# Brokering to heterogeneous resources in the CMS Computing Grid

A. Pérez-Calero Yzquierdo on behalf of the CMS Collaboration

## WLCG Workshop - CHEP 2023, May 7th 2023



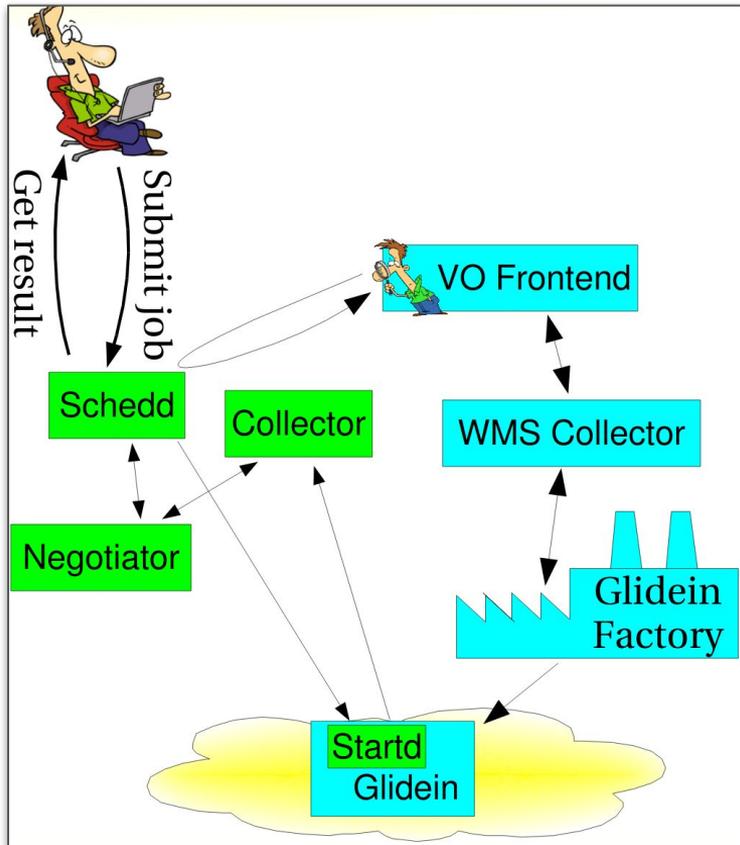
# Outline of the talk

- Intro to using heterogeneous resources
- GPUs integration in CMS Global Pool
  - GPUs in FE and pilot factories, first matchmaking
  - GPUs in the Global Pool, second matchmaking
  - Using GPUs in WM and CRAB
- GPUs already available to CMS
- Non-x86 CPU architectures
- Conclusion and next steps

- **The availability of compute power in non CPU, non x86 processor types is abundant and increasing**
  - For example, looking at the examples from the top500.org list, it's clear that:
    - A significant fraction of the processing power in HPCs is provided by accelerators.
    - Many top positions include non-x86 CPU architectures (e.g. IBM Power systems:Summit, Sierra, Marconi-100)
  - Not only at HPC facilities, but also among traditional WLCG computing sites as well
- **HEP experiments must be prepared to run substantial part of their processing tasks at HPCs**
- CMS internal study (**ECoM2x, 2020**) of the computing model and resource needs looking into the LHC Run 3, but specially at the HL-LHC phase, includes in its recommendations:
  - *“CMS should continue to **aggressively expand the resources** accessible to it for production processing at facilities beyond those dedicated to the LHC”*
  - *“**CMS should strive towards using HPC resources effectively**”*
  - *“Use of on board **accelerators** as much as possible during reconstruction”*
  - *“Work towards enabling CMS software on CPUs, GPUs, FPGAs and TPUs as primary targets for its software stack”*
  - *“**Support non x86\_64 CPU architectures**”*
- The CMS Phase-2 Computing model update document (**CMS-NOTE-2022-008**) also advises to **abandon the assumption of uniformity of our compute resources, evolving our WM and SI systems to embrace heterogeneity**

In relation to the use of GPUs in CMS offline computing, we need to consider that all GPUs available and in use by CMS today are still opportunistic, not pledged:

- **Pledged CPUs: agreed upon standard job execution slot**
  - 8-core slots, 48h of runtime, minimum of 2 GB/core, etc
- **No equivalent “standard job slot” for GPUs.** This affects:
  - How to approach sites on what GPU resources CMS can use: how do we even know there are GPUs available for CMS? What CEs, queues and other parameters to use?
  - What is the correct GPU type for CMS tasks?
  - How to configure the execution of tasks on these resources: no predefined rules on e.g. slot size, max execution time, memory, etc
- **Deeply affects how we use the resources:** no generic slot, requires a more careful and detailed job and slot description in order to select the correct slot and avoid wastage



