Simulation And Many Cores

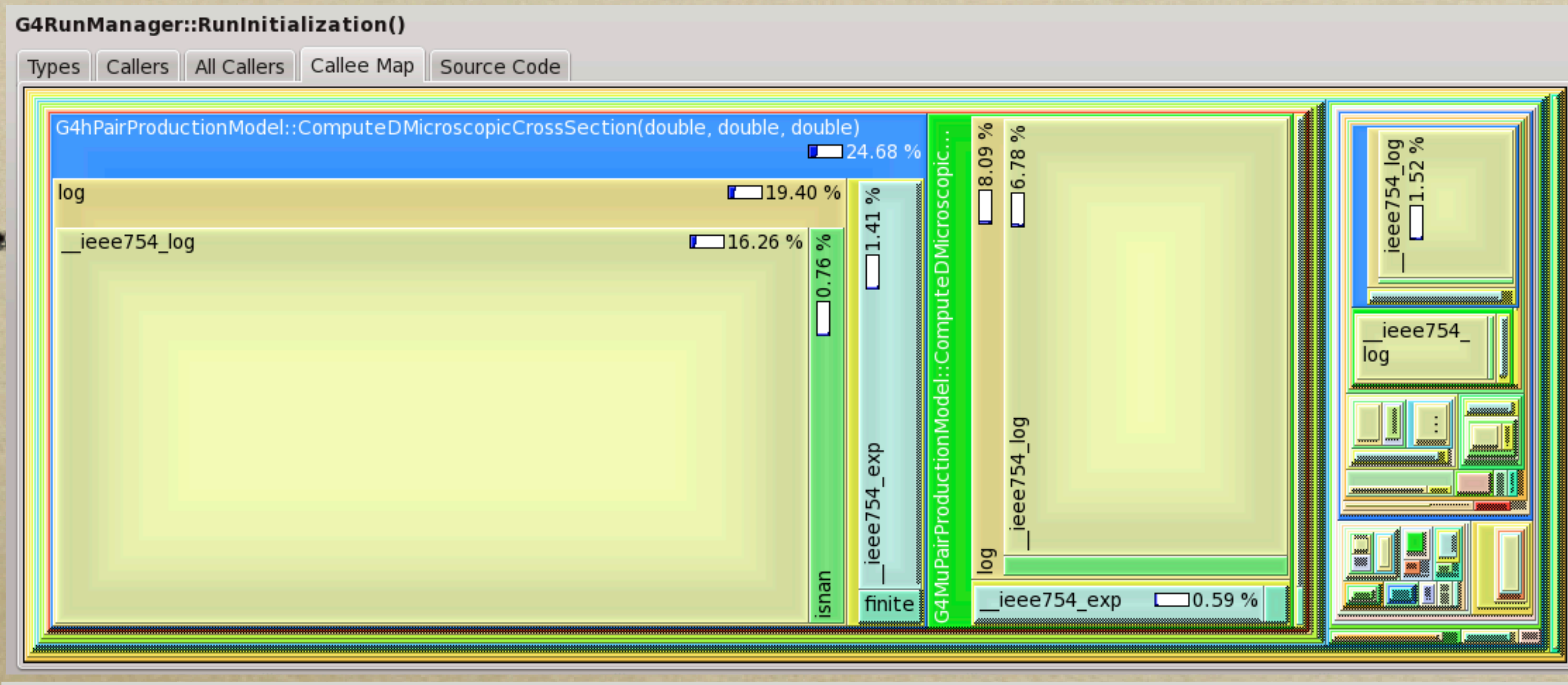
January, 2012
Philippe Canal

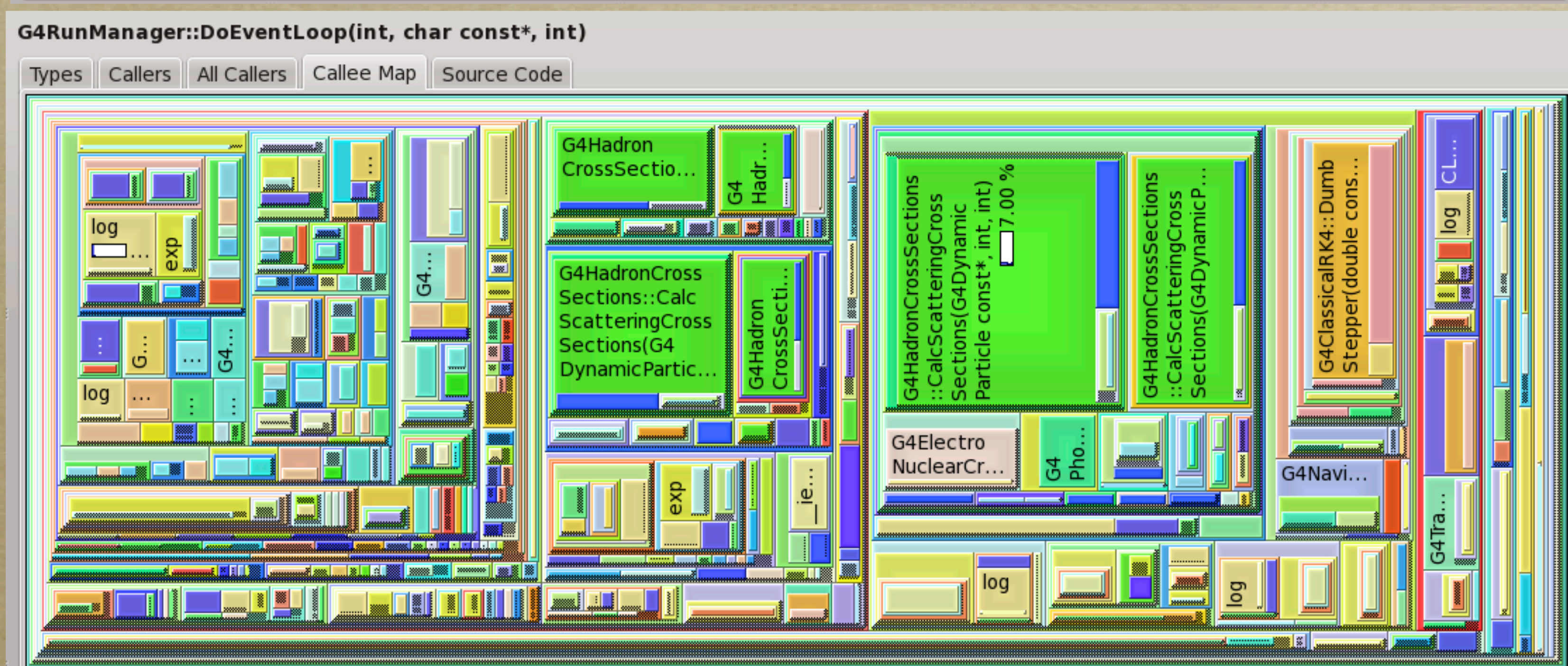
One Performance Study

- *Using **simplifiedCalo** a Geant4 example from Andrea Dotti:*
    - *Test of Shower shapes using selected simplified calorimeter setups*
    - *Using neutron particle gun at 7GeV*

A Few Observations

- *Largest fraction of the time spent in log and exp during initialization.*
- *G4HadronCrossSection::CalcScatteringCrossSection next largest contributor (18% of DoEventLoop)*
- *Time spent is spread amongst large number of functions.*


G4RunManager::RunInitialization()
Types
Callers
All Callers
Callee Map
Source Code
G4hPairProductionModel::ComputeDMicroscopicCrossSection(double, double, double)
24.68 %
log
19.40 %
__ieee754_log
16.26 %
0.76 %
isnan
1.41 %
__ieee754_exp
finite
G4MuPairProductionModel::ComputeDMicroscopic...
8.09 %
log
6.78 %
__ieee754_log
__ieee754_exp
0.59 %
__ieee754_log
1.52 %
__ieee754_
log


