

# Porting a GPU-enabled HLT workload to HEP-WORKLOADS

Andrea Sciabà

# Integration process

- Received from Tommaso B. a script that runs a GPU-enabled CMS workload
  - Set up the shell environment
  - Retrieve and compile the source code
  - Run cmsRun
- The workload is simply an example of an HLT job on MC ttbar events and has no pretense of being “the” CMS GPU benchmark!
  - It can run with and without a GPU
  - It requires an input file of 100 events, but it can run over them as many times as desired
- The **conversion** into an orchestrator and a CMSSW configuration script was **very easy**
  - Performed by analogy with cms-reco-run3, conceptually very similar

# Building the image

- No issues, it worked at the second attempt (the first failed because of “no space on device”)
  - <https://gitlab.cern.ch/hep-benchmarks/hep-workloads/-/tree/master/cms/hlt>
- No GPU-specific metrics in the results JSON for now
  - The score is still simply the event throughput

# GPU score and metrics

- We need to have some metrics to quantify the impact and the utilization of the GPU. Some candidates:
  - *num\_GPUs*: the number of GPUs visible to cmsRun
  - *speedup*: the ratio between the throughput obtained when using also the GPU and the one obtained without using the GPU
    - It requires to run cmsRun twice in the orchestrator
    - The main score should be the GPU+CPU throughput (the CPU-only one is used only to calculate the speedup)
  - *GPU utilization (%)* obtained from nvidia-smi
    - PrMon can be used for this purpose

# Caveats

- When the workload is run in  $> 1$  concurrent copies, GPU sharing becomes an issue to be considered
- By default, sharing will be done on a round-robin basis, which is very inefficient
- Need to run Nvidia MPS (Multi-Process Service) to allow concurrent processes to use the GPU at the same time
  - It can be launched even by a non-privileged user
  - Need to understand where it's best to launch it (it should be probably outside of any specific workload assuming that there may be several different GPU workloads running)
  - Or, assume that it is running on the host being benchmarked – probably better

# Known issues

- The workload was tested to be able to gracefully use only the CPU if there is physically no GPU in the system
- However, if *docker run* is not run with the *--gpu all* option, it aborts because of missing Nvidia libraries
  - Not what I would expect – to be understood
- The GPU utilization measured by *nvidia-smi* is extremely low (~1%)
  - But a new workload I just received shows a healthier 94%, so it's not a bug
- Prmon reports zero values for the GPU metrics
  - Investigating – maybe there is a problem with running it in a container?
  - **UPDATE: it works if I do *docker run --pid=host ...***

# Next steps

- Merge the code that implements the num\_GPUs and speedup metrics
- Move to the newer workload (which uses the GPU at an unrealistically high level but it's good for a prototype)
- Release a public Docker image
- Add the measurement of the GPU utilization via prmon
- Add the option to run MPS?

# Conclusion

- The goal was to **prepare the machinery** to run a GPU-enabled CMS workload in HEPSCORE
- The workload that will be eventually be used for benchmarking is still likely **several months away** and it depends on what CMS chooses to be a realistic benchmark