**Two matchmaking stages** in Submission Infrastructure:  
resource allocation (GlideinWMS) and job to slot  
matchmaking (HTCondor)

## 1. GlideinWMS and pilot jobs

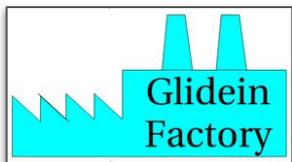
- Submit resource requests on sites CEs in order to join them into the Global HTCondor Pool

## 2. HTCondor matchmaking

- Slot joins the Global Pool, then resources are negotiated and assigned to payload jobs

**Heterogeneous resources just follow the same general logic**

- **GPU resource description for first matchmaking in pilot factory entries**
  - **Limited information about the available GPUs:**
    - **CEs that allow access to GPUs, queue names, etc**
    - Slot attributes **agreed upon** with the site admins, (e.g. GPU slot lifetime 24h? 48h?)
    - **Statically configured, with most GPU specs only available at pilot runtime**
  - Typically, GPU slots are configured along with standard pilot features (8 CPU cores, 2 GB/core RAM)

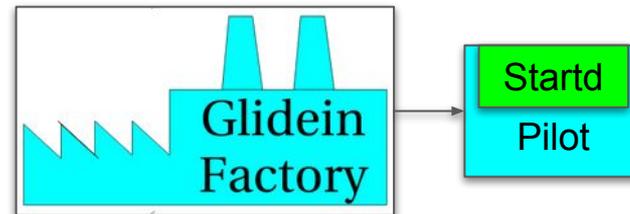


```

<entry name=" CMSHTPC_T2_US_Wisconsin_cmsgrid01_gpu" auth_method="grid_proxy"
gatekeeper="cmsgrid01.hep.wisc.edu cmsgrid01.hep.wisc.edu:9619" gridtype="condor" ...>
  <config>
    (...)
    <submit_attrs>
      <submit_attr name=" +maxMemory" value="20000"/>
      <submit_attr name=" +xcount" value="8"/>
      <submit_attr name=" Request_GPUs" value="1"/>
    </submit_attrs>
  </submit>
</config>
<attrs>
  <attr name="GLIDEIN_CMSSite" ... type="string" value="T2_US_Wisconsin"/>
  <attr name="GLIDEIN_CPUS" ... type="string" value="8"/>
  <attr name="GLIDEIN_MaxMemMBs" ... type="int" value="20240"/>
  <attr name="GLIDEIN_Max_Walltime" ... type="int" value="216000"/>
  <attr name="GLIDEIN_Resource_Slots" ... type="string" value="GPUs,1,type=main"/>
  (...)
</attrs>
</entry>

```

- **First matchmaking:**
  - Only jobs with **RequiresGPU** will trigger GPU pilot job submission
  - **Slot description very limited** (“slot has some kind of GPU”), runtime parameters are not available at this stage yet
  - But of course other parameters, such as site whitelist, still apply here
  - Upon successful match, pilots are submitted to all corresponding CEs
  
- **Being opportunistic**, we abide to site preferences concerning how to use their GPUs, **reserving the slot for GPU-like workloads:**
  - No CPU jobs will start for the first 30 mins
  - Then either open for all jobs, or return the slot back to site (site preference)
  - Once GPU is in use, try to saturate the remainder CPU part
  
- Given **late-binding**, to minimize waste, **avoid pilots landing on a GPU node once the GPU workload is done**
  - Remove stale pilots still queued at sites CEs
    - Common practice for CPU pilots, but even more critical for GPUs



- Once a pilot starts execution on remote resources, **GPU properties** are updated to the slot classad:
  - **condor\_gpu\_discovery** retrieves most of the relevant matchmaking attributes
  - **CMS\_CUDA\_SUPPORTED\_RUNTIMES** from CMS script in cvmfs

## GPU slot attributes

```
CPUs = 8
TotalSlotMemory = 20000
GPUs = 2
CUDACapability = 8.0
CUDAClockMhz = 1410.0
CUDAComputeUnits = 108
CUDACoresPerCU = 64
CUDADeviceName = "NVIDIA A100-PCIE-40GB"
CUDADriverVersion = 11.3
CUDAECCEEnabled = true
CUDAGlobalMemoryMB = 40536
CUDAMaxSupportedVersion = 11030
CMS_CUDA_SUPPORTED_RUNTIMES = 10.1,10.2,11.0,11.1,...
CMS_NVIDIA_DRIVER_VERSION = 515.48.07
```