G4RunManager::DoEventLoop(int, char const*, int)
Types
Callers
All Callers
Callee Map
Source Code
G4Hadron
CrossSectio...
G4
Hadr...
G4HadronCross
Sections::Calc
ScatteringCross
Sections(G4
DynamicPartic...
G4Hadron
CrossSecti...
G4HadronCrossSections
::CalcScatteringCross
Sections(G4Dynamic
Particle const*, int, int)
7.00 %
G4HadronCrossSections
::CalcScatteringCross
Sections(G4DynamicP...
G4ClassicalRK4::Dumb
Stepper(double cons...
G4Electro
NuclearCr...
G4
Pho...
G4Navi...
G4Tra...
CL...
log
exp
G4...
G...
log
exp
__ie...

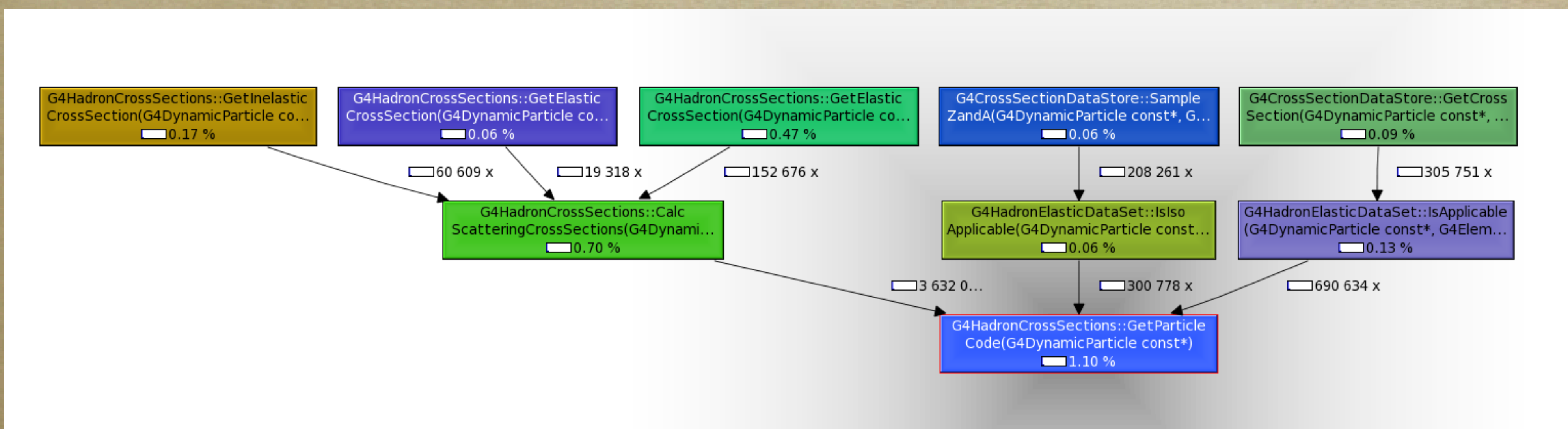
Opportunities

- *1% of time spent in 'IsApplicable' routines*

Incl.	Self	Called	Function	Location
0.19	0.06	690 634	G4HadronElasticDataSet::IsApplicable(G4DynamicP...	libG4processes.so
0.04	0.04	690 565	G4CHIPSElasticXS::IsApplicable(G4DynamicParticle ...	libG4processes.so
0.27	0.09	267 772	G4CrossSectionPairGG::IsApplicable(G4DynamicPart...	libG4processes.so
0.07	0.02	261 061	G4HadronCaptureDataSet::IsApplicable(G4Dynamic...	libG4processes.so
0.07	0.02	258 577	G4HadronInelasticDataSet::IsApplicable(G4Dynamic...	libG4processes.so
0.07	0.02	249 343	G4HadronFissionDataSet::IsApplicable(G4DynamicP...	libG4processes.so
0.18	0.01	237 382	G4ElectroNuclearCrossSection::IsApplicable(G4Dyn...	libG4processes.so
0.02	0.01	169 626	G4PhotoNuclearCrossSection::IsApplicable(G4Dyna...	libG4processes.so
0.00	0.00	839	G4Decay::IsApplicable(G4ParticleDefinition const&)	libG4processes.so
0.00	0.00	446	G4VProcess::IsApplicable(G4ParticleDefinition const&)	libG4error_propagation.
0.00	0.00	348	G4BGGPionElasticXS::IsApplicable(G4DynamicParticl...	libG4processes.so

