Geant 4 Performance

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

Geant 4 Performance

- *A look at Geant4MT*

- *Performance improvement opportunities*

A look at Geant4MT

- *Concept:*
  - *Use automatic tool to discover shared variables.*
  - *Semi-automatic modification of the code base making most shared variables thread local.*
  - *Use thread private malloc library.*

Geant4MT Memory Gains

- *Memory changes on 2 main examples*
  - *Parallel version of Novice02*
    - *no significant memory gain.*
  - *Parallel version of full CMS simulation.*
    - *140Mb of resident memory saving per thread/process (out of the 170Mb used per single thread process).*

Memory Use Measurement

- *Best measure of the memory cost of additional thread or process:*
  - *Resident Size - Shared Memory*
    - *For example with ParFullCMS:*
      Virtual Size: 420Mb,
      Resident Size: 190Mb, Shared Size: 20Mb.
  - *Checking the decrease in free memory (as reported by top) under heavy load confirm that this is the salient number.*

Geant4MT Tests

- *Appears to work on the examples it was build upon.*
- *However is unstable when thread count increases*
  - *On a 4 cores CPU, at 12 threads the processes segfault about half the time.*
  - *Failure rate increases with the load of the machine.*
  - *On the same machine:*
    - *'make -j24' always succeeds.*
    - *Running at the same time 800 regular Geant4 processes always succeeds.*

Geant4MT Performance

- *Lower CPU performance*
  - *The (expected) cost of thread local memory and synchronization.*
  - *Geant4MT is 25% to 35% slower than non Geant4MT processes.*

Initialization was about 5% of total time.

	4.9.4.p01	*Geant4MT*
N02	*28s*	*39s*
CMS	*2188s*	*3460s*

Geant4MT

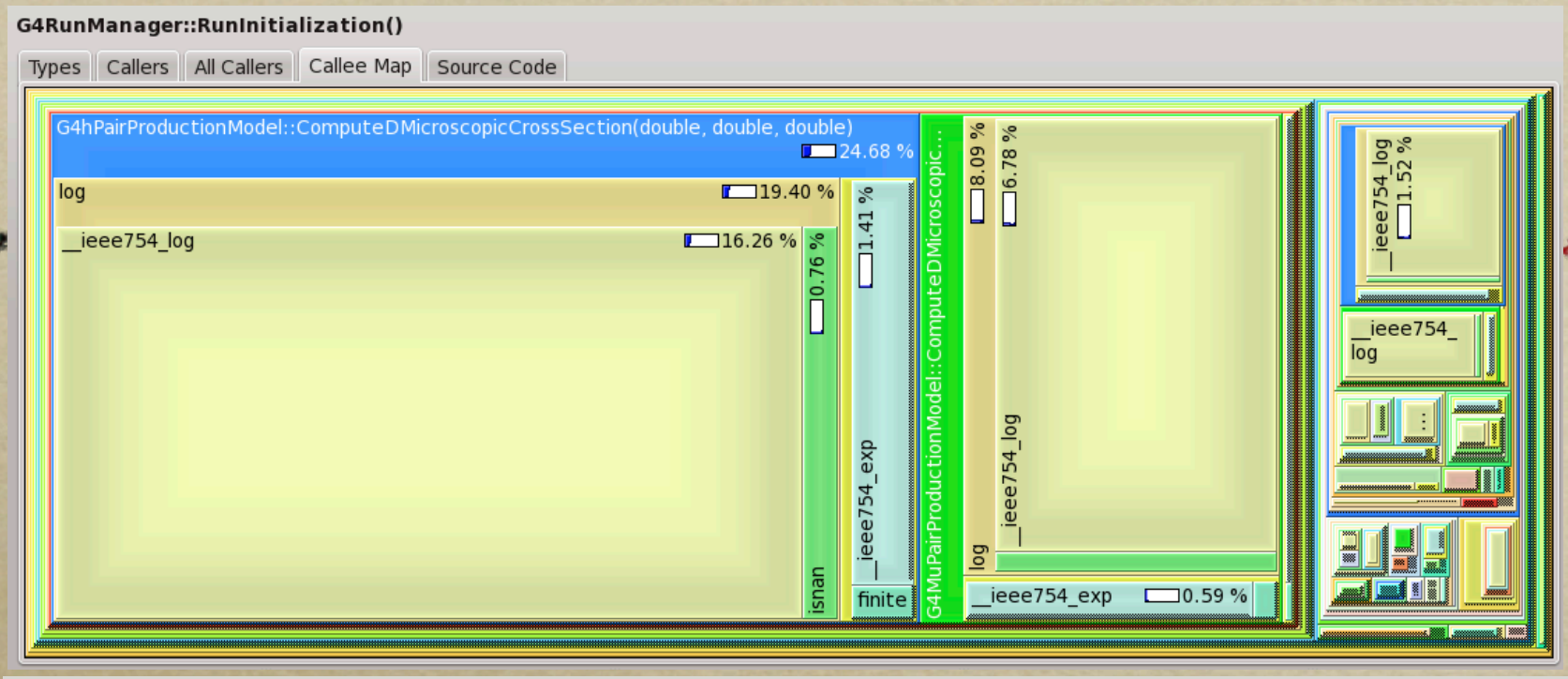
- *Can save some memory.*
- *At a CPU cost (25% to 35%)*
  - *And at least for now the Geant4MT make the thread cost unconditional (i.e. cost even if there is only one thread).*
- *Require non trivial changes in user code.*

