CMS input to HEPscore TF

D.Lange, T.Boccali
What we have now

- What is in HepScore now is already quite advanced: 

- 3 WFs:
  - GEN-SIM: process TTbar from Pythia + Geant4 (with simplified PU and beamspot etc wrt to production)
  - DIGI: Digitization with Premixing (using a pre-produced Premixed library - a few files in the containers)
  - RECO: process Reconstruction

- Technical details
  - OS: SLC6; ARCH: amd64; COMPILER: GCC 7.0.0
  - CMSSW_10_2_9 (released Dec 2018)
  - 2018 CMS/LHC configurations (beam spot, pileup, ...)
  - Runs @ 4 threads all the steps, all MT (no internal MP)
Which are the differences wrt to a more recent release (late 2020)?

- We use CC7
- We use GCC 9.0.0
- Release is 11_2_0 (or later)
- We use Run3 CMS + LHC configurations
- Technically, not much else!
  - Premix, MT, MINI, NANO, Russian Roulette and VecGeom for G4 in our sw stack since long
- + The capability for accelerators ... see later
What we propose

- Still 3 WFs
  - GEN+SIM; DIGI with Premix; RECO+MINI. All realistic albeit with some simplifications in PU, beamspot
  - No NANO: it is fast, a negligible fraction of the CMS offline resource utilization. If we ran it, it would have a very low weight.

- Internal weights:
  - Can we set them? Ideally we would like to imbalance towards RECO:
    - We run RECO on MC and DATA, the other WFs are MC only
    - Ideally, we run RECO more than once on each DATA and MC event
  - Is this technically possible? It would mean putting weights as something like (1, 1, 2.x)
What we have

- David prepared a single script which runs out of no input and processes everything
- Decisions to be taken:
  - The script is also able to generate the premix library from scratch. In the previous version, the library was in the container. Which way is preferred?
    - In case we generate it, the processes to do so should have weight 0 in the average
    - In case, they could be generated once and not three times for three benchmark runs
    - The premix file could still be in the container
  - # of threads: as we discussed last time, the best is to use the most probable settings we use in production. It would be 8
  - # of events to process: it just depends on the total time allowed for the test. In the parser we subtract part of the startup time, but not all, so either we make it smarter or we use at least a few 100s events.

/afs/cern.ch/user/t/tboccali/public/210126_bench_hepscore/example_cmssw_bench_TomMod.sh
Naive measurement on a random lxplus

- **100 events processed take ~40 minutes** (using 8 cores of lxplus)
  - Out of which, ~20 minutes to generate from scratch the Premixed PU input for the DIGI step (it will have 0 weight in the average)
  - If we put the Premixed PU in the container, it needs **290 MB** (scales with the number of events) but reduces by ½ the total processing time

- All these numbers scale linearly with the number of events
A little treat ...

- We tested the same exact commands also on Power8/9 and AARCH64
- They work w/o problems; the script runs as-is
- We never explicitly discussed preparing a cross platform benchmark already now, but if we want to try, all is ready (including the GPU part, see later)

- Power8 (ibm-minsky2.cern.ch): some ~ 55 min
- AARCH64 (techlab-arm64-thunderx-02): some ~ 140 min
GPUs

- The CPU part is basically good to go (needs a final handshaking with Domenico et al)
- What about the GPU part?
  - As discussed, we do not assume to have sizeable offline resources with GPUs in Run3
  - We could still run some tests, mostly for technical reasons; if we do so, they should have weight
    \[\text{weight} = 0\] in the average
- What do we have?
  - Already contacts between A. Bocci (CMS-HLT) and D. Giordano for the standard test we use to
    measure the offload to GPU
- It needs a machine with a proper CUDA driver
- It *can* use CUDA libraries from CMSSW in case the local libraries are too old (but it still needs a recent CUDA driver)
- What we propose is to run twice the same HLT Workflow
  ○ First time forcing using CPUs
  ○ Second time allowing for GPUs if discovered - if not it will result in an identical result wrt the first run
  ○ (how to run MP to fill the machine? It needs a MPS daemon to share the GPU between processes...)

If the benchmarking FW has a way to assess the presence of a properly configured system, the second could be skipped
Conclusions

- Thanks to the work by David, a proposal very similar wrt the current one is available for CPUs
  - There are some options which depend on the way we want to run it
- For GPUs, we have a candidate, but we need to investigate better the environment we will run into
  - Libraries, drivers, multi processing daemon ...