FastCaloSimV2 on GPU Status

Charles Leggett

ATLAS Software and Computing Week #64
December 5 2019

HPC for LHC: Accelerating ATLAS Fast Calorimeter Simulations on GPUs


Zhihua Dong
BNL


Tadej Novak
Jozef Stefan Institute


Kwangmin Yu
BNL


Ahmed Hasib
U. of Edinburgh


Charles Leggett
LBNL


Doug Benjamin
ANL


Heather Gray
LBNL

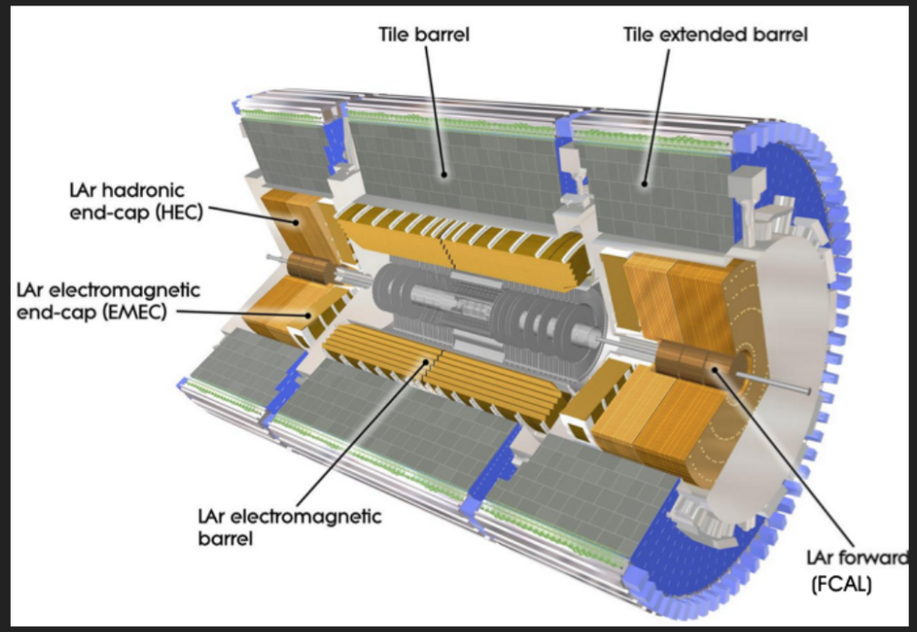

Meifeng Lin
BNL

Work supported by DOE HEP via HEP Center for Computational Excellence (CCE)

BROOKHAVEN NATIONAL LABORATORY

ATLAS FAST CALORIMETER SIMULATION

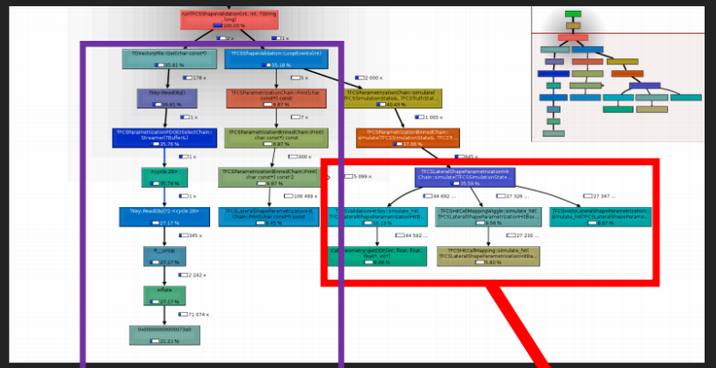
- Calorimeter simulation measures the energy depositions of O(1000) particles after each collision.
- Full detailed simulation uses Geant4, which is very slow
- Fast calorimeter simulation uses parametrization of the calorimeter: less accurate but much faster than Geant4 [T. Yamanaka (ATLAS) 2011]
- FastCaloSim (FCS): a relatively self-contained code for fast ATLAS calorimeter simulation
  - Good candidate for proof-of-concept GPU/ portability studies



