Summary of Geant4 Computing Performance Activities

V. Daniel Elvira (Fermilab)

for the G4 Users and Performance Teams

Outline of Session

Summary presentation on recent computing performance progress by users and collaborators:

- Geant4 Application for Tomography Emission (GATE).
- CMS HEP experiment at the LHC.
- ATLAS HEP experiment at the LHC.
- Fermilab G4 performance team.

Discussion with opportunity to ask questions and bring up topics

Low Energy EM Perf. Improv. (GATE)

Nicolas Karakatsanis (NTU-Athens) will make a detailed presentation on improvements to the low energy EM processes performance the next week.

- Geant4 low-energy electromagnetic physics processes
  high performance cost compared with standard EM processes

- Initial profiling results on a G4 Appli for Tomography Emission (GATE) performance benchmark
  G4LogLogInterpolation::Calculate method
    at each iteration step, five log10 and one pow10 calculations are required
    61% of the total running time spent on this method
    specifically, 42% spent on log10 method and 16.2% on pow10 method

Revision #1: New G4LogLogInterpolation::Calculate method – reduced number of log10 calls required per iteration from five to four resulted in 10% improvement in CPU time.

Revision #3: Modified data loading and interpolation mechanism – reduced total running time by 33.5%.

Low Energy EM Perf. Improv. (GATE)

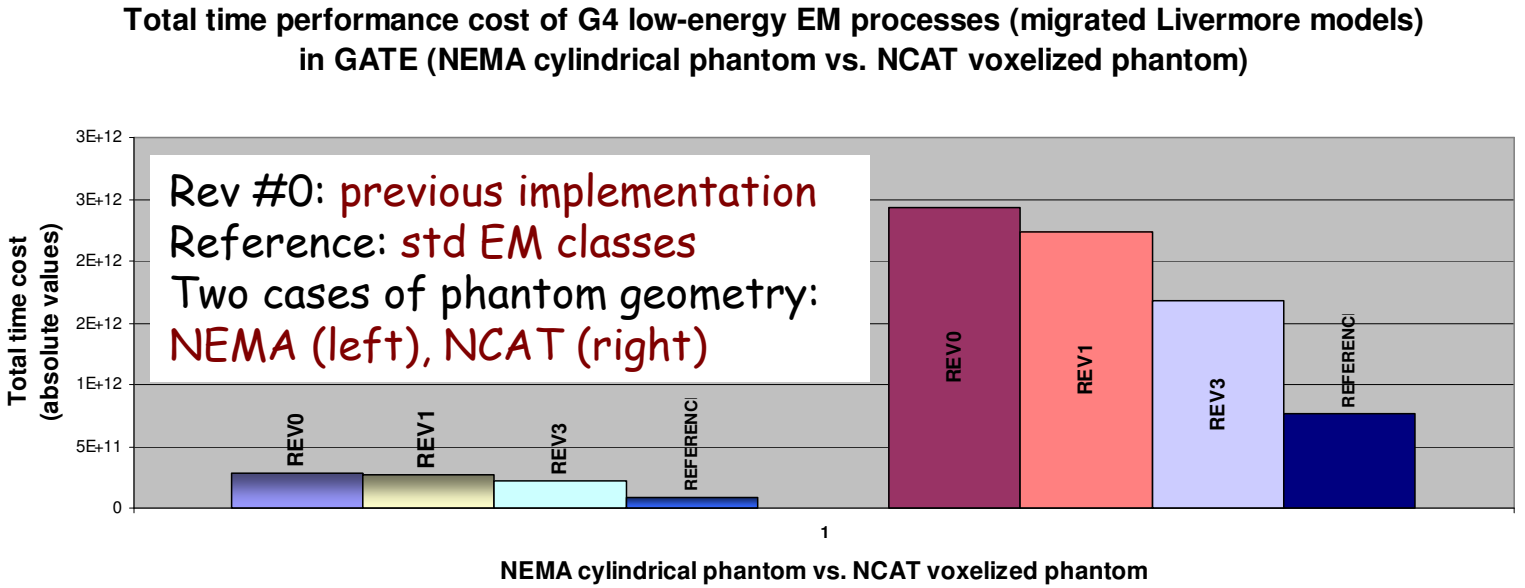
Application: G4 Application for Tomography Emission (GATE).

G4 Version: emlowen-V09-02-54 tag included in geant4-09-02-ref09

Package/Process: low energy EM physics.

Profiler: Valgrind.

Improvements: 10% (Revision #1), 33.5% (Revision #3).



FNAL G4 Performance Team

- Past activities (2008): See D. Elvira's presentation in Kobe - 2008

  - In *G4Qhadron & G4QNucleous* (CHIPS model), std::vector<G4Double>* T was replaced by std::vector<G4Double> T with a ~1.5% timing improvement in the CMS offline environment
  - Reorganization of *G4ElementaryParticleCollider* (Bertini), removing 20 of 21 data members resulted in ~4% timing improvement.

