Geant4 and GPU/vectorization

Philippe Canal (FNAL)
Soon Jun (FNAL)

Studying Code Efficiency

- Studying cmsExp
  - Not evaluating the intrinsic quality of the algorithms; focusing instead on whether memory access (and lack of vectorization) are a bottle neck.
- Tools used: cachegrind, CodeAnalyst, igProf
- Observations
  - Cache misses spread pretty much throughout code.
  - A few hotter spots
    - G4PropagatorInField::ComputedStep
      8% of instructions but 17% of cache miss and 36% of real time (portion of G4RunManager::DoEventLoop, inclusive of all sub calls)
      - In contrast its caller G4Transportation::AlongStepGetPhysicalInteractionLength uses 19% / 23% / 43%

Transportation and GPU

- Goals:
  - Learn more about GPU programming and estimate how much it could help speeding up simulation.
- Focus:
  - Transportation (and magnetic field effect)
- Extracting realistic sample of trajectories:
  - Regular run recording every primary and secondary and their steps length.
  - Use this as input of simulation with a physics list composed of only transportation (and magnetic field) + a mechanism to kill the particles after they were propagated as long as in the original case.
  - Currently working out the last kinks.

Next Steps

- Start prototyping on GPU.
  - With or without (existing) CPU framework?
- Decide whether to use any of:
  - Existing Geant4 code by extracting and ‘vectorize’ some portions.
  - Otto Seiskari’s Prototype of G4 navigation on GPU.
  - SFT’s Geant prototype
  - Existing GPU implementation(s) of similar tasks.
    - Any ideas, pointers?
- Will use the *trajectories samples* from cmsExp (and possibly SimplifiedCalo) to test and contrast the resulting code.