# More on Lessons learned and Follow-up

Philippe Canal, Daniel Elvira for the FNAL GeantV team

# Some lessons learned

- Main factors in the speedup: modern code developed from ground up
  - Better cache use
  - Tighter code (e.g., less indirections and branching)
- Vectorization's impact (much) smaller than hoped for
  - Effectively small fraction of the GeantV code has been vectorized or is run in vector mode. (Amdahl's law limits overall speedup.)
- Basketization is challenging
  - Either extra memory copy (using collection of tracks)
  - Or lower memory access coherency (using collection of pointers)
- "Localized" benchmark performance challenging to extend to full use case
- Migration of backward incompatible solutions [not necessarily GeantV] to a real experiment is manageable within the timescale of HL-LHC
  - Example: Successful integration tests with CMSSW, including physics validation and computing performance evaluation – GeantV events reconstituted, hits ready for digitization!
  - Heterogeneous computing solutions possible – CMSSW external work feature

# Options and Potential Strategies

1. "Physics Preserving" options
   a) Code modernization using lessons learned from GeantV and other R&D lines of work
   b) Adaptation to HPC and efficient use of accelerators
2. Fast Simulation options
   a) Improvements to parametrized simulation or extend fast simulation use cases – Example: CMS' current parametrized simulation ( <<1 sec/event and reproduces most physics observables within 10%)
   b) R&D on Machine Learning (ML/AI) to achieve acceptable speed/fidelity balance

External conditions from experiments and computing technology

- Experiments need high precision for e.g. MET, boosted objects, jet sub-structure, particle separation in high-granularity detectors – need full G4 simulation as in run 2
- ML may prove to replace full sim (physics preserving) or be just another fast sim option with limited applications – need large "full simulation" for training anyway (HPC?)
- HPC facilities increasing fraction of HEP computing resources – guidance from funding agencies: "adapt, run on accelerators, save us money"

# One Way Forward

## R&D Questions

- How much further can we push the cache efficiency increase ?
- Given the challenges of basketization, how can we leverage the 1000s of cores on GPUs ?

## Geant Exascale Pilot Project

- Goals
  - Investigate how to best use GPUs for full HEP simulation
  - Explore memory access, computation ordering, and CPU/GPU communication patterns
  - Avoid over-simplification
- Collaboration
  - Fermilab, Lawrence Berkeley Lab, Oak Ridge National Lab, participants from US-CMS, US-ATLAS
- Strategies
  - Reuse or leverage existing packages, not bound by backward compatibility.
  - Focus on **NVidia** compiler at first (later look at Kokkos and others)
  - **Partial Static Polymorphism**: allow upload/download of data to device without transformation
  - **Separation Of State and Access and Functional Approach**: allow significant data memory layout change without code change

# Backups

# Exascale: GPU for the Foreseeable Future

- Perlmutter (NERSC-9) 2020: AMD and Nvidia GPUs
  - GPU nodes will have a GPU to CPU ratio of 4:1
  - 256GB memory per node or greater
  - >4000 node CPU partition, approximately same capability as full Cori system today
- Frontier (ORNL) 2021 and El Capitan (LLNL) 2023:
  - one AMD EPYCTM processor to four AMD Instinct graphics cards
- A21 (ANL) 2021:
  - Intel Configurable Spatial Accelerator
    - Dataflow graph engine
    - Maps compiler execution graph (IR) to hardware fabric
    - Elements of modern switches, FPGAs, KNL
  - Intel GPU, Intel ONE API


# DOE, Exascale, GPU, and HEP

- DOE push for use of Exascale Computing (and thus GPU) not limited to ECP.
  - ECP is through the DOE Office of Science's ***Advanced Scientific Computing Research***
- DOE Office of Science's ***High Energy Physics*** program is also investing towards use of GPU
- Explicit requirements for experiments in upcoming years

  Use (mostly/only) Exascale machines
  ***and***
  Use of Exascale machines allowed ***only*** if making efficient use of accelerators

- Started first (AFAIK) official collaboration between HEP and ECP
  - Investigating general purpose detector simulation on GPU
  - Fermilab, Lawrence Berkeley Lab, Oak Ridge National Lab
  - Includes researchers associated with US-ATLAS and US-CMS

# R&D Questions

- How much further can we push the cache efficiency increase ?
- Given the challenges of basketization, how can we leverage the 1000s of cores on GPUs ?

# Goals

- Investigate how to best use GPUs for full HEP simulation
  - Estimate higher limit of speed-ups.
- Explore memory access, computation ordering, and CPU/GPU communication patterns
  - Require flexibility on data structure and code structure
- Avoid over-simplification
  - Look at the whole Geant simulation chain not just a few components
- Proxy for generic Monte-Carlo transport simulation framework that executes on either the CPU or GPU or combination of both
  - E.g. use GPU where speed-up has been demonstrated, avoid GPU where speed-up is highly unlikely (e.g. physics with significant branching; particles with short-lifetimes and/or a relatively small average number of interactions)
  - Cost of data transfers will be a key metric

# Strategies

- Avoid constraints
  - Backward compatibility
  - Tracking change ‘upstream’
- Reuse or leverage existing packages
  - Newer components like VecGeom, latest magnetic field and physics implementation.
- Focus on NVidia compiler at first
  - Will need to understand AMD and Intel programming direction
  - Likely might switch to a library like Kokkos
- Partial Static Polymorphism
  - Allow upload/download of data to device without transformation
- Separation Of State and Access
  - Allow significant data memory layout change without code change