Simulation And Many Cores

Geant4 Collaboration Meeting
September 22nd, 2011
Philippe Canal, FNAL

Lessons Learned

- *Retrofitting thread safety is expensive*
  - *In development time*
  - *In run-time CPU*
  - *In user development time*
    - *Any user callback needs to also be made thread safe*
- *Most memory savings can also be achieved via fork-and-copy-on-write technique*

Structural Opportunities

- *Geant4 code often tests repetitively for applicability*
  - *Many calls to IsApplicable, GetParticleCode.*
- *Several cases of repeated calls with slow varying inputs and outputs*
  - *G4hPairProductionModel::ComputeDMicroscopicCrossSection*
  - *G4HadronCrossSections::CalcScatteringCrossSections*

Structural Opportunities

- *Particles/tracks propagated through the same volumes*
- *Many decisions can be precomputed, at least partially:*
    - *which physics processes apply to which set of particles*
    - *for which set of particles should the magnetic field be used*
    - *physics process dependent on the particles' energy or other variable properties*

Goals and Constraints

- *Increase CPU efficiency*
- *Enable use of many cores and GPU*
- *Use the need for potentially significant user changes as an opportunity for larger structural changes*

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*

Advantages

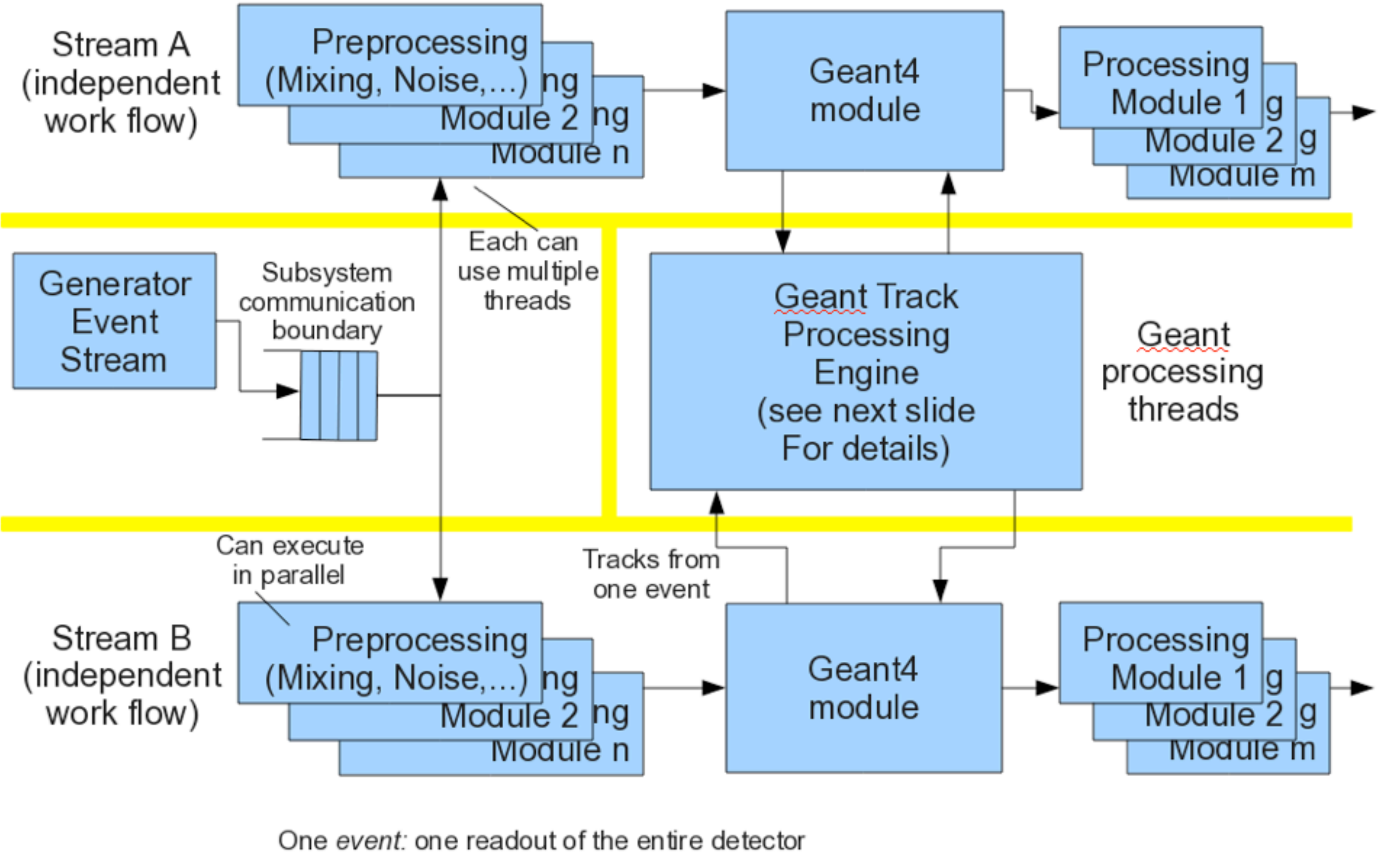
- *Lightweight objects and vectorization is more in line with GPU and other small cache CPU*