# Second matchmaking in HTCondor

- For each properly configured GPU slot that joins the Global Pool, the HTCondor *negotiator* will compare **job requirements** with machine attributes (**second matchmaking**)
  - Any dynamically retrieved GPU property can now be used to filter suitable slots in relation to a given workflow

## Job requirements

```
RequestGPUs = 1
RequiresGPU = 1
...
Requirements =
CUDACapability >= 3 &&
CUDARuntime = "11.4" &&
GPUMemoryMB = 8000 && ...
...
```

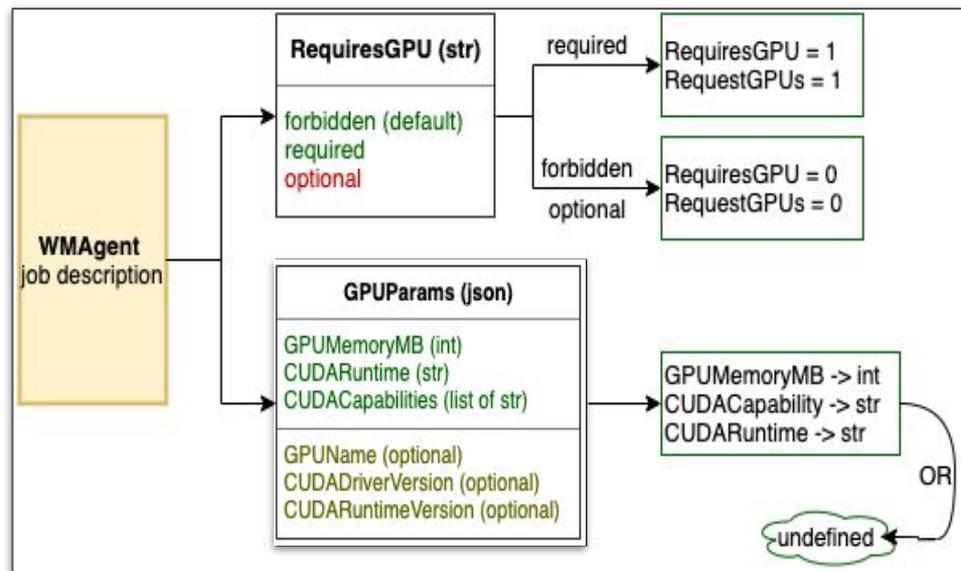
Collector  
+  
Negotiator

```
...
Machine.CUDACapability in Job.CUDACapability
Job.CUDARuntime in Machine.CMS_CUDA_SUPPORTED_RUNTIMES
Job.GPUs <= Machine.GPUs
...
```

## Machine attributes

```
CPUs = 8
GPUs = 2
CUDACapability = 8.0
CUDAClockMhz = 1410.0
CUDAComputeUnits = 108
CUDACoresPerCU = 64
CUDADeviceName = "NVIDIA A100-PCIE-40GB"
CUDADriverVersion = 11.3
CUDAECCEEnabled = true
CUDAGlobalMemoryMb = 40536
CUDAMaxSupportedVersion = 11030
CMS_CUDA_SUPPORTED_RUNTIMES =
10.1,10.2,11.0,11.1,...
...
```

- WM layer acts as interface between user requests and the actual **job creation** into the HTCondor *schedd* queues
  - Map user request into a job jdl with attributes to be used in both matchmaking stages
- Agreement on the introduction of a new **set of key/value pairs to be employed by any GPU workflow**
  - mapped into a **total of 5 htcondor job attributes (plus 3 optional)**



