CMS Benchmark for GPUs

A. Bocci, A. Sciabà, T. Boccali

HEPscore workshop
19-20 September 2022

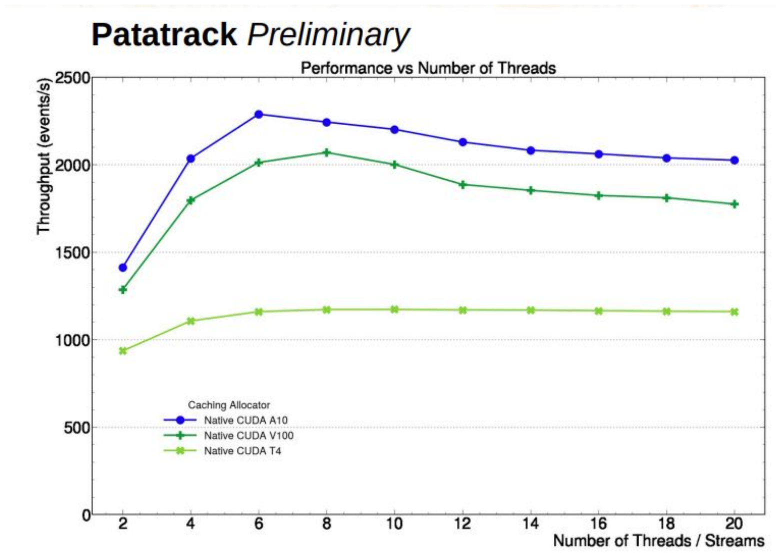
Introduction

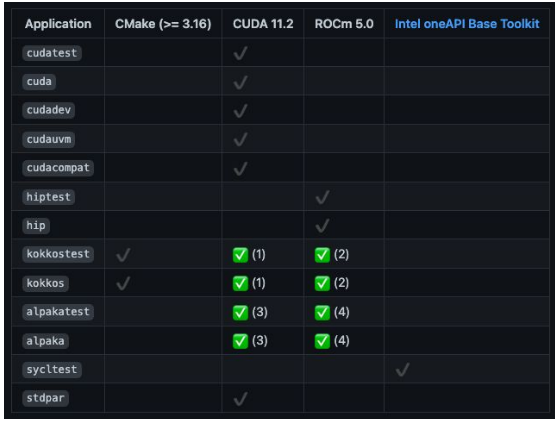
- Usage of GPUs in CMS
- Goals of a GPU benchmark in HepScore
- Description of the prototype
- How to run the CMS GPU benchmark
- Issues found (and solved)
- Future plans

(forgive the inability to answer to some questions, the talk was initially prepared by Andrea who is on sick leave atm)

Usage of GPUs in CMS

- Originated from the Patatrack project, an incubator for GPU-related R&D (algorithm porting, demonstrators, adoption in CMSSW…)
- Currently used in production at HLT:
  - full chain of the Pixel reconstruction (local reconstruction, calibrations, pixel-only track and vertex reconstruction)
  - calorimeters local reconstruction (ECAL and HCAL)
- More under development for offline and online:
  - primary vertex reconstruction
  - particle flow reconstruction
  - HGCAL reconstruction (targeting Phase-2)
- Also maintaining a separate, standalone platform
  - simplified version of the CMSSW framework
  - playground to prototype new algorithms, support different hardware, investigate *performance portability* solutions
  - an old version was used as a GPU benchmark in the HEPiX benchmarking working group years ago





Application	CMake (>= 3.16)	CUDA 11.2	ROCm 5.0	Intel oneAPI Base Toolkit
`cudatest`		✓		
`cuda`		✓		
`cudadev`		✓		
`cudauvm`		✓		
`cudacompat`		✓		
`hiptest`			✓	
`hip`			✓	
`kokkostest`	✓	✅ (1)	✅ (2)	
`kokkos`	✓	✅ (1)	✅ (2)	
`alpakatest`		✅ (3)	✅ (4)	
`alpaka`		✅ (3)	✅ (4)	
`sycltest`				✓
`stdpar`		✓		

GPUs in the CMS HLT

- running in production today !
- 200 HLT nodes
  - 2 × AMD EPYC “Milan” 7763 processors, with 64 cores / 128 threads each (280W TDP)
  - 2 × NVIDIA T4 (70W TDP)
- software configuration
  - 8 concurrent jobs with 32 threads each
  - each group of 4 jobs is pinned to one CPU socket and share a single GPU
  - *not using* the NVIDIA Multi-Process Service (MPS), due to stability issues
- actual speedup in production is 74%
  - it would be 84% with NVIDIA MPS !

Description of the prototype #1

- Main interest is NOT YET measuring the speedup from the point of view of procurement (usage still too low for offline typical workloads)
    - Final goal: certify that a given CPU+GPU is worth XX HepScore for a given use case, so site YY can decide to acquire GPU equipped systems (“one day”)
- … but testing the machinery of GPU enable workflow as early as possible
    - See how one can attach the GPU hardware inside the container
    - See if we have a clean way to use the OS drivers
    - Establish a proof of concept CPU/GPU comparison: run @CPU and @CPU+GPU and compare result (“speedup”)

Description of the prototype #2

- A very first CMSSW configuration was based on the full HLT prototype from late 2021
  - The speedup was modest, and not very interesting to show differences
- Replaced by another configuration that artificially inflates GPU utilization: the pixel track reconstruction in HLT
  - Not a realistic HLT workload, but very measurable speedups
- **The current workload must be considered as a placeholder for the future CMS GPU benchmark**
- Only Nvidia GPUs are supported (currently)
  - In principle with CMSSW13 we plan to have Alpaka, it would be the next step; we currently use CMSSW12

