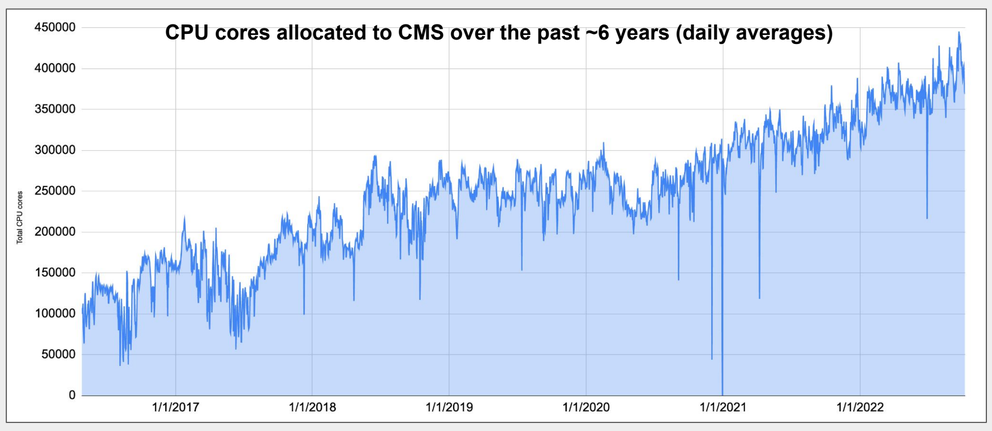
Evolution of the CMS Submission Infrastructure to support heterogeneous resources in the LHC Run 3

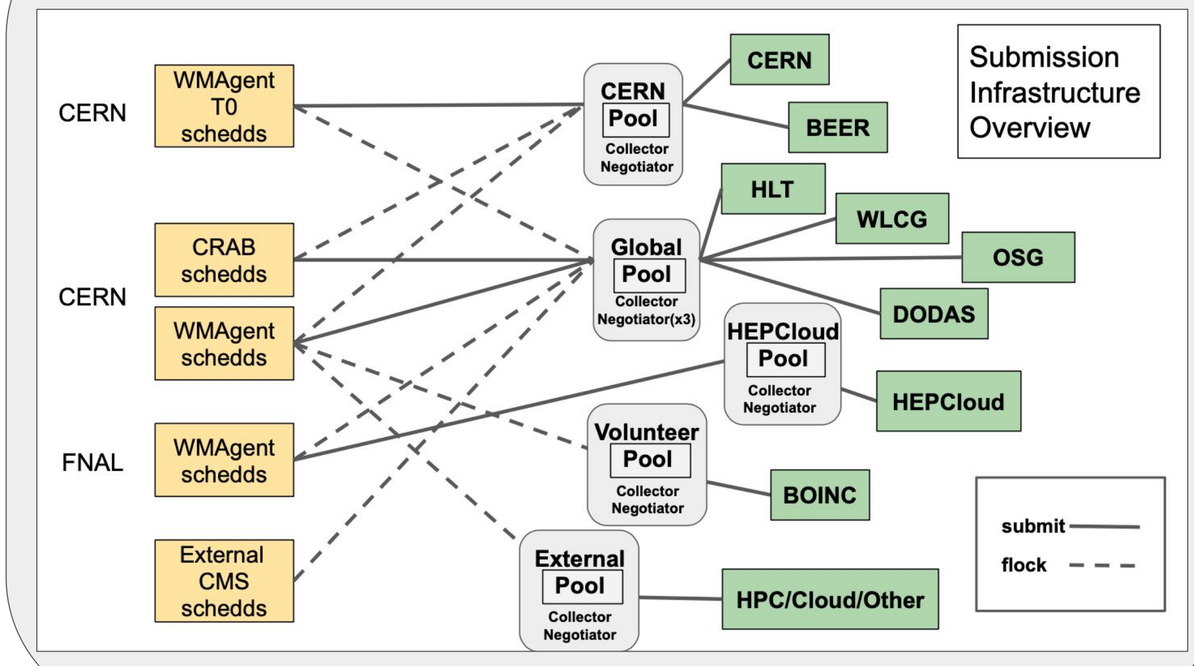
A. Pérez-Calero Yzquierdo[1], Edita Kizinevic[2], Farrukh Aftab Khan[3], Hyunwoo Kim[3], Marco Mascheroni[4], Maria Acosta Flechas[3], Nikos Tsipinakis[2] and Saqib Haleem[5], on behalf of the CMS collaboration