Opportunities

- *1% of time spent in: G4HadronCrossSection::GetParticleCode*



Opportunities

- *G4hPairProductionModel::*
  ComputeDMicroscopicCrossSection
  - *Takes 55% of the cpu time during G4RunManager::RunInitialization.*
  - *Called 211688 times but with 'only' 112871 distinct inputs.*
  - *Consecutive calls have most often 2 arguments that are the same and the 3rd one incrementing slowly.*

Opportunities

- *G4HadronCrossSections::CalcScatteringCrossSections*
  - *Takes 18% of the event processing CPU time (during G4RunManager::DoEventLoop)*
  - *called 376,200,793 times with only 34,588,580 (9%) distinct input and output values.*
  - *Series of calls where 2 of the three main inputs are the same for 5 or 6 consecutive calls while the 3rd argument varies slowly and the results are numerically very close.*
  - *Same exact series of calls (with the same results) are done many times in close proximity.*

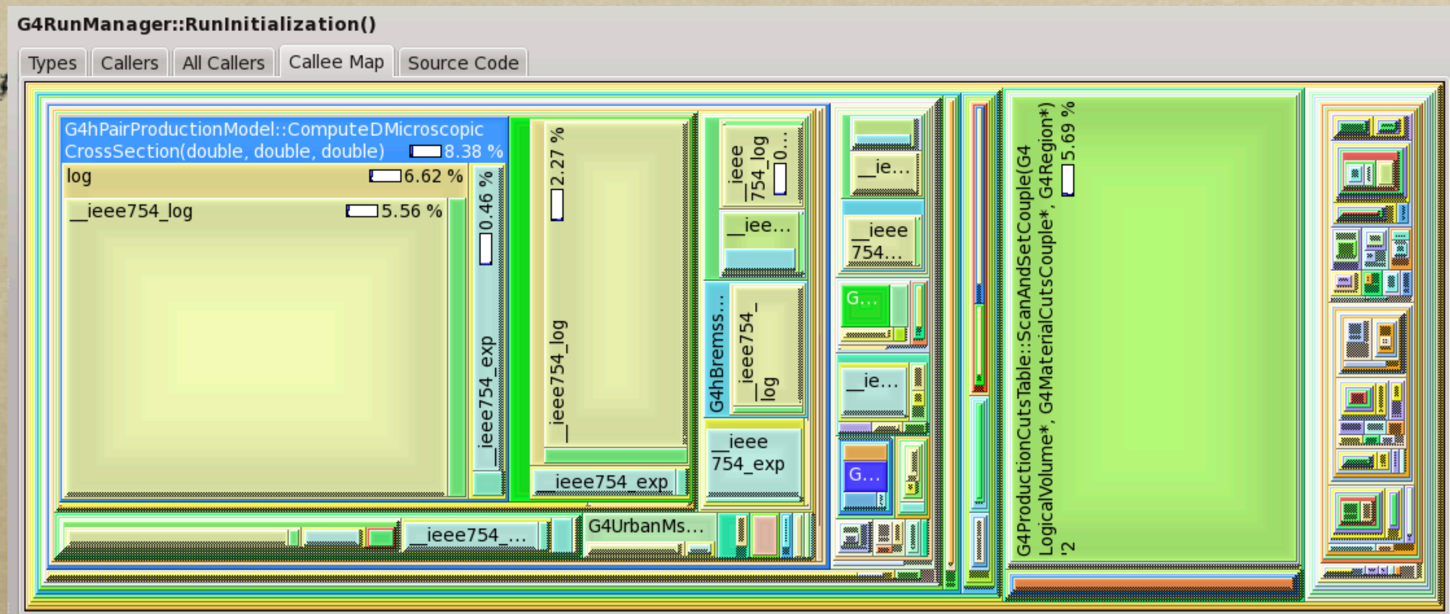
CPU Efficiency

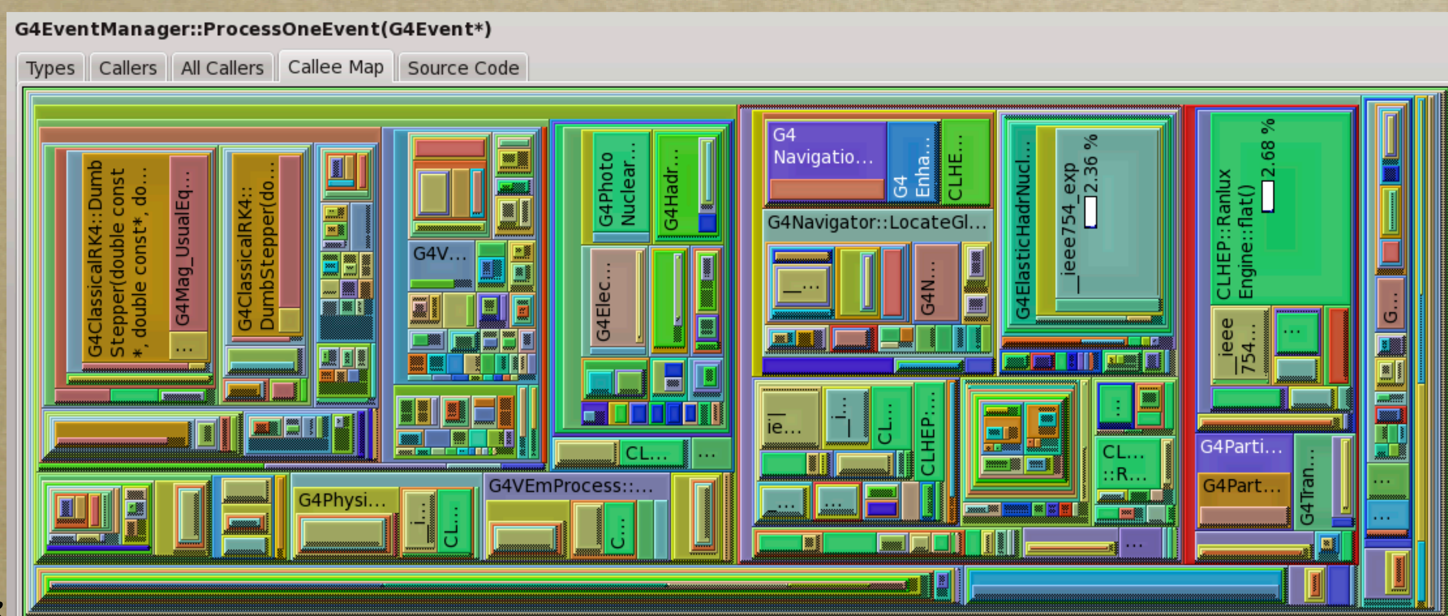
AMD's CodeAnalyst performance Analyzer can calculate the number of instructions per CPU clock cycles in each libraries.

This tables shows the result for the novice example N02

Library	*Inst. Per Cycle*
libm	*0.8*
G4Geometry	*0.71*
CLHEP	*0.72*
G4Processes	*0.56*
G4Tracking	*0.55*
G4Track	*0.52*
G4Globals	*0.65*

