Geant4 Computing Performance Activities

V. Daniel Elvira (Fermilab)

for the G4 Users and Performance Teams

The G4 Computing Performance Task

Back in February 2010, J. Apostolakis asked me to organize a G4CPT within the G4 Collaboration

Not a "Task Force" but an open ended activity organized as a group with regular meetings every ~ 6-8 weeks

Charge

(a) Profiling to identify bottlenecks in Geant4 based on main stream applications. We need to discuss profiling tools, what we want to measure, metrics. EM, Geometry and hadronics are the areas more involved in CPU usage.
(b) Code reviews geared towards improving computing performance and coding practices.
(c) Establish computing performance activities with the medical and space G4 communities.
(d) Identify issues in multi-core-multithread G4. ←

Not discussed beyond the first meeting – See Tuesday plenary

Ideas discussed during the first meeting

- Define a set of profiling tests/metrics.
- Profile simple examples and big experiments regularly, i.e. benchmark every candidate release/patch.
- Store information in a database for consistency and to track history
- Important to have a profiling tool with few dependences that can also be used outside of HEP.
- Identify unique features, advantages/disadvantages of each profiling tool to define profiling strategy and guide new users

List of Top Problems to Investigate

1. Memory Allocation
    * Navigation (G. Cosmo working on a fix - ATLAS, CMS testing)
    * Bertini (Mike Kelsey working on mem/speed improvements)
2. EM Physics Package
    * Optimization of parameters in applications
    * Revisit physics algorithms in Geant4 code: optimizations, approximations
    * Multiple scattering
    * Code review
3. Navigation speed and memory use in Voxel geometries and when handling large numbers of materials (brought up by the medical community)
4. Ion-ion inelastic models speed and memory use (medical).
5. Propagation in Magnetic Fields
    * Code review (done - no low hanging fruit from the programming side)
    * Testing/validation/profiling new steppers (ATLAS is testing Nystrom)
6. Hadronic cross-sections
    * Code review
7. Precompound/de-excitation
    * Code optimization. Many log/power functions are called. Many classes.

Challenge: define, staff, initiate remaining projects

G4CPT twiki

https://twiki.cern.ch/twiki/bin/view/Geant4/G4CPT

TWiki > Geant4 Web > Geant4 > G4CPT (10-May-2010, VDanielElvira)

Geant4 Computing Performance Task

Contact Person
Mission
Meetings
Profiling information
Code Reviews
List of Top Problems to Investigate

Contact Person

- Daniel Elvira

Mission

The G4CPT is not a task force but rather an open ended effort with the following objectives:

a. Profiling to identify bottlenecks in Geant4 based on main stream applications. We need to discuss profiling tools, what we want to measure, metrics. EM, Geometry and hadronics are the areas more involved in CPU usage.
b. Code reviews geared towards improving computing performance and coding practices.
c. Establish computing performance activities with the High Energy Physics, Medical and Space G4 communities.
d. Identify issues in multi-core, multi-thread G4.

Meetings

We intend to meet every 6-8 weeks. Agendas are available in indico.

Profiling information

Geant4 Tool Kit

HEP Applications

i. ATLAS
   - Profiling information on ATLAS can be found in the report CERN-LCGAPP-2010-01

ii. CMS
   - CMSSW_3_6_0_pre4/G4.9.3/slc5_amd64_gcc434 - 10 event high pT QCD
     - igprof perfticks and Intel PTU Basic Sampling profiles (annotated)
     - igprof total dynamic memory allocations profile

TWiki > Geant4 Web > Geant4 > G4CPT (24-May-2010, VDanielElvira)

0. LHCb
   - Presentation on performance using the Google memory allocator

Medical Applications

Space Applications

Code Reviews

i. CHIPS
ii. Propagation in fields

List of Top Problems to Investigate

Input received from a number of people in the developers and users communities. The medical community will start profiling applications in a more systematic way in the Fall of 2010. For the space community, speed is not the biggest issue at the moment but rather simulating small targets (< 1mm), tracking particles inside ~nm volumes, physics.