- Recent activities (2009):

  (I) Profiling

  Event irreproducibility correlated with bifurcation of event processing times across many of the same jobs depending on computer architecture traced to different "firmware" implementations of sin (or, in general, transcendental) functions for different CPU brands.

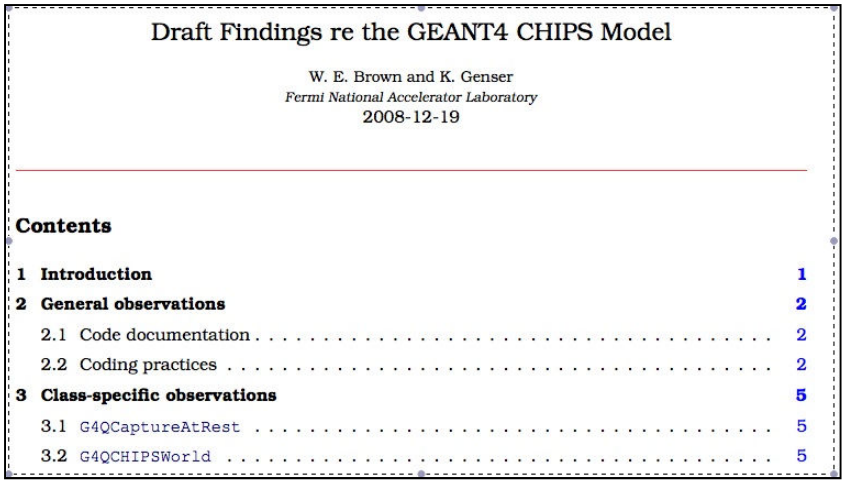
  (II) Code Reviews (see K. Genser's talk on Monday):

  Review of the CHIPS hadronic model and particle in Field Propagation Modules

FNAL G4 Performance Team

Conclusions from CHIPS and propagation in field code reviews

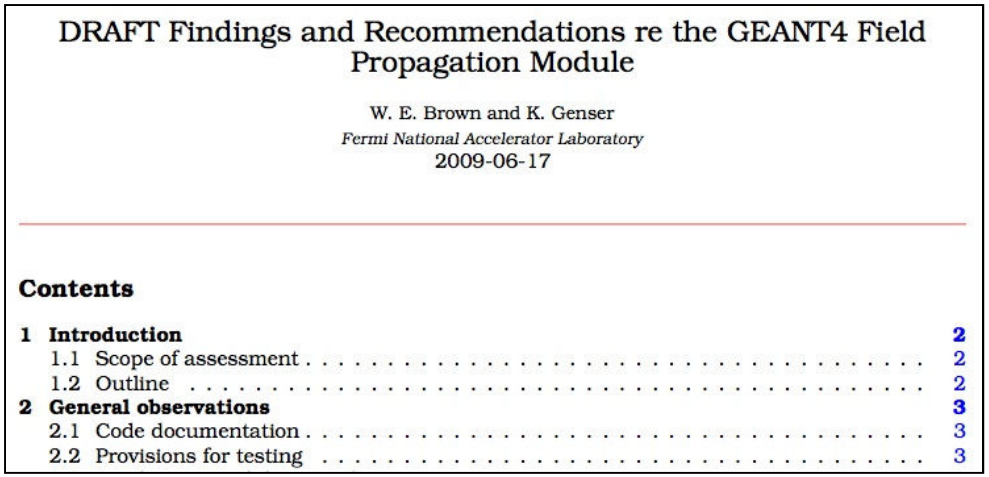
- Most of the comments were related to the C++ coding techniques having impact on code robustness and maintenance
- Among other findings: a potential ~0.5% timing improvement in CHIPS by replacing a collection of pointers to objects with collections of objects.
- No significant opportunities for timing improvement noted in Field Propagation Module.



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

FNAL G4 Performance Team

Application: G4 standalone tests and CMS simulation application.
Software Versions: G481p01 (2008 studies), G491p03 (CHIPS), G492p01 (field propagation), QGSP_EMV & QGSP_BERT_EMV.
Package/Process: CHIPS, Bertini, field propagation

Profilers: "Simple Profiler" and "Performance Data Base" (developed at FNAL).

About SimpleProfiler:

- It collects data (unbiased measurements) for the program of interest 100 times per second, captures the address of the current function and the address of each function in the call stack. (It thus collects full call path information, not provided by other tools.)

In post-processing, the function names and library location for each function and is determined and all the information is loaded into an SQLite3 database.

CMS G4 Performance Studies

QGSP_BERT_EMV adopted by mid 2008 for better physics but ……
CPU time and output data size increased by ~50% (QGSP -> QGSP_BERT).