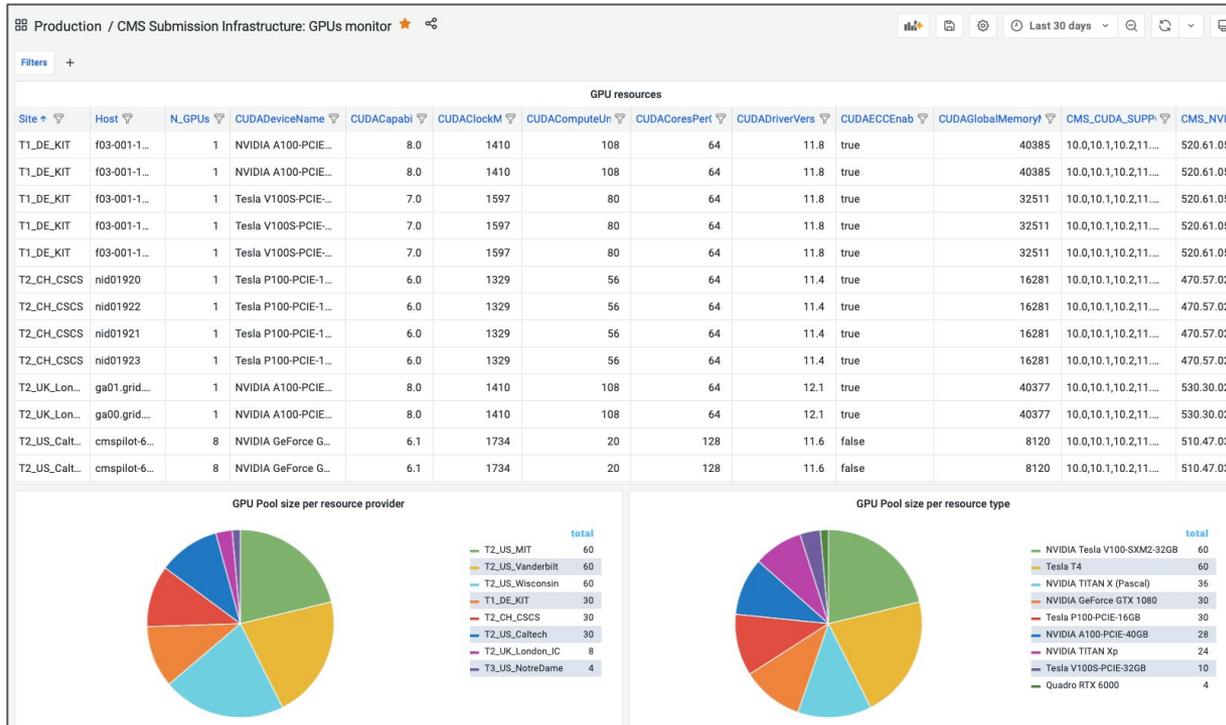
GPU support in [WMCore](#) and [CRAB](#)

- With regards to **how CMS jobs relate to GPUs**, while the situation is evolving, three use cases are considered:
  - Job must use GPUs
  - Job can use GPUs
  - Job can only use CPUs
- **Access and use of GPUs** under control of two main attributes in the job jdl:
  - **RequiresGPU**: Defines whether or not GPU pilots will be requested by the first (GlideinWMS FE) matchmaking
  - **RequestGPUs**: Number of GPU resources to be matched in the second (HTCondor) matchmaking
- The three use cases are covered:
  - `RequiresGPU = 1 && RequestGPUs>q` will trigger GPU pilot, then match the slot
  - `RequiresGPU = 0 && RequestGPUs>q` will not trigger GPU pilots but CAN match already available GPU slots.
    - Need to discuss preference for CPU+GPU over CPU slots
  - `RequiresGPU = False && RequestGPUs=q` for purely CPU task

# Availability of GPUs in CMS SI

# Advertising GPUs already available to CMS

- Multiple CMS WLCG sites already kindly making their GPU resources available for CMS use
- GPU inventory: SI regularly scanning the Global Pool for GPU resources, collecting their availability and properties, then published in the [GPU monitoring table](#)
  - Recent (e.g. last 30d) results should inform users about “what is available and where”

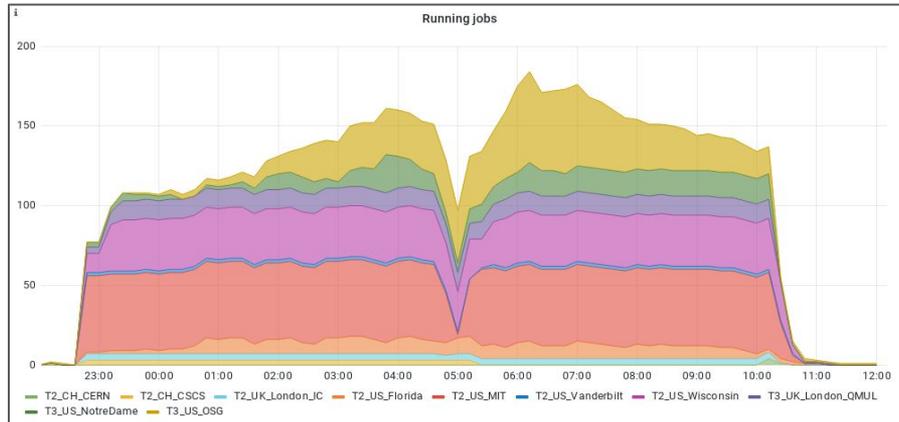


# Scale test for GPUs in CMS SI

## Executed a scale and performance test on GPUs in the CMS Global Pool

- Injected about 15k test jobs on the Global Pool, targeting any available GPU for 24h: **match as many GPUs as possible**, check how many and where they are and what type, etc