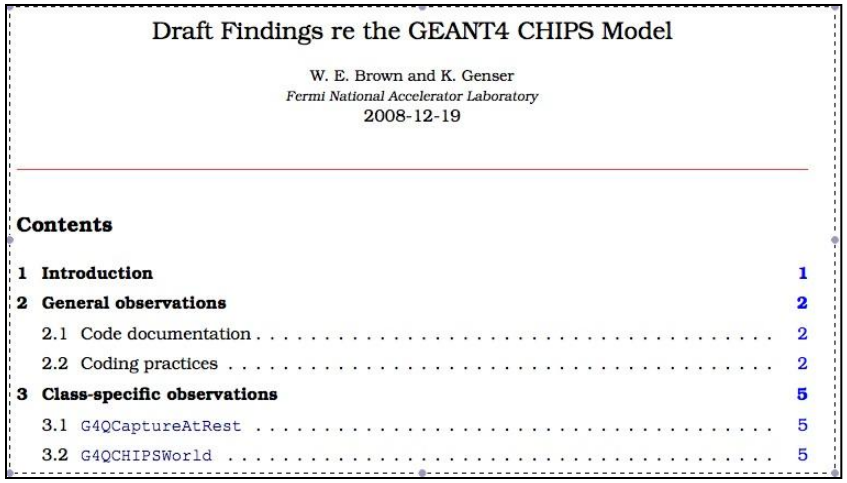
i. Memory Allocation
   - Navigation (G. Cosmo working on a fix - ATLAS, CMS testing)
   - Bertini (Mike Kelsey working on mem/speed improvements, see talk in hadronic meeting on 10-04-38
ii. EM Physics Package
   - Optimization of parameters in applications
   - Revisit physics algorithms in Geant4 code: optimizations, approximations
   - Multiple scattering
   - Code review
iii. Navigation speed and memory use in Voxel geometries and when handling large numbers of materials (brought up by the medical community among others)
iv. Ion-ion inelastic models speed and memory use (medical).
v. Propagation in Magnetic Fields
   - Code review (done - no low hanging fruit from the programing practices side)
   - Testing, Validation, profiling with new steppers (ATLAS is testing Nystrom)
vi. Hadronic cross-sections
   - Code review
vii. Precompound/de-excitation
   - Code optimization. Many log/power functions are called. Many classes.

-- VDanielElvira - 29-Apr-2010

Code Reviews

Past reviews (FNAL team): CHIPS and propagation in field code

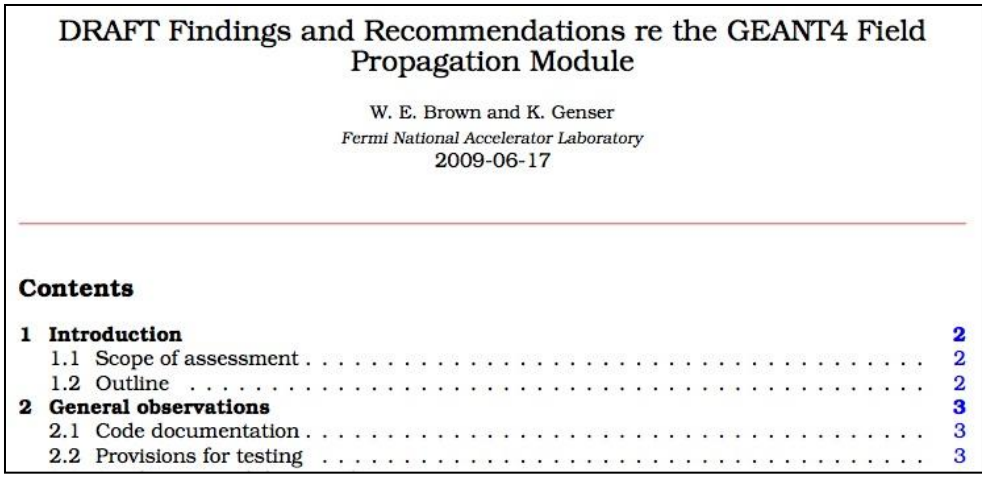
(documents linked from G4CPT twiki page)



Draft Findings re the GEANT4 CHIPS Model

W. E. Brown and K. Genser
Fermi National Accelerator Laboratory
2008-12-19

Contents



DRAFT Findings and Recommendations re the GEANT4 Field Propagation Module

W. E. Brown and K. Genser
Fermi National Accelerator Laboratory
2009-06-17

Contents

Most attendees to the G4CPT meeting agreed that
the EM physics code should come next, need to produce a charge
(coding practices, algorithms, physics approximations?)

