Leveraging New Computational Architectures and Strategies for Event Generation and Simulation

Charles Leggett

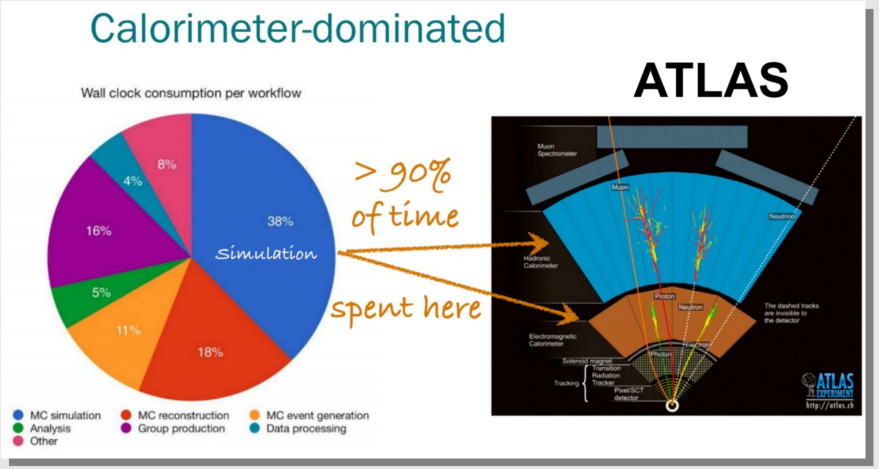
Snowmass Community Planning Meeting
Oct 7 2020

Event Generation

- For LHC, currently need $O(10^{10})$ events per year of data
  - Limited size of generated event pool can limit physics
- CPU requirements for Event Generation are currently about 5% (CMS) 12% (ATLAS) of CPU budget
  - Projected to need more cycles for HL-LHC: beyond NLO, higher jet multiplicities, etc
- In principle generator code should map well onto GPU’s massively parallel architecture
  - Compute same function over many phase space points
- Madgraph (re)port is underway using CUDA
  - gVegas, gBases
- Other generators, such as Sherpa could map well onto parallel computations
  - With a non trivial amount of work, but with great benefits
- Random numbers
  - There are several high quality RNGs that work on GPUs and other parallel architectures

Simulation

- Geant4 is a major consumer of CPU cycles
  - How can we run G4 on GPUs?
    - Lessons learned from GeantV: Not easily
    - Thread divergence due to conditional branching
  - Can we get by with non full-sim?
    - Parametrized "fast" simulations
      - » Prototype of ATLAS parametrized simulation port to GPU shows up to 60x speed up



- Work sizes are often small, memory bound – short kernels and inefficient GPU use
  - Architectural changes to group work between events
- Simplified geometries and data structures that are GPU friendly
  - eg, mesh based approaches,VegGeom
  - Can we learn from Pixar's Universal Scene Description?
  - Major challenge to rewrite massive quantities of existing code / detector descriptions and validate
- ML techniques and GANs for Simulation: excellent target for GPUs and TPUs

Validation, Numerical Precision, Portability

- Validation
  - Different code paths / Algorithms on CPU vs GPU?
  - Will be even more challenging than currently with MT, various architectures & math libs, etc
  - Even if using exact same random numbers, can still get different results due to variations in orders of operations on GPU
- Numerical precision
  - Much less silicon (if any) for DP on GPUs than SP or HP
    - Different GPU models have varied support of DP in hardware
    - “Consumer” cards are moving away from DP support
  - Can we use reduced / mixed → much faster
    - BFLOAT16, etc
- Code portability: many different architectures, software/hardware changing rapidly
  - NVidia, AMD, Intel → CUDA, hip, SYCL
  - Fugaku, FPGA, ASIC, TPU
  - Investigate Kokkos, Alpaka, OpenACC, C++ standards, etc.