# Thinking out loud for HEP-CCE++

Matti Kortelainen

HEP-CCE All Hands Meeting

20-22 April 2022

# Towards efficient use of compute accelerators

To me there are many open questions on how to use compute accelerators efficiently from general-purpose frameworks. Personal bias towards “lots of short algorithms, medium size data structures”.

- Batching/ganging data from multiple “Events” for compute accelerator processing
  - This has been the number of one suggestion from GPU experts every time I’ve gone through the Patatrack code with them for performance improvements
  - For relatively simple cases the benefits are clear: smaller overheads lead to better performance
  - But, when done within the following setting, benefits are far from clear without trying it out
    - Application with O(1k) framework modules (“algorithms”)
    - Each module potentially has large variance in resource needs between “Events”
    - Only some parts of the module DAG is offloaded
      - But there can be chains of module that process further the existing data in device memory

# Towards efficient use of compute accelerators

- Data structures in conjunction with I/O
  - Minimal requirement for GPU-friendly data structure:
    - Enables coalesced memory access
    - Minimizes the number of API calls to transfer the data between host and device
  - HEP data is often structured, with different fields having different types (`int`, `float`, …)
  - General pattern: (Array-of-)Structures-of-Arrays (but really depends on the algorithm)
  - Several approaches in the wild, in the long term would be great to serialize with ROOT
  - ROOT is internally columnar across “Events”, can we benefit from that towards batching?
    - A priori no guarantees the data “batched” in I/O would be beneficial to “batch” in processing
  - Differentiate / collaborate with other approaches?
    - I’ve understood LLAMA has been tried out with ROOT
    - CERN CMS group is developing a marco-based SoA declaration machinery, targeting also I/O with ROOT

# (Distant?) future architectures

- Should we worry about FPGAs or other architectures that are not about scalar/vector processing (like CPUs and GPUs are)?
  - Or will these be e.g. so ML specific that working on non-ML algorithms for them would not be worth of the effort?

- Should we worry about “heterogeneously heterogeneous” HPC platforms?
  - Current/next machines look more like “homogeneously heterogeneous”, i.e. each node has the same hardware
  - Having a separate CPU-only and CPU+GPU partitions (as in Perlmutter soon) is a small step towards different kinds of nodes