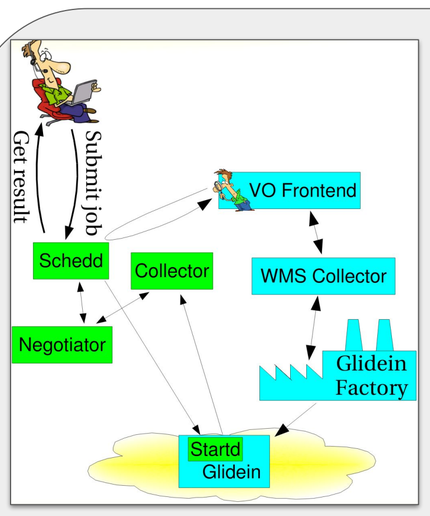
1. CIEMAT and PIC (ES), 2. CERN, 3. Fermi National Accelerator Lab. (US), 4. Univ. of California San Diego (US), 5. National Center for Physics (PK)

- **The CMS Submission Infrastructure:** team in CMS Offline and Computing in **charge** of:
  - Organizing **HTCondor** and **GlideinWMS** operations in CMS,
  - Maintaining a **Global Pool, an infrastructure of distributed compute resources where reconstruction, simulation, and analysis of physics data takes place**
  - Communicate CMS **priorities to the development teams** of glideinWMS and HTCondor
- **In practice:**
  - We operate a set of federated HTCondor pools which aggregate **resources from 70 Grid sites, plus non-Grid resources**
  - We regularly hold **meetings with** HTCondor and glideinWMS **developers** where we discuss current **operational limitations, new feature requests and future scale requirements**



Federated HTCondor pools





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

GPU resources just follow the same general logic

All GPUs in use by CMS via the Submission Infrastructure are opportunistic in nature so far, not pledged

- **GPU resource description for first matchmaking coded in pilot factory entries**
  - **Limited information about the available GPUs (CEs that allow access to GPUs, queue names, etc)** and **statically configured**
- Once a pilot starts execution on remote resources, **GPU properties are updated to the slot classad**:
  - Use **condor_gpu_discovery** tool to retrieve most of the matchmaking attributes
  - Custom CMS script for **CMS_CUDA_SUPPORTED_RUNTIMES**

Access and use of GPUs under control of two main attributes in the job jdl:

- RequiresGPU = 1 && RequestGPUs>0, will trigger GPU pilot, then match the slot
- RequiresGPU = 0 && RequestGPUs>0, will not trigger GPU pilots but CAN match already available GPU slots
- RequiresGPU = 0 && RequestGPUs=0 for purely CPU task

Matchmaking jobs to GPUs: A very **inhomogeneous set of resources**, requires a **careful and detailed job and slot description** in order to select the correct slot for each task

Job requirements

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

Collector + Negotiator

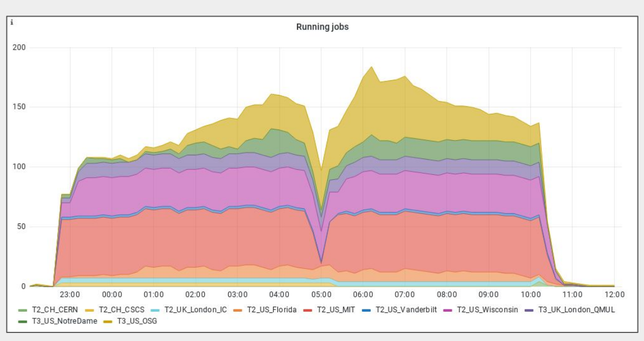
```
...
Machine.CUDACapability in Job.CUDACapability
Job.CUDARuntime in
Machine.CMS_CUDA_SUPPORTED_RUNTIMES
Job.GPUs <= Machine.GPUs
...
```