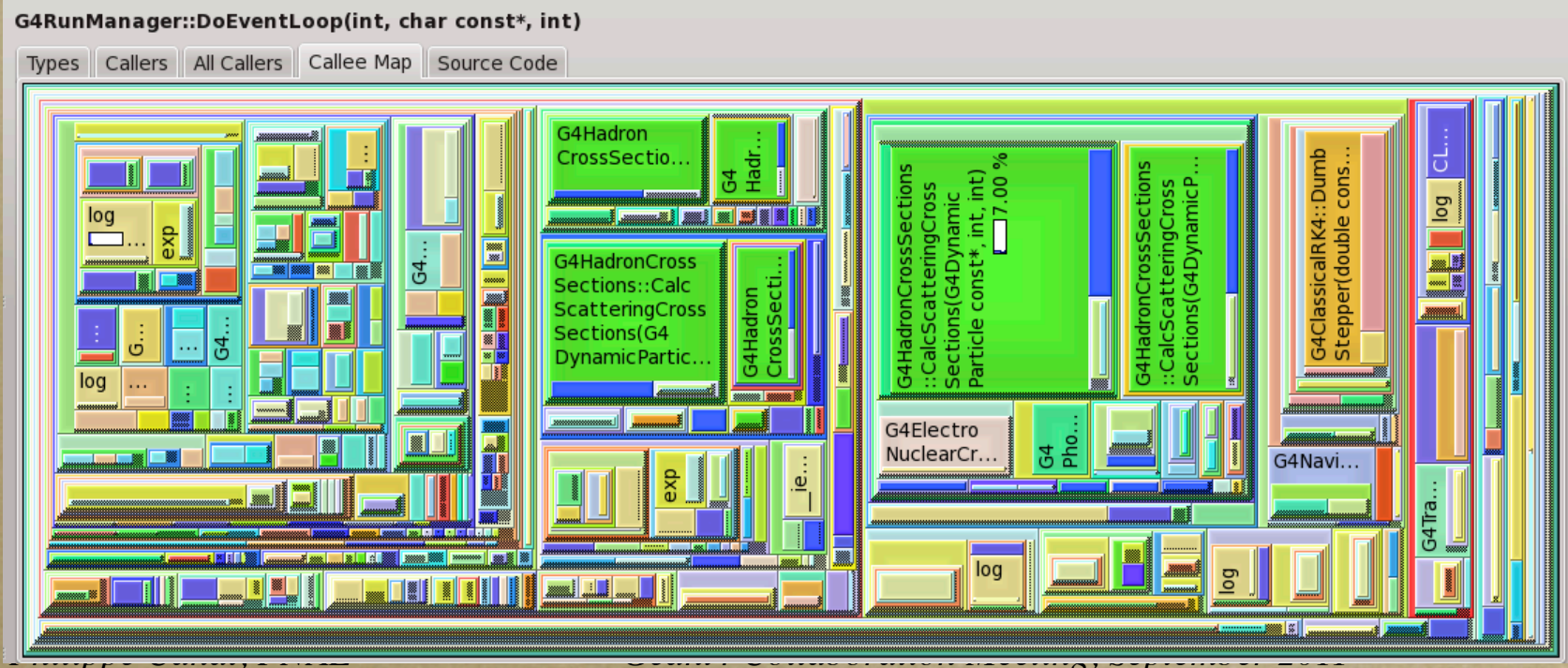
One Performance Study

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

Performance Opportunities

- *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 %
6.78 %
log
__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
log
exp
G4Hadron
CrossSectio...
G4Hadr...
G4HadronCross
Sections::Calc
ScatteringCross
Sections(G4
DynamicPartic...
G4HadronCross
Secti...
G4HadronCrossSections
::CalcScatteringCross
Sections(G4Dynamic
Particle const*, int, int)
7.00 %
G4HadronCrossSections
::CalcScatteringCross
Sections(G4DynamicP...
G4ClassicalRK4::Dumb
Stepper(double cons...
G4Navi...
G4Electro
NuclearCr...
G4
Pho...
G4Tra...
CL...