Credit: ATLAS

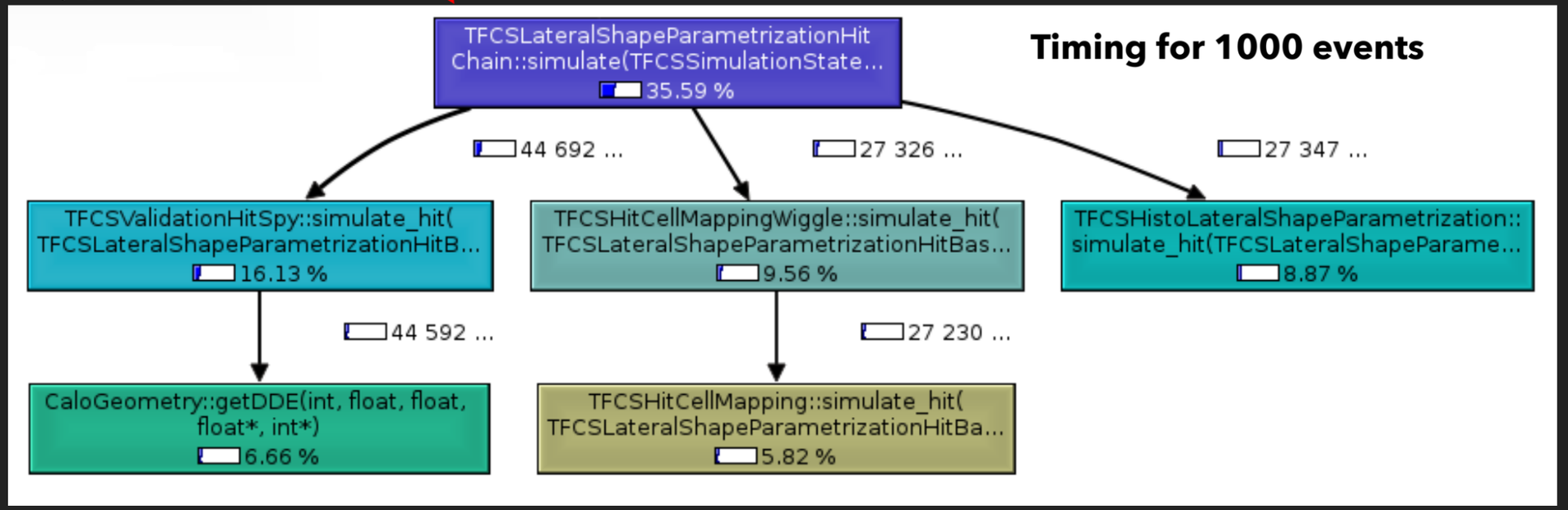
BROOKHAVEN NATIONAL LABORATORY

PERFORMANCE PROFILING



I/O routines

- **TFCSLateralShapeParametrizationHitChain::simulate()** is the **most significant** routine except I/O (~30%).
- **TFCSLateralShapeParametrizationHitChain::simulate()** The running time **scales with the number of events.**
- **TFCSLateralShapeParametrizationHitChain::simulate()** is our target to parallelize/port to GPUs.