**Results:** Achieved a peak allocation of over **150 GPUs in parallel** in the Global Pool



## About 230 distinct opportunistic GPUs discovered during the test

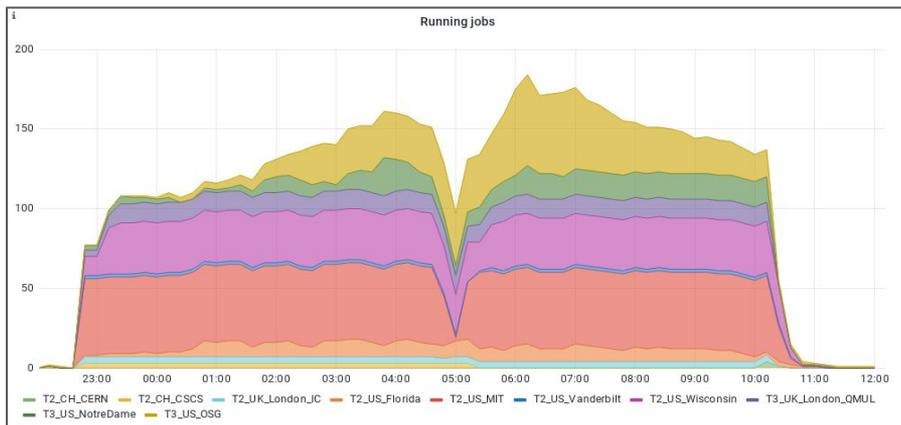
Site	GPU_Type	N_GPUs
T2_CH_CERN	Tesla V100S-PCIE-32GB	8
T2_CH_CERN	Tesla T4	8
T2_CH_CERN	Tesla V100-PCIE-32GB	5
T2_CH_CSCS	Tesla P100-PCIE-16GB	4
T2_UK_London_IC	NVIDIA A100-PCIE-40GB	4
T2_US_Florida	NVIDIA GeForce RTX 2080 Ti	72
T2_US_MIT	NVIDIA Tesla V100-SXM2-32GB	3
T2_US_Purdue	NVIDIA Tesla T4	4
T2_US_Vanderbilt	NVIDIA TITAN Xp	3
T2_US_Wisconsin	Tesla T4	32
T3_UK_London_QMUL	NVIDIA A100-PCIE-40GB	12
T3_US_NotreDame	Quadro RTX 6000	15
T3_US_NotreDame	Tesla V100-PCIE-32GB	2
T3_US_OSG	Tesla V100-PCIE-32GB	31
T3_US_OSG	Tesla P100-PCIE-16GB	9
T3_US_OSG	Quadro RTX 5000	8
T3_US_OSG	Tesla V100-PCIE-16GB	3
T3_US_OSG	Tesla P100-PCIE-12GB	2
T3_US_OSG	NVIDIA GeForce GTX 1080 Ti	2
T3_US_OSG	Quadro RTX 8000	2
T3_US_OSG	NVIDIA GeForce GTX 1060 6GB	1

# Scale test for GPUs in CMS SI

## Executed a scale and performance test on GPUs in the CMS Global Pool

- Injected about 15k test jobs on the Global Pool, targeting any available GPU for 24h: **match as many GPUs as possible**, check how many and where they are and what type, etc

**Results:** Achieved a peak allocation of over **150 GPUs in parallel** in the Global Pool



## About 230 distinct opportunistic GPUs discovered during the test

Site	GPU_Type	N_GPUs
T2_CH_CERN	Tesla V100S-PCIE-32GB	8
T2_CH_CERN	Tesla T4	8
T2_CH_CERN	Tesla V100-PCIE-32GB	5
T2_CH_CSCS	Tesla P100-PCIE-16GB	4
T2_UK_London_IC	NVIDIA A100-PCIE-40GB	4
T2_US_Florida	NVIDIA GeForce RTX 2080 Ti	72
T2_US_MIT	NVIDIA Tesla V100-SXM2-32GB	3
T2_US_Purdue	NVIDIA Tesla T4	4
T2_US_Vanderbilt	NVIDIA TITAN Xp	3
T2_US_Wisconsin	Tesla T4	32
T3_UK_London_QMUL	NVIDIA A100-PCIE-40GB	12
T3_US_NotreDame	Quadro RTX 6000	15
T3_US_NotreDame	Tesla V100-PCIE-32GB	2
T3_US_OSG	Tesla V100-PCIE-32GB	31
T3_US_OSG	Tesla P100-PCIE-16GB	9
T3_US_OSG	Quadro RTX 5000	8
T3_US_OSG	Tesla V100-PCIE-16GB	3
T3_US_OSG	Tesla P100-PCIE-12GB	2
T3_US_OSG	NVIDIA GeForce GTX 1080 Ti	2
T3_US_OSG	Quadro RTX 8000	2
T3_US_OSG	NVIDIA GeForce GTX 1060 6GB	1

