CMS workloads (and needs)

outline

- In a single slide, the basic needs we have expressed
- The HepScore solution: our proposed workloads for the production level benchmark
- The future / work in progress

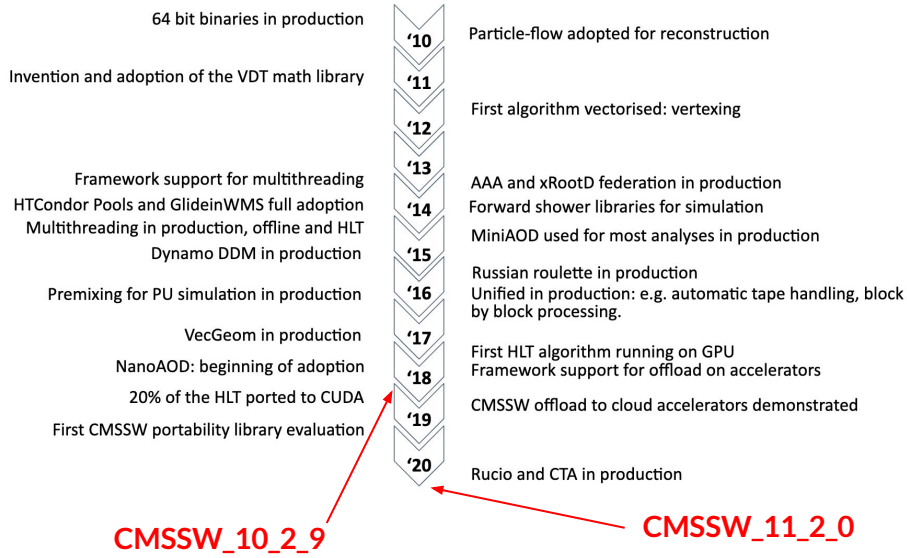
The ideal benchmark (suite + benchmark)

(on top of what HS06 – which served us well for more than a decade – offers)

1. Openly distributed (no license needed)
2. Scaling linearly with our workloads, now and for the CPUs coming in the next decade
3. Tunable for specific use cases (not for pledges!) – for example in single VO environments
4. Open to the technologies we think may become relevant (GPUs, FPGAs, ….)
5. Easy to deploy even in testbench setups (no network, bare systems)
6. Measuring full systems, not only the CPU (fast / slow drives, memory setups, …)
7. Automatically building an as-large-as-possible DB of configurations; (no more handwritten twikis on best effort)
8. DB as complete as possible: no only CPU but also type of disks, memory setups, GPUs, buses available to accelerators, versions of most important drivers
9. Fast enough to be practical (no >> 1 day runs)
10. Working in realistic situations (full machine, no turbo boost, …)

The CMS workloads

- As close as possible to what we really run on our distributed system
  - While details may vary, we mostly run separated processes for (GEN+)SIM, DIGI and RECO - as in the workloads
- Use a typical generator, which does not need access to large gridpacks ➡ Pythia
- Use a Run3 Configuration / Setup / Pileup
- Use a newish CMSSW release (11, from last year), after evaluation of whether we would miss anything from going to something newer we might still need for the “experimental” part, see later)



CMSSW 11 includes all the features CMS has been pushing in the last ~ 10 years. No revolution expected with CMSSW 12 (but see the GPU talk)

The current proposal

- Preload the container with a Premixed PU=35 Run3 library
  - There was also the option to generate it on the fly, but this is not what we typically do in production
- Generate 1000 (tunable for total test speed, linear scaling) Pythia TTbar events (“SIM”)
- Mix them with the PU library (“DIGI”)
- Reconstruct them, and produce AODSIM and MINIAODSIM (“RECO”)
- Add some mild data quality monitoring (in production we do that for data reprocessing and for a fraction of Monte Carlo)
- Execution at 8 threads would be our preference (it is what we do in production since 2017)

Typical running times in an early test setup:

- x86 (a random lxplus node): 40 min
- IBM Power8 (ibm minsky): 55 min
- AARCH64 (thunderX): 140 min

YES! CMSSW is already multiarch since long; we are now ready for WMS (WMAgent) for non x86 archs, and almost ready for multiarch support (a single workflow can use more than 1 architecture)

We are closing on physics validation on Power, to be able to use some large HPCs lying around; but also to define the procedure to validate a new architecture (in practice, not different from the validation procedure we do everytime we change gcc, Geant4, or when we switched from 32 to 64 bits)

A more future oriented look

- CMSSW supports “external work” (GPUs, remote facilities, ...) since at least 3 years
- Currently GPUs are used in production at the HLT, and are measured to boost by 74% our online processing
  - This is not ad-hoc code, fully integrated in CMSSW and usable also offline
- We have plans to increase consistently the fraction of offloaded code from now to Run4
- One of the possible configurations we will evaluate on the GRID is to autodiscover hardware at runtime, and use best sw modules at any moment
- We would like to start experiencing how a future version of HepScore would behave on the large variety of systems which can include accelerators (starting from Nvidia GPUs)
- See Andrea Sciaba’s talk on this, based on CMSSW 12
- Our proposal for the moment would be to execute the “HLT” payload (which is not a production level HLT, but a setup chosen to maximise the offloading to GPU) with final weight “0” on the HepScore result, but in a way to still collect results.
  - Also Power+GPU and ARM+GPU have been tested, we do not lose any universality!

By the time HepScore 2 is out (Run4? LS3?) heterogeneous offloading could become part of the “standard” benchmark

Put credit where it is!

- CMSSW is a fantastic tool, adaptable to many configurations. As such, it was easily adapted to the needs of an HepScore; we could not thank more the **hundreds of developers** who took part in developing it since 2004
- **David Lange** was selecting the best workflows, and providing proper configurations
- **Andrea Sciaba'** was packing the payloads in a self sufficient container, as needed for the execution
- **Andrea Bocci** provided an ad-hoc HLT setup, and hints on how to allow for containerized execution
- **OpenLab, TechLab and CINECA** have provided heterogeneous hardware to test for Power and ARM setups

OK ("once per Run or once per Run+LS")

a) The selected composition of HEPscore (ie. the selected HEP workloads), won't change for at least 3-5 years.
b) Some workloads of your Experiments could be excluded from the selected mix, if other workloads (from other Experiments) are representative enough and can run faster.
   i) An example is the case of the "SIM" workloads, that are all based on Geant4 and some of them are correlated. Therefore not all the "SIM" workloads need to be included in HEPscore to represent all the Experiments.
c) What is your opinion about adopting for HEPscore a workload mix that is
   i) Equally weighted
   ii) Weighted on the basis of the percentage of grid jobs running gen, sim, reco workloads
   iii) Weighted on the percentage of grid jobs per experiment
   iv) Do you have any other weight to propose?

Difficult answer. We would like to be convinced (with numbers) that the difference between 1-4 is minimal

In principle OK, but please do not exclude those from major experiments! (and: not all G4 are equale: vecgeom, fastsim algorithms, tricks, ...)