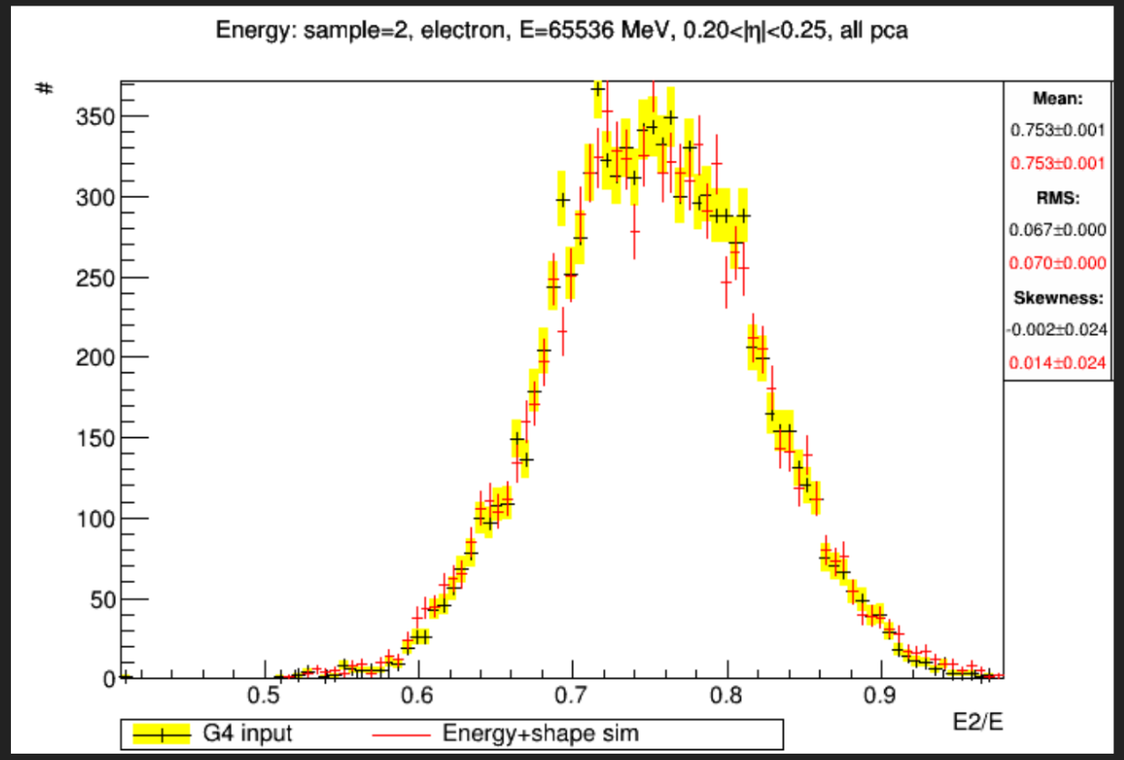


GPU PORTING

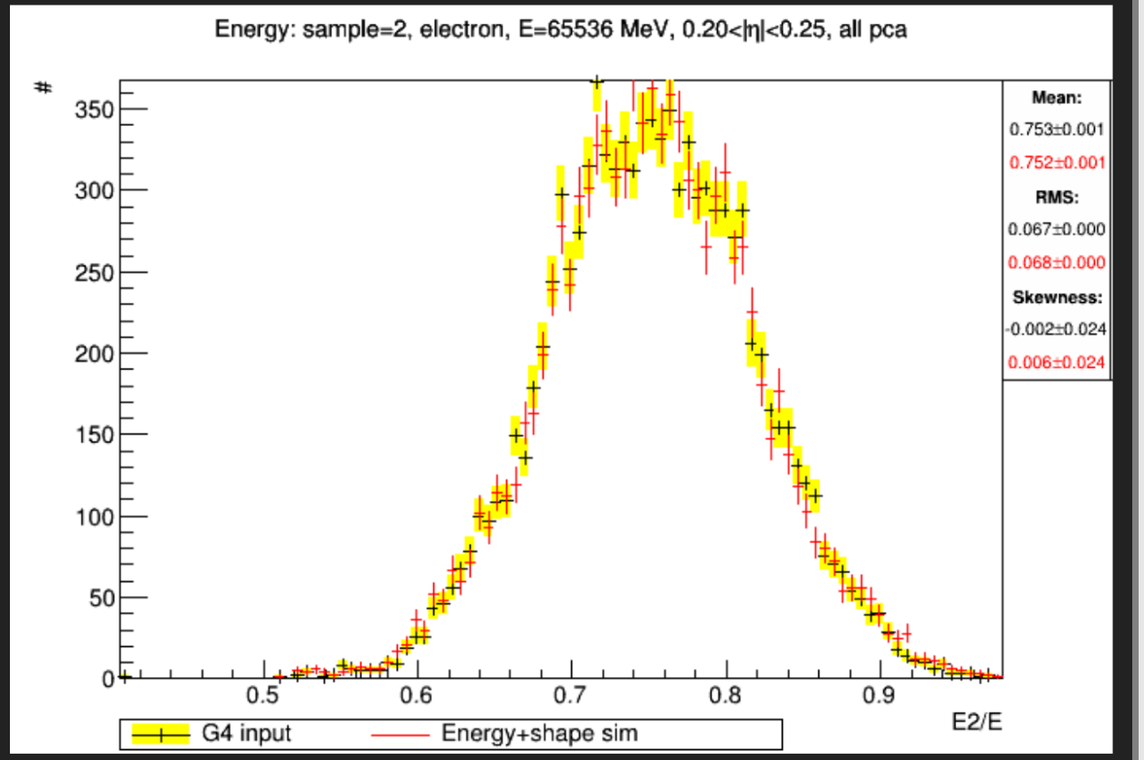
- Initial strategy: CUDA
  - to identify feasibility and challenges with GPU porting
- Data structure modification from CPU to GPU:
  - Implemented new GPU CaloGeomory structure and supporting Classes
  - Simpler, no ROOT Dependence
  - CaloGeometry data can be loaded once and be reused: ~25MB
- Multi-stage CUDA kernels to generate histograms
  - Blockwise atomic update with shared memory
  - Followed by reduction across all blocks
- To get # of hits in the calo cells
  - only ~200 cells get hit out of 20,000 cells - trial run to narrow down the hit cells
  - Reduces memory requirement, and load imbalance