K. Genser is, in principle, available for this but we need to agree on what we want from him. It should be sorted out this week.

Profilers: FAST

(developed by the FNAL team: A. Baldacchi, J. Kowalkowski, M. Paterno. J. Torola)

Goal: Routinely profile Geant4 using simple and complex HEP simulation applications (CMS chosen for obvious reasons) – K. Genser

Much of the tools and scripts and other machinery necessary for automation are complete and working. Final integration and debugging are still underway. Based on FAST (… previously known as SimpleProfiler)

FAST (Flexible Analysis and Storage Toolkit) features

- Set of tools for collection (Database) , analysis, display of performance data.
- Sampling based profiler that extracts call path, function call, library call.
- Graphical display of call paths, documented file format facilitates stat analysis.
- Profiles multi-process programs but not multi-thread.
- Few dependences.

<10% crash rate on CMS simulation application

See M. Paterno's presentation on Wednesday: usage, example
FAST documentation available at https://cdcvs.fnal.gov/redmine/projects/fast

Profilers: Perfmon

(HP tool, G4 work by Daniele Kruse, J. Apostolakis – CERN)

Goal: Routinely profile Geant4 using simple (20 GeV π on calorimeter) and complex HEP simulation applications (CMS) – Daniele Kruse

Fully automated procedure based on python script to instrument G4 user applications with Perfmon (HP) to monitor Geant4: runs the application with perfmon, analyze results (cvs output), check results with your favorite browser.

Perfmon features

- Modular Performance Monitoring. User Actions at the Step level to study different parts of the code separately.
- Study interaction of application with hardware.

Counts hardware event conditions that happen within a processor while the application is being run.

See J. Apostolakis' presentation on Wednesday: usage, example
For tool installation, unpack the following archive in your application directory:
http://dkruse.web.cern.ch/dkruse/G4_pfm.tar.gz

Profilers: IgProf

(Work by M. Kelsey - SLAC)

Goal: Review, fix, optimize Bertini Cascade code (Hadronic Physics)

Run test jobs of Bertini with IgProf frequently to compare with last baseline.
Make IgProf available to G4.

IgProf features

- Combines features of gprof and valgrind into common interface: CPU, memory.
- gprof report format for easy analysis
- No special compilation (uses shared-library substitution)
- Reports calls, memory churn (allocations), and memory leaks
- Text reports or navigable SQL-based Web files

Improvements

Main issue is memory churn. Many millions cycles creating/deleting small objects, passing/copying vectors. Fixes related to vector handling yielded:
5-8% reduction in memory churn in G4.9.3-ref-05.

See M. Kelsey's presentation on Tuesday in the Hadronic Validation & Development session: "Recent Improvements of Bertini Code"

Profilers: PTU from Intel

(Work by David Levinthal – Intel, J. Apostolakis)

The Intel PerformanceTuning Utility (PTU) is a tool that dips into the operating system and accesses information typically unavailable to other profilers.

G4, CMS and ATLAS are working with David Levinthal (Principal Engineer – Intel).

David's example: ATLAS G4 application, 10 GeV photons or electrons, 1k events.

As a result of his studies, David is offering advise on how to optimize large HEP software systems:

- Large OOP can suffer significant performance limitations (~50%), data parallel might be better.
- Branching (large if-then-else) within loops hurts performance (non optimal use of units in the cpu).
- Pre-fetching hurts sometimes.
- Large number of calls to functions can be harmful.

See J. Apostolakis' presentation on Wednesday.

G4CPT and the Space/Medical Communities

• Space (R. Weller): geometry and physics are presently of greater concern than absolute computational efficiency (speed/memory). While spacecraft applications will benefit from optimizations for HEP detectors.