# Further Challenges in the use of GPUs

# Further Challenges in the use of GPUs

- Use of GPUs in CMS MC production **multi-step workflows**?
  - Multi-step jobs (multiple cmsRun executions with different settings per job), not all of them capable of making use of the GPUs
- **Benchmarking** of GPUs (as soon as HEPsScore is available)
  - A requirement for **pledge definition** and **resource acquisition**
- **Predictable workflow runtimes**, a key parameter for an efficient matchmaking of jobs to slots
  - Requires benchmarking
  - Hard because of the high diversity among GPU types
- **GPU usage accounting**
  - Also requires GPU resource benchmarking
  - Could use HTCondor's GPUsAverageUsage as proxy
    - Cron job uses the NVIDIA driver and tools libraries to query statistics on all of the GPUs
    - Generate usage **report** back to the slot info and **payload job classad**

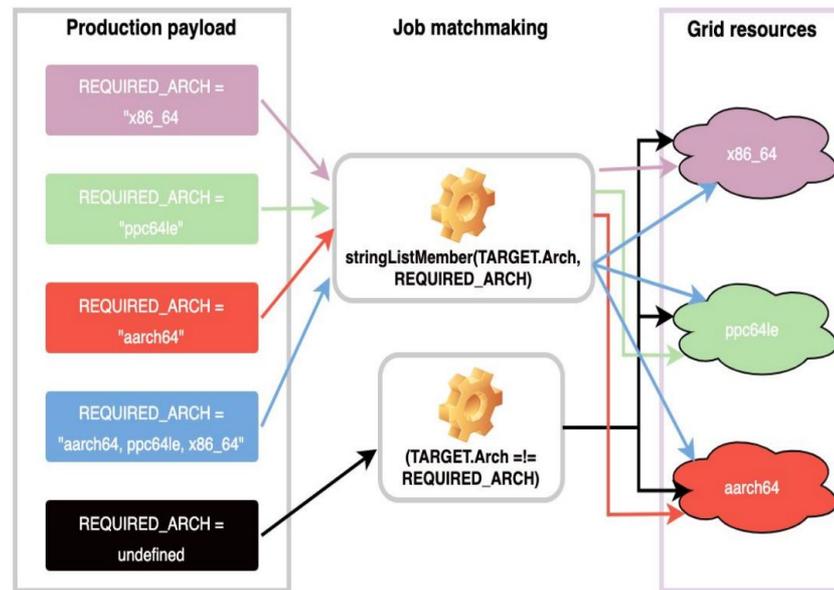
# Integration of non-x86 architectures

The **CMS SI and WM systems** were originally **built and deployed assuming the x86\_64 standard**

- **The WM and SI underwent adaptation** to allow access and use of other architectures
  - **Power9** was the first non-x86 platform available at scale for CMS (at CNAF/CINECA)
    - Initially tested with manually launched pilots (ad-hoc compiled HTCondor startd)
    - no singularity container
  - By mid 2022, HTCondor support for ARM (aarch64) and Power PC (ppc64le) CPU architectures available
    - Startds added to the GlideinWMS pilot factory, can now access these resource types with GlideinWMS pilots
- **Power9 arch is now fully commissioned** and its CMSSW has been physics-validated.
- **Next is ARM:**
  - CERN granted access to a few ARM machines to be accessed as Grid resources, and again at CNAF
  - Ongoing **integration** and **physics validation** tests
- RISC-V architecture by the HL-LHC era?