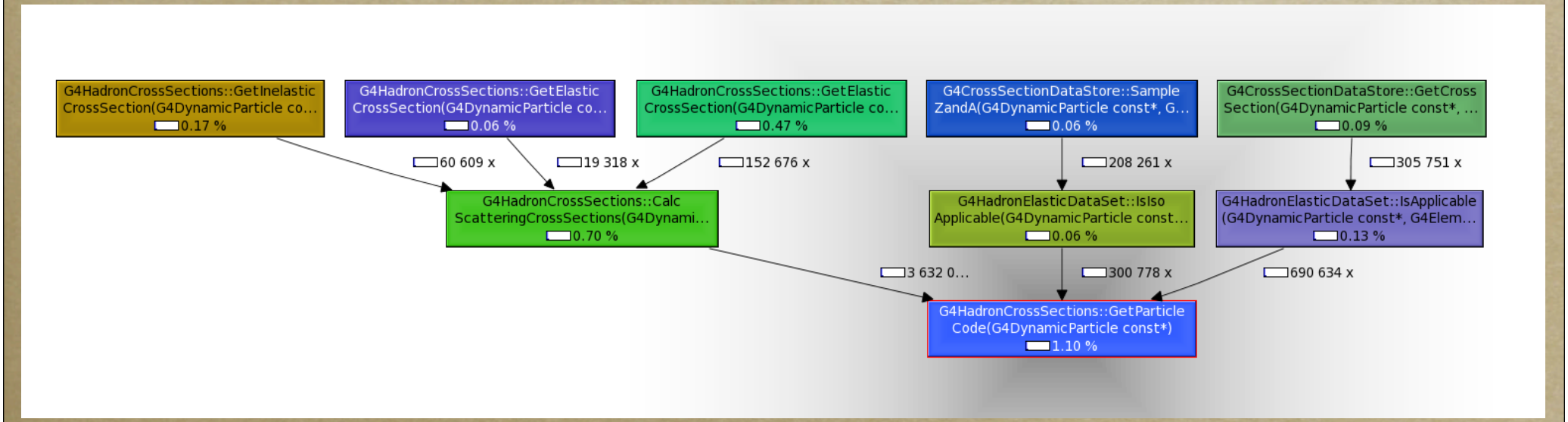
Opportunities

- *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.s
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.*