- Ion-ion nuclear reactions up to cosmic ray energies.
- Physics in integrated circuit sized objects.

• Medical (J. Perl): there are memory issues due to large number of materials and voxels, speed issues due to the need to iterate for treatment plans (~10 min). MGH Clinical Plan takes 240 CPU hrs, 12 hrs in 20 CPUs.

- Speed of navigation in Voxel Geometries.
- Speed of swapping in and out physics tables for large numbers of materials.
- Speed of physics models, particularly for ion-ion inelastic models.

The medical community does not expect G4 core developers to resolve their issues but needs to rely on their statements on whether they'll do it to make according man-power allocations – No significant activity in the G4CP front yet but interested in profiling tools with few dependences that can be used outside HEP.

ATLAS Simulation Performance

(from Andrea Dotti)

- Report from a joint G4/ATLAS task force to identify simulation bottlenecks: CERN-LCGAPP-2010-01
- Profilers: valgrind (call trees) and hephaestus (ATLAS memory profiler):
  - Priority: reduce CPU usage
  - Also important: reduce memory churn (to reduce CPU)
- Main focus has been on "compromising" on details of the physics performance to reduce CPU usage.
- Technical aspects of coding have been touched only partially

ATLAS Simulation Performance

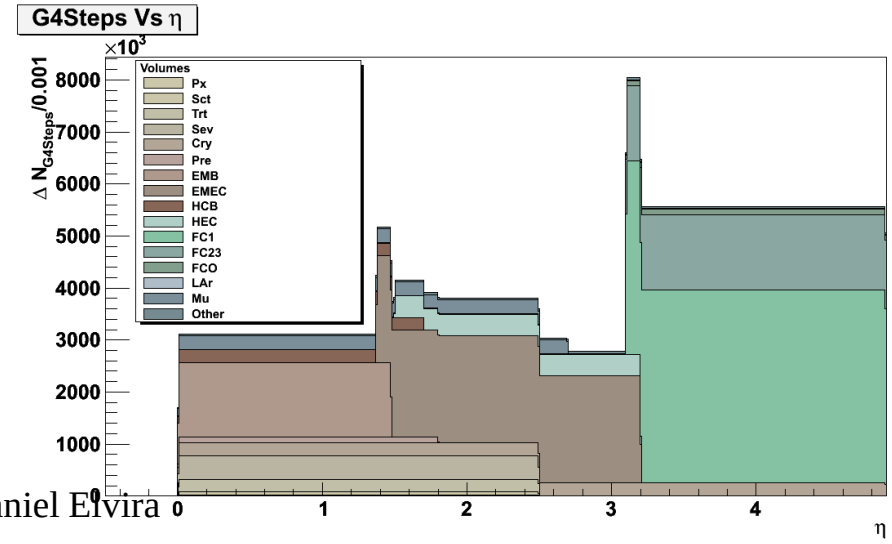
Number of G4Steps, units in 10^6	e-	e+	gamma	charged pion	proton	neutron	other	Total (%)
Px	2.32	0.24	2.37	0.68	0.14	0.30	0.21	*0.3%*
Sct	2.42	0.37	6.52	0.82	0.20	0.89	0.25	*0.6%*
Trt	5.37	1.05	40.20	1.82	0.50	5.46	0.53	*2.9%*
Sev	49.88	7.35	40.63	0.42	0.35	3.82	0.46	*5.4%*
Cry	48.92	6.48	49.01	0.34	0.40	7.38	0.41	*6.0%*
Pre	3.99	0.49	10.41	0.09	0.06	1.73	0.03	*0.9%*
EMB	79.94	11.76	66.26	0.74	0.56	32.59	0.38	*10.2%*
EMEC	137.13	22.73	114.97	1.34	1.03	63.34	0.65	*18.1%*
HCB	10.10	0.96	14.88	0.14	0.18	11.46	0.25	*2.0%*
HEC	17.49	1.67	23.01	0.23	0.30	20.79	0.58	*3.4%*
FC1	303.56	31.82	204.14	1.79	1.61	64.26	0.80	*32.2%*
FC23	90.15	13.18	42.67	0.91	0.83	87.99	0.72	*12.5%*
FCO	10.23	1.46	3.80	0.10	0.12	1.01	0.03	*0.9%*
LAr	2.50	0.25	2.97	0.06	0.11	0.54	0.05	*0.3%*
Mu	31.85	4.34	22.09	0.11	0.23	4.25	0.50	*3.4%*
Other	4.89	0.86	10.13	0.13	0.11	1.05	0.04	*0.9%*
Total (%)	*42.4%*	*5.6%*	*34.6%*	*0.5%*	*0.4%*	*16.2%*	*0.3%*	*100%*