- The "EMV" version saved ~15% of CPU time – no significant impact on physics.
- FNAL/G4 team improved time performance by ~6% in Bertini, CHIPS.
- Peter Elmer (CMS) improved memory allocation mechanism (Bertini) and saved an additional 10%.

Problem:

• Time/event size growth traced to a significant increase in the number of calorimetric hits associated with the Bertini cascade model, compared to QGSP.

• Increase in memory usage at runtime due to increased number of allocations when creating temporary calorimetric hits during the development of the shower.

CMS G4 Performance Studies

Solutions:

CMS readout is not sensitive to what happens > 500 ns after t0. Time cut-offs were defined:

- On a G4Region basis to:
  - stop particle tracking (SteppingAction).
  - prevent the insertion in G4 particle stack (StackingAction).

  As a side effect the long lasting tracking of slow neutrons is suppressed.

- For calorimeter sensitive detectors to reject transient CaloG4Hits (118 bytes) before they are made persistent PCaloHits (26 bytes).

Gains in CPU time, event size, and the memory usage, in terms of memory footprint and in number of allocation/de-allocation cycles.

On a 100 ttbar event sample the memory footprint of the simulation job was reduced by about 30 MB.

CMS G4 Performance Studies

Application: FullSim under CMSSW312, G492p01.

G4 Version: G492p01, CLHEP194.

Physics Lists: QGSP_EMV, QGSP_BERT_EMV, QGSP_BERT_EML

Cuts: time < 500 ns, hit energy < 10 (15) KeV in Ecal Barrel (Endcap).

Profiler: Timing and SimpleMemoryCheck services, IgProf, Valgrind.

Architecture: Intel Xeon 5160 @ 3 GHz dual-process dual-core setup, slc4, gcc3.4

1000 single pions of E=50 GeV fired in the CMS calorimeters only geometry

Physics list	⟨CPU⟩ (s/ev)	⟨EBHits⟩ (B/ev)	⟨EEHits⟩ (B/ev)	⟨ESHits⟩ (B/ev)	⟨HcalHits⟩ (B/ev)
QGSP_EMV no cut	0.94	2701.02	2454.51	45.919	1220.87
QGSP_BERT_EMV no cut	1.45	8233.88	6039.22	53.688	5541
QGSP_BERT_EMV cuts	1.38	3304.85	2934.21	51.403	4734.78
QGSP_BERT_EML cuts	1.13	3297.68	3077.16	47.111	4726.4

ATLAS G4 Performance Studies

QGSP_BERT replaced QGS_EMV in 2008 for better physics but ……
CPU time and output data size increased by a factor of ~2.5 (150%) .

Mitigation measures were taken, as for CMS:

- Neutron time cut < 150 ns saved 20% CPU time. Now playing with the idea of a neutron energy cut.

- Total memory use (VMEM) trimmed by ~150 MB:

  - Changed voxelization options in two large volumes: 50-70MB
  - Changed tcmalloc versions: 7-10MB
  - Changed to light Oracle libraries: 20-40MB
  - Reduced physics tables to go only up to 7 TeV: 5MB
  - Changed ROOT file writing options: 5-10MB
  - Changed doubles to floats in one key place in the code: 13.5MB

- Started to look at alloc/de-alloc per event.

  - Used the Hephaestus tool to find and fix a problem in G4String::operator==(), which was creating temporary objects totaling 600 MB/event in ATLAS.

ATLAS G4 Performance Studies

Magnetic field access is the big killer (20% of simulation time):
Plans to improve (a new Runge Kutta stepper, magnetic field caching, a new 'stepper dispatcher').

A couple of other technical improvements worth mentioning:

- Changing from G483 to G492 gave a ~10% CPU improvement.
- Changing to slc5/gcc43 gave us a ~20% CPU improvement.
- A patch to build EM processes in a vector saved 3.5 minutes in initialization

ATLAS documentation on validation/monitoring benchmarking:

http://atlas-computing.web.cern.ch/atlas-computing/packages/simulation/geant4/validation/Comparisons.html\
http://atlas-computing.web.cern.ch/atlas-computing/packages/simulation/geant4/validation/Studies.html
By-domain memory monitoring:
http://atlaspmb.cern.ch/rtt-mon/data/sim-dev-prod-i686-slc4-gcc34-opt/results/histplot_domains_vmem.png