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
CUDAECCEnabled = true
CUDAGlobalMemoryMB = 40536
CUDAMaxSupportedVersion = 11030
CMS_CUDA_SUPPORTED_RUNTIMES =
10.1,10.2,11.0,11.1,...
CMS_NVIDIA_DRIVER_VERSION = 515.48.07
```

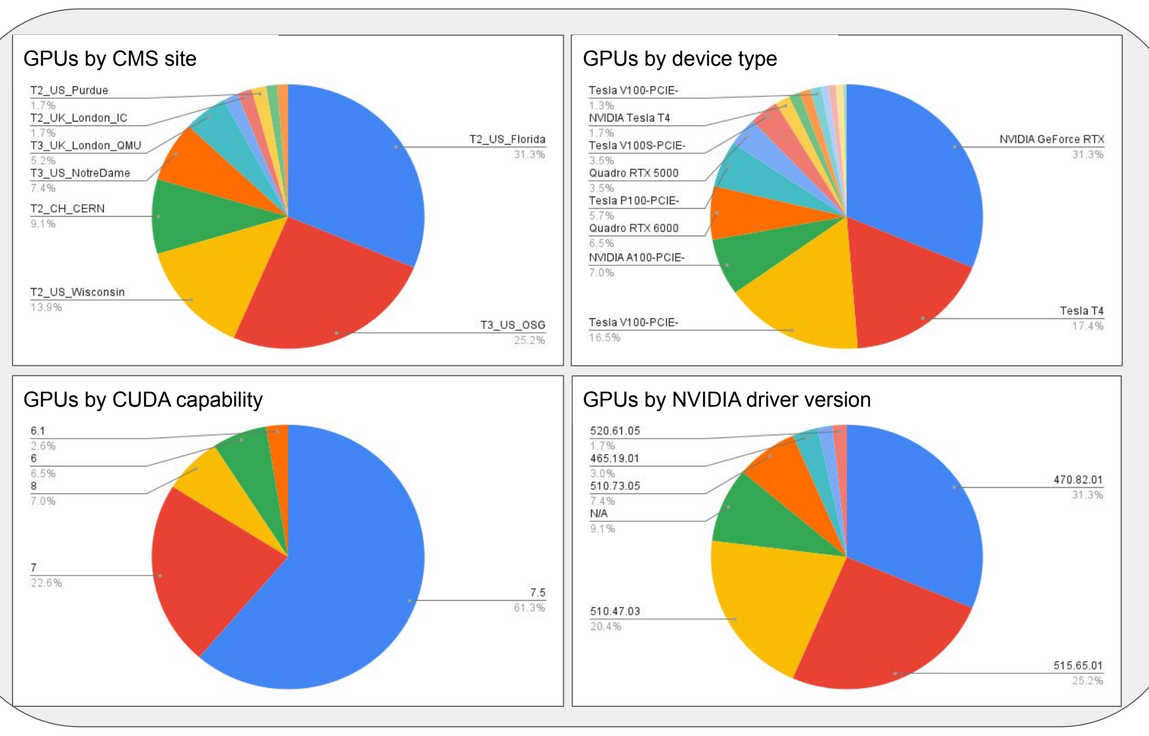
Running a scale and performance test on GPUs in the CMS Global Pool

- 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
- Used a simple TensorFlow multiple (10k x 10k, float16) **random matrix multiplication** script as GPU payload
- Recorded total **execution time**, correlated to accelerator performance
- Results: Achieved a peak allocation of over **150 GPUs in parallel** in the Global Pool

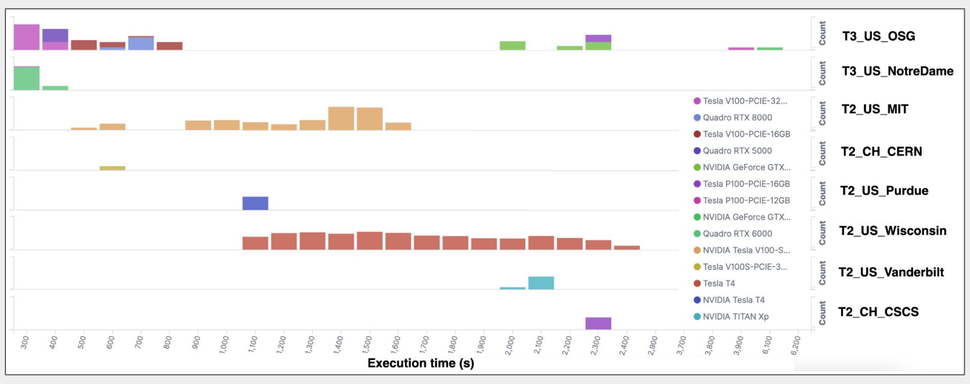


About 230 distinct opportunistic GPUs discovered during the test

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



Test job execution time as a function of site and GPU type



Further Challenges in the use of GPUs

- **Benchmarking** of GPUs is a requirement for pledge definition and resource acquisition
  - Predictable workflow runtimes, a key parameter for an efficient matchmaking of jobs to slots
    - hard now because of high heterogeneity among GPUs
- **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 to generate resource usage report

Conclusions and Future work

- Despite still being **opportunistic**, **a sizable collection of GPUs is already available in the CMS Global Pool**
  - Distributed over a number of sites, so far mainly in the US
  - Non-homogeneous collection of resources, no standard definition of a "GPU slot" exists
  - Scheduling workflows on GPUs relies on careful description of slot properties and workload requirements
- **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**
  - **Detailed inventory of available GPUs** required to promote GPU resource exploitation by CMS users
  - Complex environment for operations, valuable debugging tool in **condor_ssh_to_job**, to be enabled in CMS submission infrastructure