RESULTS VALIDATION

CPU



GPU



- Use of different random numbers on CPU and GPU: results not bit-wise reproducible
- Statistically equivalent
- Looking at ways for better reproducibility

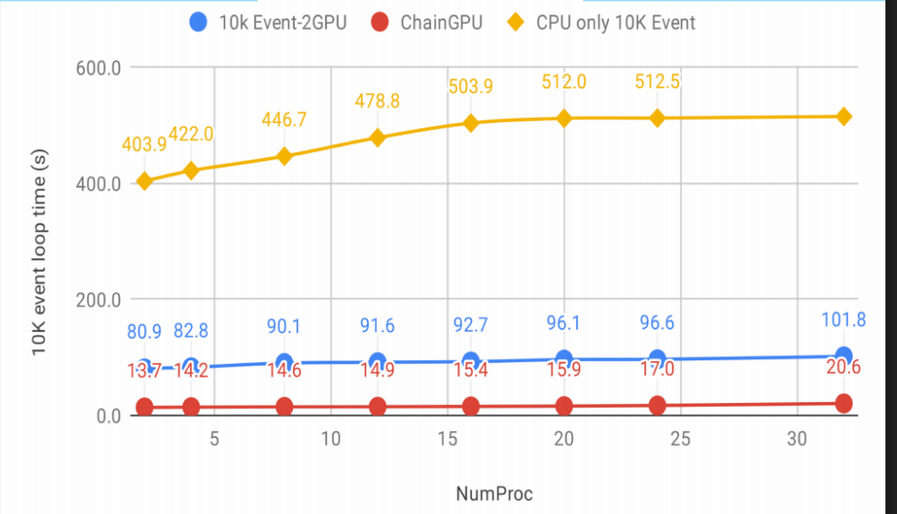
BROOKHAVEN NATIONAL LABORATORY

CPU VS GPU PERFORMANCE COMPARISON

- Validation against GEANT4 most time consuming (~50K hits)
- CPU: “embarrassingly parallel” - different processes simulate different events
- GPU: Use CUDA-MPS to share 2 P100 GPUs on BNL Institutional Cluster*
- ~5X gain with 50K hits compared to CPU only runs (32 parallel processes).
- Actual production runs have fewer hits - less compute
  - Less performance gain: 2-3X vs. CPU

