Minutes of the G4CPT 2010-03-08

Introduction (Daniel Elvira)

Daniel E presented the draft zero of the charge to the G4CPT that emerged from conversations with a group of G4 developers. Input from this meeting will be incorporated.
Joseph P: the word HEP should be moved from (a) to (c). As it reads now, the charge seems HEP centric.

Action Items:
- Daniel will create a twiki page for the G4CPT.
- Daniel will fix the text of the charge and uploaded to the twiki page.

This first meeting is dedicated to general discussions on the organization, activities, and priorities of the task. In addition, we focus on HEP usage (ATLAS, CMS talks). A second meeting will focus on Medical and Space usage and needs.

Action Item:
- Daniel will talk with Robert Weller and Joe Perl over the phone as a first step to organize a Medical/Space focused G4CPT meeting.

Robert W: what's the timescale for the activities of the task?
Daniel E/John A: one year or two but it is open ended. The results will feed the architecture review process but the design revision is not part of the task.
Mark F: the effort of profiling G4 for medical and space usage may be significant; who will do it?
Joseph P: the effort is ongoing in the case of the medical community (Japan).
Robert W: the space community has not paid to much attention to computing performance yet but to physics accuracy. For example, they are interested to simulate volumes of the order of 10 nm^3 but have not profiled these applications yet.

ATLAS Detector Simulation Performance (Andrea Dotti)

ATLAS/G4 group was setup to assess computing performance of the ATLAS simulation application. There is an internal report that will become public soon:
ATL-COM-SOFT-2010-008
CERN-LCGAPP-2010-01.

The tools used are Valgrind (Callgrind) for profiling and Hephaestus (developed by ATLAS) for memory .

The study has identified a few areas for improvement.

The ATLAS specific are (info may be useful to other experiments/projects):

- Use of more aggressive production thresholds and eta cuts.
- Re-design EMEC custom solid.
- LArG4Identifiers redesign to reduce memory churn.
- Try new stepper: G4Nystron (available in G493); implement field caching.

The G4 specific are:

- Re-design of BERT code to reduce memory churn.
- Revisit G4TouchableHistory, G4NavigationHistory to reduce memory churn.
- Revisit the following functions responsible for large CPU contributions: UrbanModel2 (4%), cross sections retrieval (5%), G4Track::GetVelocity and G4PhysicsVector::GetValue (2%), use of exp and log (25).

Gabriele C: points out that G4TouchableHistory is fixed in G493.
Michael K: mentioned that he would re-write the Bertini Cascade code that can be dramatically improved during the upcoming months by fixing classes that are copied and duplicated. He will proceed "adiabatically" first making changes that will facilitate further re-design.
Question: how do we quantify memory churn? Hephaestus returns an output Valgrind style (number of calls to malloc/size of object) which can bee read with standard graphical tools.
Peter E: mentioned that a 5-10% improvement in time performance could be achieved just by removing the abuse in the use of vectors in navigation history copying.

It was noted that ATLAS does not profile the G4 simulation application very often or systematically. The information is typically available in presentations (slides). This time it was included in a report document because there was an organized effort ATLAS/G4 to assess computing performance.

Action Item:

- John A mentioned there is an effort in ATLAS to look at the number of cycles per instruction, memory cache misses, etc using Perfmon is under way and results should be presented and discussed during the next meeting.
- Daniel will write a proposal for a repository where all projects (HEP and non-HEP) can upload their profiling results and share them with the rest of the G4 Collaboration and the users community. It may be based on twiki pages.

G4 Computing Performance in CMS (Peter Elmer)

There is a computing performance effort in CMS lead by Peter. The studies he presents are focused on technical performance issues which should leave the physics unchanged.

The goals of the CMS effort are:

- Compare 32bits and 64bits builds toward deployment later in 2010.

- Identify opportunities to improve algorithmic performance, data structures in memory, dynamic memory use, etc.
- Document findings and recommendations.
- Improve CMS understanding of newer tools based on CPU performance counters.
- Systematically investigate to "top" functions to investigate and improve.

CMS uses Igprof for profiling and Perfmon/pfmon for memory, Intel Performance Tuning Utility (PTU). Peter showed numerous examples on Igprof performance profiling (32bits vs 64 fits, heap, Intel PTU source view which is both useful and fast).

