Data locality in G4 detector simulation

Philippe Canal

Design Directions

- *Replace the looping mechanism from handling one single element at a time to handling multiple elements (vectors)*
  - *Reduce the number of decisions and thus the number of incorrect branch predictions*
  - *Reduce the number of overall functions calls*
  - *Reduce the number of calculations*
    - *For example if several tracks are in the same volume, lookup/calculate/use parametrization only once*
  - *Improve memory locality for example by having collections of light weight objects*

Track/Tracklets Bundles

- *Gather tracks/particles together to minimize run-time decisions*
- *Explore which set of dimensions is best*
  - *Particle type, Energy range, Location, etc.*
- *Explore when to move the bundles from core to core and when to bring external data to the bundles*
  - *For example a set of volumes might be pegged to a core/GPU*
- *Split objects in subsets of datum that are used together*
  - *Increase data locality, minimize data transfer (GPU)*
  - *One possible example: the 'location' of all the track in a bundle could be in a vector<location>*

Data Locality Effect in G4

- *The idea is to extract actual data flow from G4*
- *Then emulate the type of reorganization that the 'Detector simulation' demonstrator might push for.*
- *And Measure the actual effect on the use and leveraging of the cpu caches.*
- *Steps:*
  - *Extract data flows from simplifiedCalo and simplified 'fullCms' examples.*
  - *Emulate the current flow.*
  - *Emulate the new data organization.*
  - *Compare the results using callgrind, AMD's CodeAnalyst and/or Intel's VTune and (of course) real time.*
- *Results will feed into the 'Detector simulation' demonstrator*