* CPU: Intel Xeon “Broadwell” 32 cores per node
* GPU: 2 NVIDIA P100 per node

Validation — Event loop time (s) — ~50,000 hits



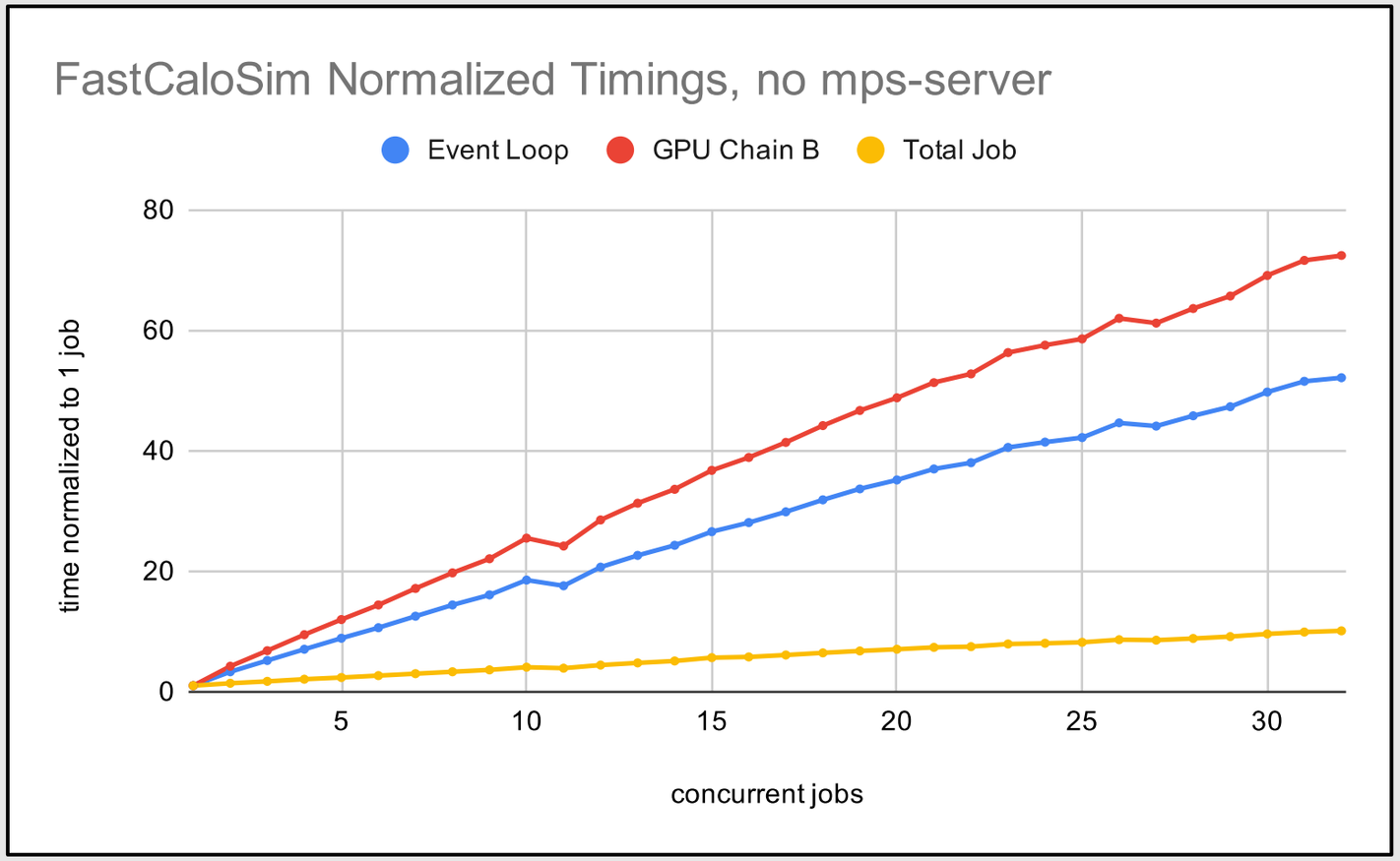
Production Run — ~5,000 hits

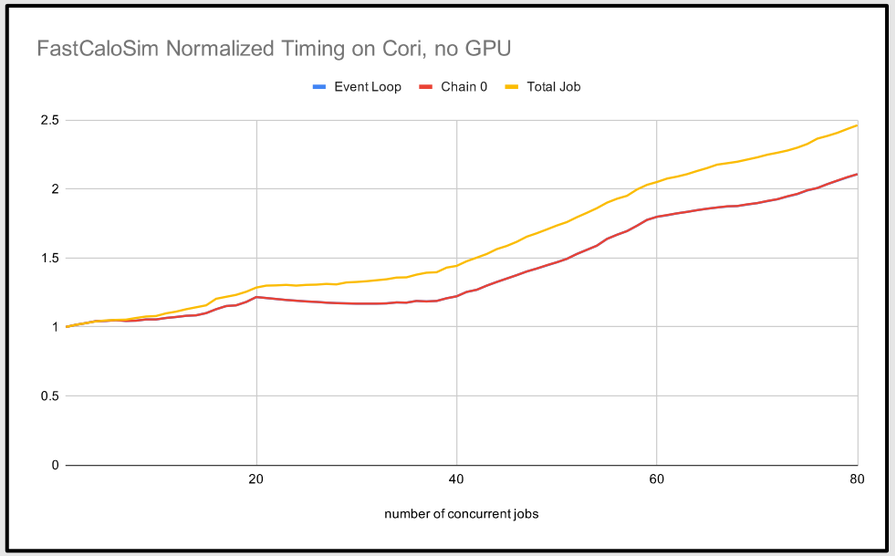
#MPI Processes	Particle	Energy	Min Eta	CPU (s) /10K event / process	GPU (s) / 10K event / process
1	Electron	65536	2.2	**18.8**	**6.0**
32	Electron	65536	2.2	**24.0**	**7.1**