- *Necessary rewrite will be an opportunity to be efficiently thread-aware*

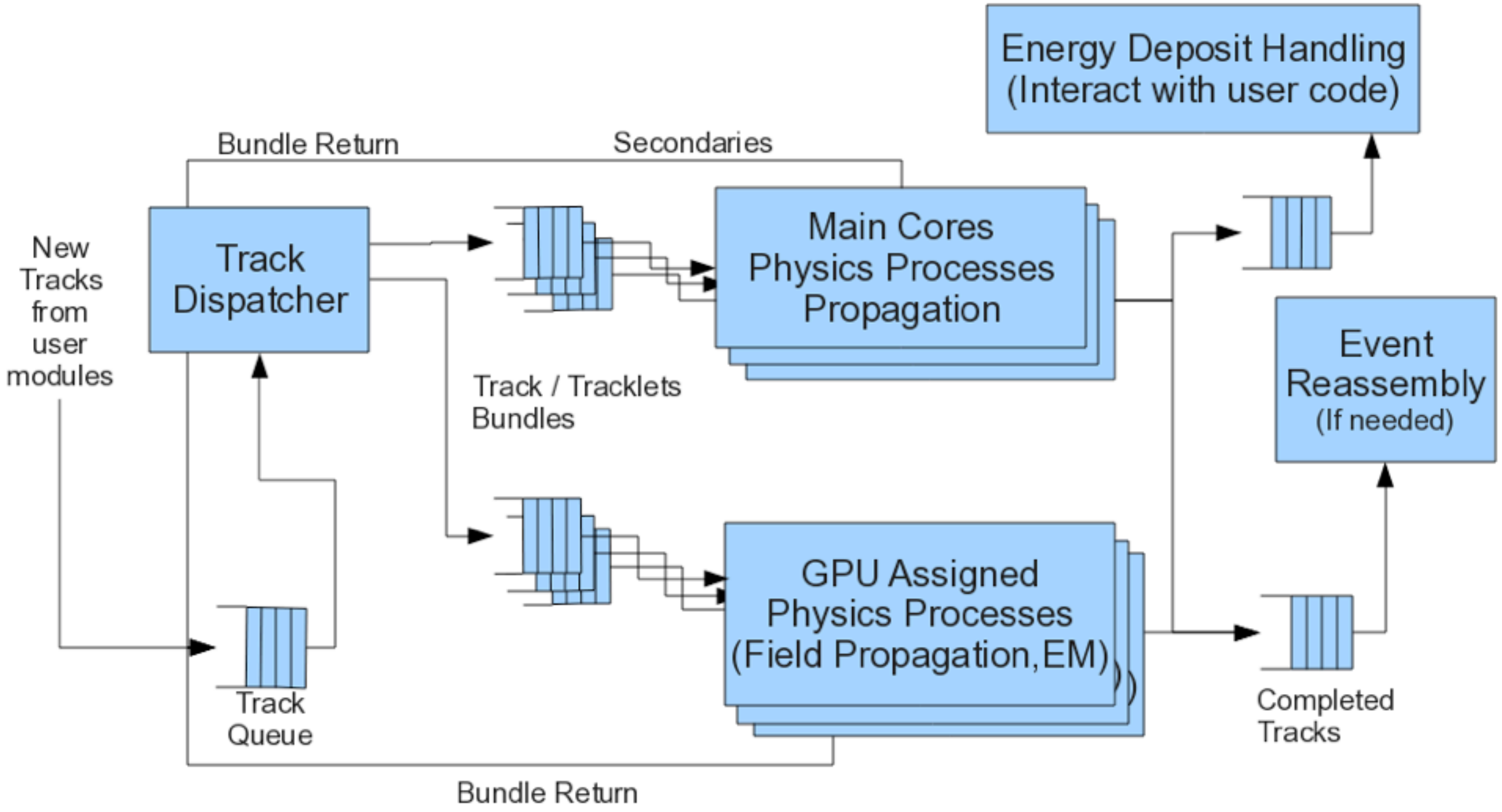
High Level Architecture

- *(Some of the) Future frameworks will be thread capable*
    - *FNAL supplies the* ***art*** *framework to several Intensity Frontier experiments*
    - ***art*** is being updated *to be able to process multiple events in parallel*
    - *Other efforts have similar visions ; may want to have face to face meeting later this year*

⚠ *Requires* ***coordination*** *between the framework and Geant to* ***not over compete*** *for computing resources*


Stream A
(independent
work flow)
Preprocessing
(Mixing, Noise,...)
Module 2
Module n
Geant4
module
Processing
Module 1
Module 2
Module m
Each can
use multiple
threads
Generator
Event
Stream
Subsystem
communication
boundary
Geant Track
Processing
Engine
(see next slide
For details)
Geant
processing
threads
Can execute
in parallel
Tracks from
one event
Stream B
(independent
work flow)
Preprocessing
(Mixing, Noise,...)
Module 2
Module n
Geant4
module
Processing
Module 1
Module 2
Module m
One *event*: one readout of the entire detector

Track Processing



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>*

Track/Tracklets Bundles

- *Each track/tracklet will need to know*
  - *to which event it belongs*
  - *which module instance contains the context for digitizing*
    - *Geant callbacks must be associated with the right module*
- *The reader of the output queue of tracks will need to*
  - *Assemble tracks back into events*
  - *Know when all tracks are complete for an event*
    - *The event then needs to be given back to the right module instance*
- *Need both event and sub-event level parallelism*
  - *See Rene Brun's conclusions.*

Conclusion

- *Leap in performance requires infrastructure changes:*
  - *Vectorization*
  - *Light-weight (array of) objects*
  - *Sub-event and across events parallelism*