ParFullCMS Example





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*

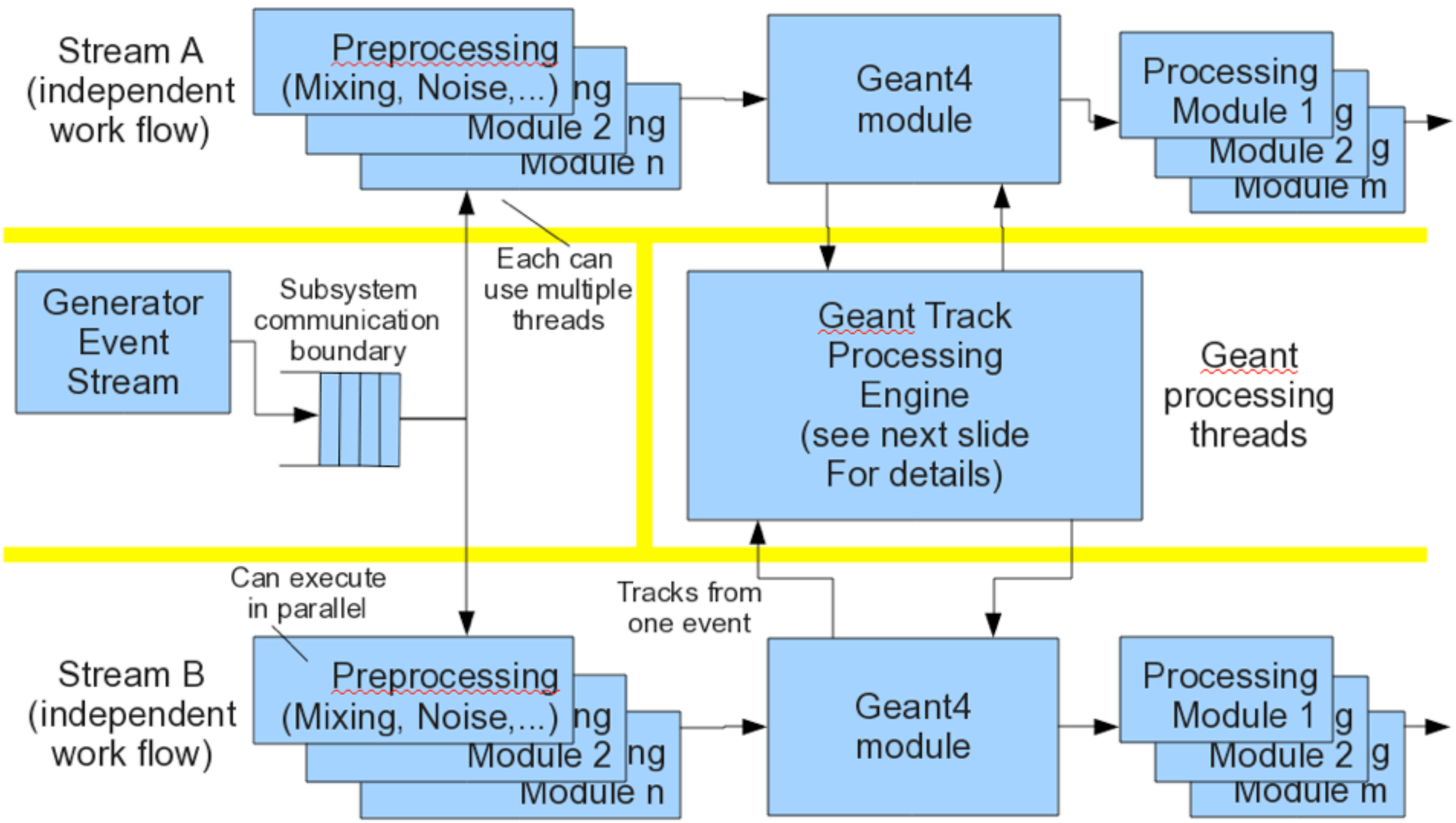
Advantages

- *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

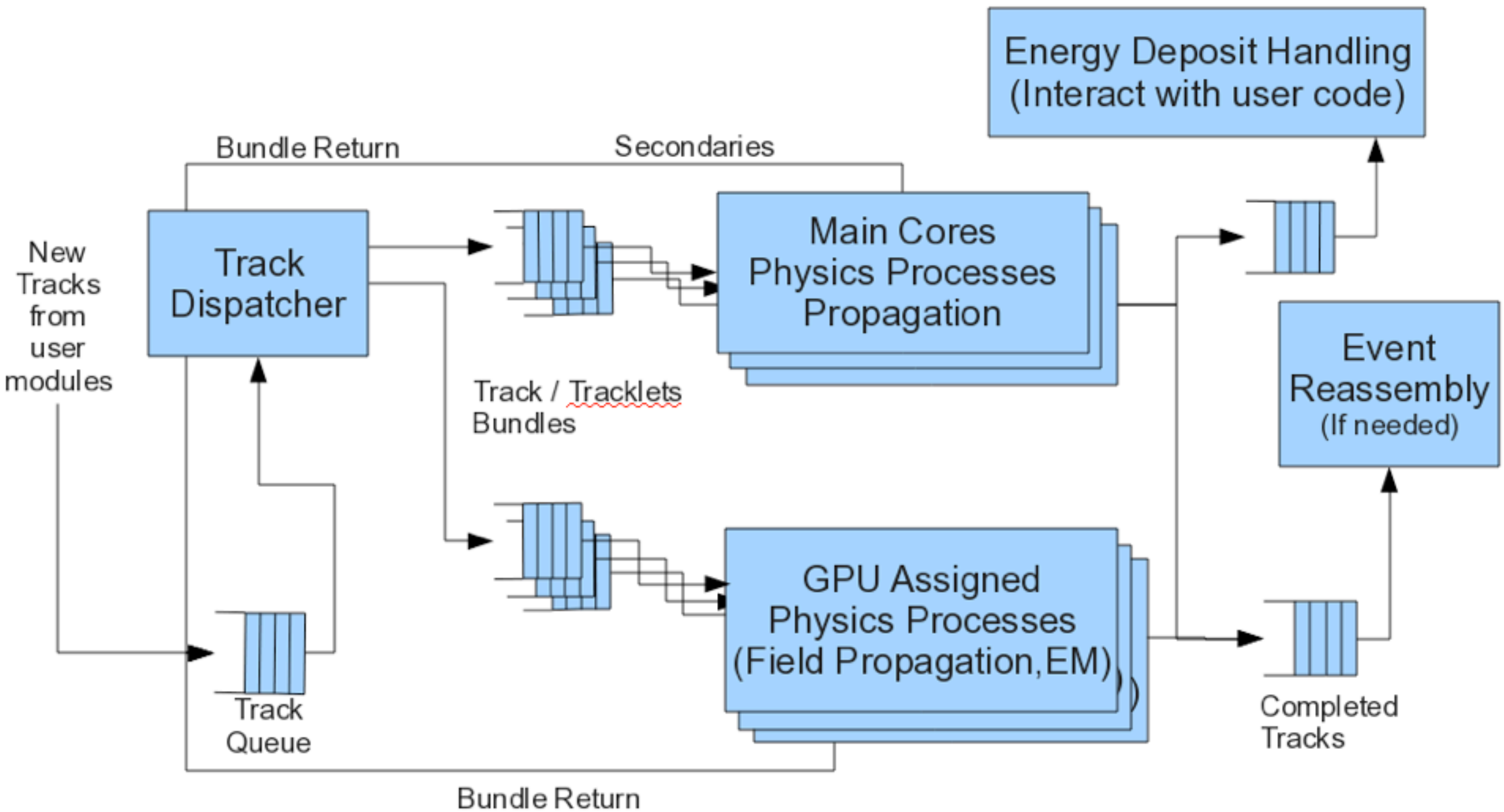
- *(Some of the) Future frameworks will be thread capable*
  - *FNAL supplies the **art** framework to several Intensity Frontier experiments*
  - ***art*** is being updated *to be able to process multiple events in parallel*

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



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*
- *Upcoming tasks*
  - *Concretely evaluate the potential gain brought by improving data locality*
  - *Prototype adapting the Geant4 particle propagation components to use GPGPU and sub-event parallelism.*