Peter also explained the CMS "multicore" strategy:

- Phase 0: run one job per core.
- Phase 1: deploy basic feature for enabling memory sharing between processes.
- Phase 2: exploit more fine-grained parallelism (threading, openmp, etc).

Peter suggests that a systematic "performance survey" of G4 would be a very useful activity for this G4 Computing Performance Task. (Profiling, discussion, concrete changes.)

Robert W: Has CMS compared performance with a build using the Intel compiler?
Peter E: Each time he tried the Intel compiler there was a problem and the code would not build. In addition, many features are on by default that should not be. He also mentioned that it would be more interesting to compare different features in gcc. For example gcc vs gcc fast math.
Andrea D: ATLAS does not have a build using the Intel compiler.
John A: Only 15% of the cycles were floating point operations and this poses a limit to the improvements that can be achieved.
Peter E: If we remove the first "n" most basic problems and other things that we can fix will become more important.
Peter E: What are the plans to make multi-core G4 available?
Gene C: Alpha version by the end of the year. Gene provided a few slides with details of the multi-core G4 project which Daniel make available in the meeting agenda.

G4 profiling using the CMS simulation application (Krzysztof Genser)

Krzysztof represents the FNAL G4 performance team that is upgrading and automating the process used for profiling of Geant4 in the past, when doing code reviews and code optimization (Jim Kowalkowski, Marc Paterno).

The goal is to routinely profile G4 using a mainstream application. CMS application was selected given the existing expertise at FNAL.

The profiling is done using SimpleProfiler (C++ dynamic library collecting detailed call stack samples using libunwind plus a set of auxiliary libraries). SimpleProfiler is part of the FAST package, a collection of tools for gathering, managing, and analyzing data about code performance. See:
https://cdcvs.fnal.gov/redmine/projects/show/fast
The data is uploaded into PerformanceDataBase. Time and function call measurements are the result of a number of runs producing results stored in the database.

Currently migrating the system from old to new version of SimpleProfiler. Debugging situations where SimpleProfiler does not recognize call stack functions or when libunwind incorrectly processes call stack or crashes.

Action Items:

- Jason Tortola, Krzysztof Genser: modify database to make classification of run and build configurations more convenient.
- Jason and Krzysztof will add scripts for automatic performance data analysis and presentation including plots pertaining comparisons between different collections of test runs.

General Discussion (all)

Andrea D: ATLAS does not store performance information in database.
Peter E: CMS is moving in the direction of systematic set of tests and storing in database. Peter believes, however, that this is not as important as it is to look at the existing profiles systematically and address the problems.
Daniel E: What would happen with CMS and ATLAS profile work if the experiments decided to freeze the simulation (version) for a few years to provide stability to physics analysis?
Peter/Andrea: One would think that the simulation groups would continue to test and profile new versions of G4.
Joseph P: It is important to have a profiling tool that can be used outside of HEP and help us see the changes in G4. Michael K agrees and suggests we should focus on profiling with regression tests with a tool that has no dependencies.
Jim K: SimpleProfiler only has one or two dependencies.
Marc P: The SimpleProfiler tool can be downloaded and tested (core code, script to manipulate data, database application).
John A: We should profile more regularly development releases (not all) but benchmarking every candidate release and patch. We should also understand correlations between improvements in simple tests and big experiments; why we may see 15% improvement in a simple case and only 7% in the big experiments.
Michael K: we could use twikis to display summary plots with results from experiments but a database approach which records history such as SimpleProfiler for internal G4 use.

Krzysztof G: the SimpleProfiler infrastructure is not a monolithic thing. SimpleProfiler can be used by any executable to produce useful information. No need to use auxiliary tools to fill database or analyze the data.
Peter E: We should have a forum for discussion (hypernews?).
Joseph P: How much o the task work would be G4? The medical community is interested in geometry, navigation.
John A: Who will undertake the profiling in areas other than HEP? Users will have to do a significant part of the work.
John A: suggests to put together a twiki page with the advantages/disadvantages of each profiling tool to guide new users.
Daniel E: What packages need a code review?
All: EM physics packages.

Action Items:

- Daniel will collect from different people (developers/users) what they think are at the top problems to investigate based on available information. Then we'll how to decide how to proceed.