BROOKHAVEN NATIONAL LABORATORY

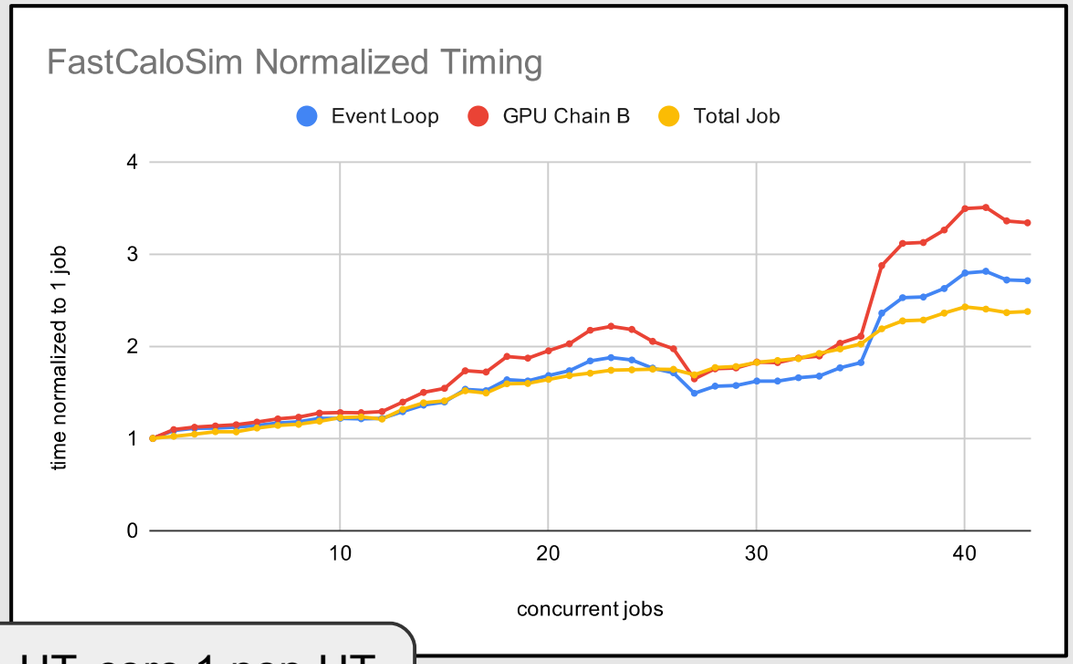
Benchmarks on Cori GPU Nodes

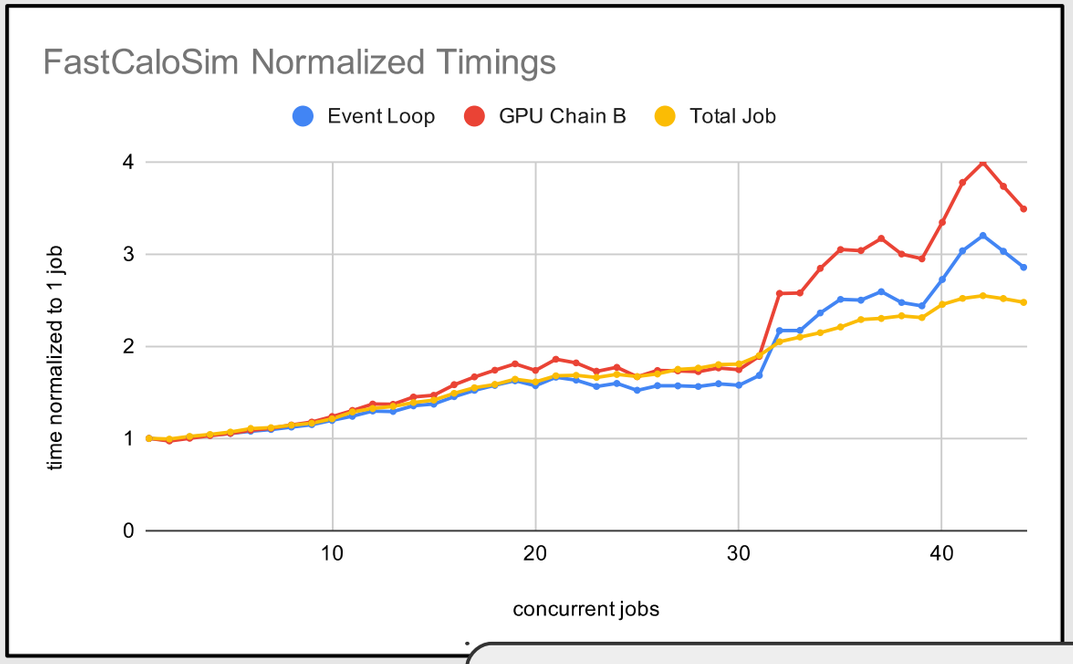
- Cori GPU node: 2x40 core Haswell + 8x NVidia V100 w/ 16GB
- launch multiple jobs concurrently