ATLAS G4 Performance Studies

Application: ATLAS FullSim application under ATHENA, slc4, gcc34

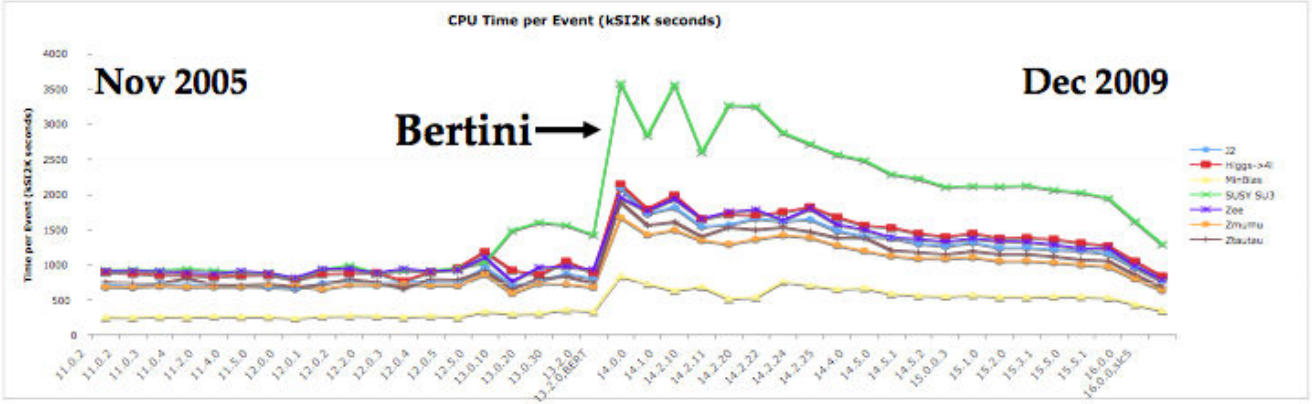
G4 Version: G483p03, G492p01, CLHEP1942.

Physics Lists: QGSP, QGSP_BERT.

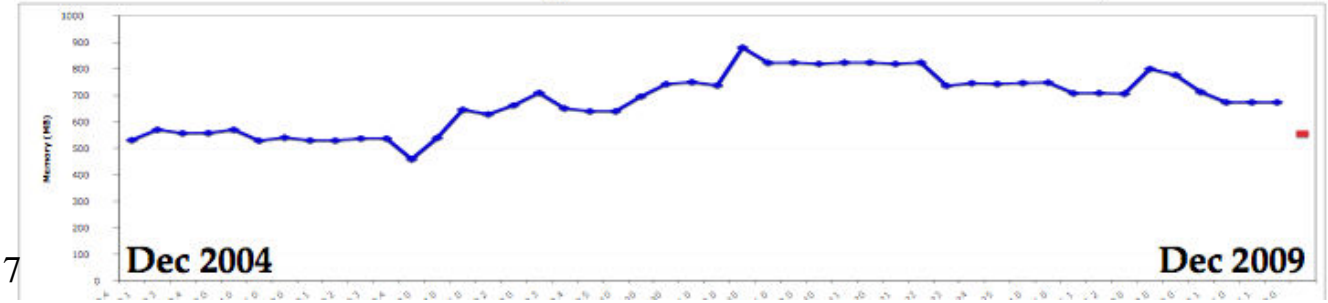
Cuts: time < 150 ns time cut for neutrons.

Profiler: Valgrind, Memprof, Perfmon, Hephaestus.

CPU Time
Initial ~ Final:
500-1000 kSI2K



Memory
Initial – 525 MB
Final – 650 MB



Some ATLAS/CMS Differences

ATLAS and CMS simulations have both achieved high levels of physics accuracy and technical robustness. CMS application is significantly faster because…

- ATLAS' geometry is more complex than CMS'.

- CMS uses by default a shower library in the Forward Hadron Calorimeter (HF) and will probably move to a GFlash shower parameterization both in the HF and ZDC detectors. ATLAS uses G4 showers everywhere.

- CMS uses the faster QGSP_BERT_EMV physics list. ATLAS EM sampling calorimeter is more sensitive than CMS crystal EM calorimeter to multiple scattering. ATLAS uses QGSP_BERT.

- CMS uses field caching to access magnetic field values: re-evaluated field only if G4Step size > 1mm. ATLAS does not do field caching.

G4 Performance Summary and Issues

- The LHC experiments have (successfully) spent significant effort to improve CPU time performance and memory usage after migrating to better physics models

- The GATE & EM G4 colleagues have significantly improved the time performance for low energy EM processes

Some Topics for Discussion

- Time and memory footprint studies; any more input from users? Where do we go from here?

- Would we benefit from more regular profiling; other tools that were not mentioned?

- Applications – more ideas on approximations to improve performance? User code performance tips: https://twiki.cern.ch/twiki/bin/view/Geant4/Geant4PerformanceTips

- G4 Tool kit improvement – code reviews; we appreciate input from users on what they learn about core G4 code as they profile their applications.

- Multi-threading; any plans from users?

Some Topics for Discussion

- Valgrind (ATLAS, GATE/G4)
- IgProf (CMS, ATLAS?)
- SimpleProfiler (G4 FNAL team)
- Perfmon (ATLAS)