Default parameters

- Threads: **-t 4**
- Copies: **no. of cores / threads**
- Events per thread: **-e 25000** (so standard process 100k Events)
- The job can loop over the few input RAW events, with memory caches, so the total number of events can be arbitrarily large (only case among the CMS workloads, needed since HLT is by definition sub-second, for a decent runtime)
- Not done atm wrt to production systems:
    - Attach GPUs to processes, via " CUDA_VISIBLE_DEVICES", needed only in multi GPU configurations

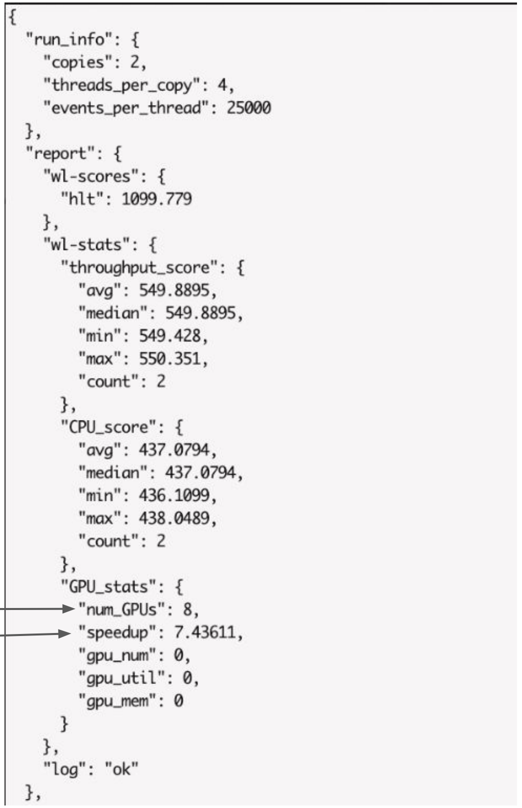
How to run the CMS GPU benchmark

- Find a node with singularity (or docker) and an Nvidia GPU
  - Set SINGULARITY_CACHEDIR as desired
  - Run the workload with singularity

    ```
    singularity run --nv -C -B /tmp:/tmp -B /results:/results
    docker://gitlab-registry.cern.ch/hep-benchmarks/hep-workloads/cms-hlt-bmk:latest
    ```
- The WL will run a first time and use a GPU if available
- If a GPU was found, the WL will run a second time after setting CUDA_VISIBLE_DEVICES="", which forces it not to ignore the GPU
- The final scores refer to the GPU run (if there is a GPU)
- The second run is used <u>only</u> to calculate the speedup!
- If you want to run the WL without the GPU even if the node has one, just omit the `--nv` option

Some preliminary runs

- The container machinery works out of the box, on every node we tested it
- The magic comes from the –nv option in singularity, which works like a charm
  - On top of that, CMSSW comes with CUDA libraries in case the OS are not ok, so in principle only the driver needs to be “good enough” (slight fix needed)
- Largest scale test on a Nvidia DGX-1 (256 cores, 8x A100)
  - Limited run, only 2 processes (whole machine difficult to grab), with 8x A100
  - Speedup calculated to 7.4, take it as a random number at this point (there is no GPU-to-process attachment); it just shows it works



```
{
  "run_info": {
    "copies": 2,
    "threads_per_copy": 4,
    "events_per_thread": 25000
  },
  "report": {
    "wl-scores": {
      "hlt": 1099.779
    },
    "wl-stats": {
      "throughput_score": {
        "avg": 549.8895,
        "median": 549.8895,
        "min": 549.428,
        "max": 550.351,
        "count": 2
      },
      "CPU_score": {
        "avg": 437.0794,
        "median": 437.0794,
        "min": 436.1099,
        "max": 438.0489,
        "count": 2
      },
      "GPU_stats": {
        "num_GPUs": 8,
        "speedup": 7.43611,
        "gpu_num": 0,
        "gpu_util": 0,
        "gpu_mem": 0
      }
    },
    "log": "ok"
  },
```

Some other tests, atm outside the container

- Tested on a CINECA M100 node (Power9 + 4x V100) → ok
- Tested on an Nvidia devel kit (Ampere ARM Altra Q80 + 2x A100) → ok

For these architectures, it is only a matter of preparing proper images (unless they are already there and I do not know…)

Issues found (for the record)

- CMSSW needs at least some stubs for the Nvidia drivers, taken from CVMFS, even on GPU-less nodes
- When building the image on a node with a GPU, the Nvidia drivers were taken from the node and the stubs were not picked up by the CVMFS shrink-wrap mechanism
  - As a result, the WL crashed when the image was run on GPU-less nodes
- Solution: force the inclusion of the driver stubs

Future plans & conclusions

- Refine the workload parameters' defaults
  - For example, better distribute load between GPUs if several, and possibly use MPS (but CMS in production does not, so….)
- Update the workload configuration or even produce multiple configurations. Examples:
  - A realistic HLT configuration
  - A configuration to represent GPU usage in offline reconstruction
  - A configuration specifically intended to measure GPU power, reducing CPU usage as much as possible
- **But, as a proof-of-concept test, what we have is already valuable if run everywhere HepScore runs, in order to collect statistics / see possible problems**
  - **Needless to say, we are perfectly ok in setting this WL's weight to 0 in the final score!**