**Resource architecture** has been turned into a configurable **parameter**, to be used in the two CMS SI matchmaking phases:

- Payload jobs describe their resource requirements, which **may** include matching a **specific or multiple architectures**
- Provisioning of the corresponding resources via the HTCondor Global Pool
- HTCondor second matchmaking step



```
Requirements = (stringListMember(TARGET.Arch,"ppc64le,X86_64")) && (TARGET.OpSys == "LINUX") && (TARGET.Disk >= RequestDisk) && (TARGET.Memory >= RequestMemory) && (TARGET.Cpus >= RequestCpus) && (TARGET.HasFileTransfer)
```

# Conclusions

# Conclusions

- **Resource heterogeneity** in HEP computing is a **key element** moving forward
  - CMS is adapting its WM and SI systems to properly manage new resource types and providers and put them to good use
- Heterogeneous collection of resources: **no standard slot definition exists**
  - Workload scheduling requires careful description of slot properties and workload requirements for effective matchmaking
- With regards to **GPUs**, still **opportunistic** but a sizable number and variety **already available** in the CMS Global Pool
  - Detailed inventory of available GPUs produced to promote GPU resource exploitation by CMS users
  - **CMS scientists and central production can now use GPUs for their workflows on the Grid**
- Non x86 CPU architectures **integration** to CMS computing is **ongoing**
  - Power9 fully integrated and validated
  - ARM being commissioned
- Challenges ahead
  - Efficient execution of CMS multi-steps jobs on CPU/GPU heterogeneous resources
  - Benchmarking and accounting required for realistic usage at scale in the WLCG context

# Acknowledgements

Projects FPA2016-80994-C2-1-R,  
PID2019-110942RB-C21, BES-2017-082665  
and PID2020-113807RA-I00 funded by:



**Ciemat** Centro de Investigaciones  
Energéticas, Medioambientales  
y Tecnológicas

 **cfp**  
CIEMAT  
física de partículas

 **PIC**  
port d'informació  
científica

# Backup Slides

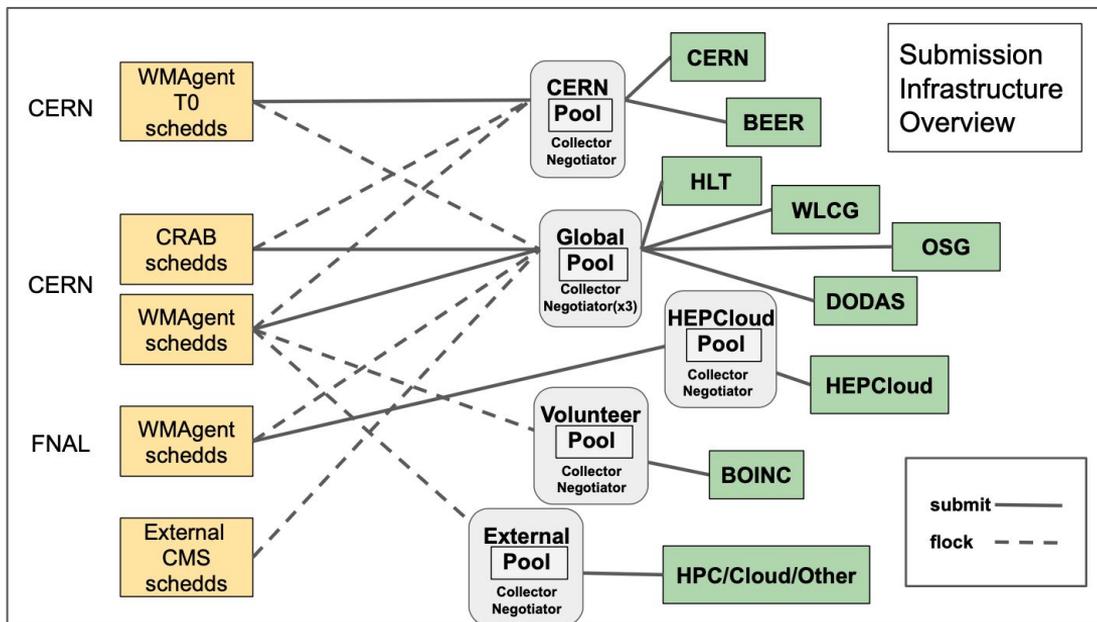
The **CMS Submission Infrastructure** team in CMS Offline and Computing is in **charge** of operating a set of federated HTCondor pools which aggregates **resources from 70 Grid sites, plus non-Grid resources, where reconstruction, simulation, and analysis of physics data takes place**