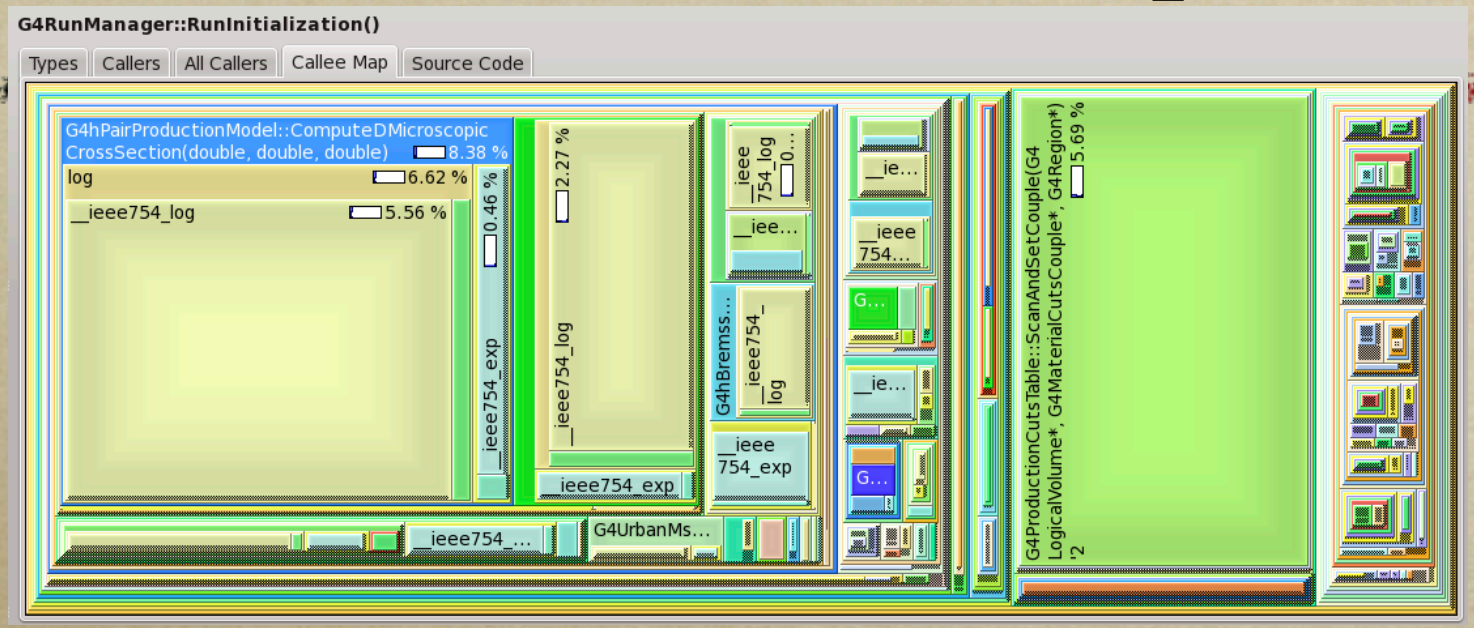
CPU Efficiency

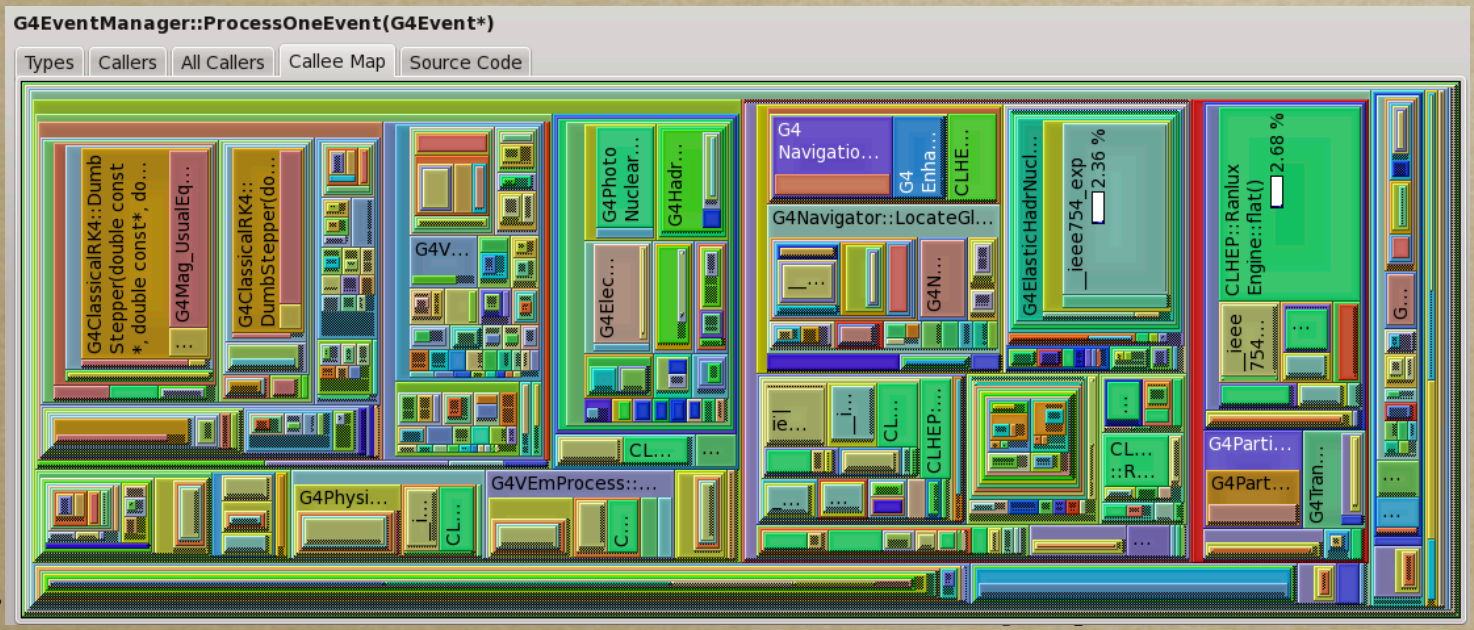
AMD's CodeAnalyst performance Analyzer can calculate the number of instructions per CPU clock cycles in each libraries.

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





Conclusion

- *Geant4MT*
  - *Can save significant memory.*
  - *Still some unresolved instability.*
  - *Significant CPU cost.*

Conclusion

- *Geant4 Performance*
  - *No clear easy-to-remove hotspot.*
  - *Some improvement opportunities left.*
  - *Likely to require structural changes for significant increases.*