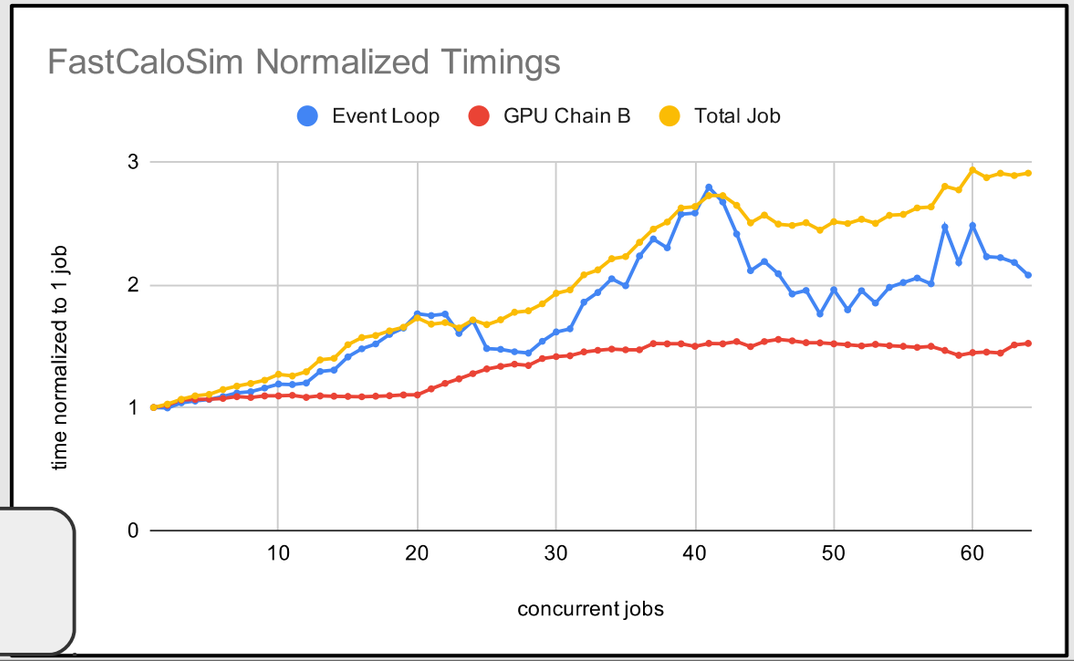
Benchmarks on Cori w/ nvidia-cuda-mps-server





core 0 non-HT, core 1 non-HT
core 0 HT, core 1 HT

alternate cores, use HT before non-HT



- taskset -c to set core affinity
- add concurrent jobs until GPU runs out of memory

core 0 non-HT, core 0 HT
core 1 non-HT, core 1 HT

- cuda kernels are too small and do insufficient work to keep GPU occupied, even with 60 concurrent jobs

NEXT STEPS

- GPU utilization is still low, especially in the production runs (fewer numbers of hits)
  - Looking into parallelizing code on
    - Particles level
    - Events level
- Portable solutions?
  - Don't want to rewrite code for every new architecture
  - HL-LHC will be in post-exascale era
  - Need something that can stand the test of time

```
Loop on events (O(10,000) or more )  {
…
    loop on  particles (~1000 / event) {
        …
     loop on calorimeter layers ( <20) {
            loop on hits ( 1-6k )  {
                …
            }
        …
        }
    …
    }
…
}
```

Current GPU acceleration

See Vince's talk next

Also: CCE proposal intends to look into a Kokkos port