## The challenges:

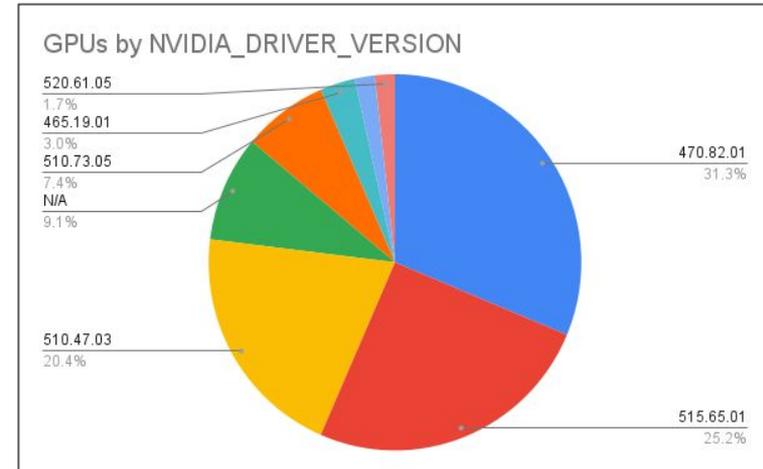
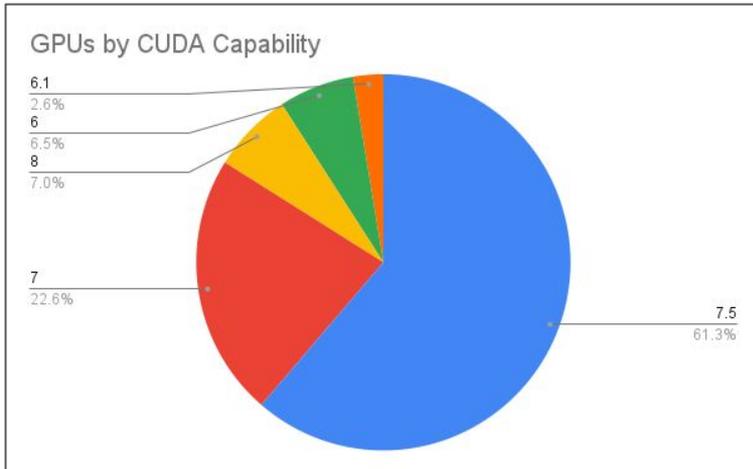
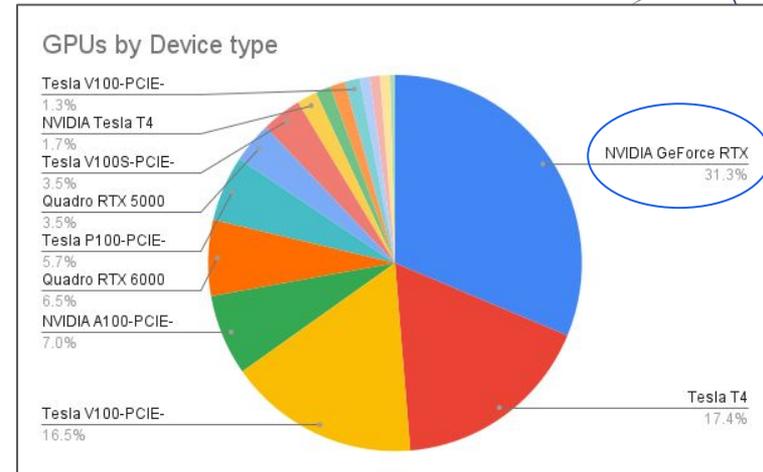
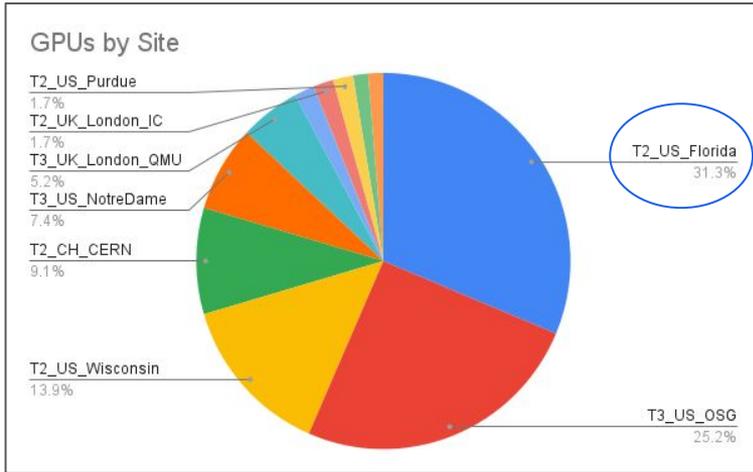
- Operate our infrastructure managing an ever growing collection of computing resources
- Use all of our resources efficiently, maximizing data processing throughput
- Enforce task priorities according to CMS research programme
- Connecting new and more diverse resource types (including non-x86 architectures and GPUs) and sources (WLCG and OSG, HPC, Cloud, volunteer)



The topic of this talk



# GPU scale and performance test



# GPU scale and performance test

- Execution time as a function of site and GPU type: a factor 10x variance in results observed overall