- Event studies: ttbar and MB
- Basic assumption: the simulation time is proportional to the number of G4Steps
- Performance improvement should focus on e.m. showers in EM-calorimeters

… EM code review… ?

ID, Calos

- E.m. physics in calorimeters is the key element to optimize
- In particular most of the time is spent in "forward" region of the detector. This region has also less stringent physics requirements



ATLAS Simulation Performance

Improvements/potential improvements

- Review of particle cuts in the forward region to gain time performance (not started)
- Re-write of (custom) geometry volumes in forward region (ongoing)
- Bertini (in particular memory churn) was identified as an issue. (Need to test recent improvements by M. Kelsey.)
- Multiple Scattering (Urban Model 2) is the most time consuming process (review?)
- Two utility methods are called several times and are responsible for 2% of the CPU time: G4Track::GetVelocity and G4PhysicsVector::GetValue. (To be tested again with new improvements for >= G4.9.3.)
- The new G4Nystron stepper (and appropriate B-Field caching) helped a lot ATLAS simulation to solve speed and memory issues related to tracking in magnetic field. (Solved.)
- Memory churn drastically reduced with new G4TouchableHistory/G4NavigationHistory. (Solved.)

CMS G4 Performance Studies

(from Sunanda Banerjee)

QGSP_BERT_EML adopted by mid 2009 for the same physics and with CPU time reduced by ~20% (QGSP_BERT_EMV -> QGSP_BERT_EML).

- Use of "ApplyCut" option saved CPU and caused no significant impact on physics
- Gave up optimization options (shower lib in HF) to handle anomalous hits in the calorimeter PMT/HPDs: ~30% increase in CPU time, also memory size .
- Peter Elmer (CMS) suggested improvements in the geometry code by reducing the use of some of the trigonometric functions.

Problems:

• Time growth due to usage of Geant4 tracking of hadrons in the forward detectors.

• Increase in memory usage at runtime due to usage of shower parametrization (Gflash) in a part of the code (EM particles through HF).

CMS G4 Performance Studies

Application: FullSim under CMSSW380, G493p01.
G4 Version: G4.9.3.p01, CLHEP2.0.4.2.
Physics Lists: QGSP_BERT_EML
Profiler: Timing and SimpleMemoryCheck services.
Architecture: Intel Xeon E5335 @ 2 GHz dual-process dual-core setup, slc5, 32bit, gcc4.3.4

Sample	Type	Time (s)	Memory (MB)	Data (GB)
MinimumBias	Default	14.6	482	0.17
	Extend Fwd	21.2	609	0.17
ttbar	Default	109.1	490	1.32
	Extend Fwd	134.3	553	1.32
High p_T QCD	Default	238.6	520	2.34
	Extend Fwd	240.8	531	2.25

Peter Elmer's code "Performance Survey": comparison of 32/64bit builds, algo code performance improvements, reduction of dynamic memory used, opportunities for parallelism, explore new CPU counters tools.

Use IgProf, PTU

Summary and Outlook

- Significant progress testing and automating different profilers on simple and complex G4 applications
- Slow progress on reviewing and fixing code based on what is learned.
- List of top problems not fully covered – man power issues.

What's next?

- Finish automation of profiling tools for use in G4 as well as documentation on usage
- Define a set of profiling/performance tests to be run frequently on some subset of ref/candidate/public releases – staff the effort.
- Cover all elements in list of top problems to investigate and those that arise from the systematic and frequent profiling of G4 applications
- Do we want a hypernews list or similar to post/follow progress?