Krzysztof Genser/Fermilab
For the Fermilab Geant4 Performance Team

Geant4 Profiling using CMS Simulation Application

Fermilab Geant4 Performance Team

- Walter Brown, Mark Fischler, Krzysztof Genser, Jim Kowalkowski, Marc Paterno, Ron Rechenmacher, Jason Torola
  - a part time involvement for all the members
- With thanks to Sunanda Banerjee, Julia Yarba, Liz Sexton-Kennedy for their assistance.

Project Goals/Description

- The goal is to routinely profile Geant4 using a realistic application.
  - Given Fermilab involvement in CMS using CMS software application was a natural choice.
- Over last few months we have undertaken an effort to upgrade and to automate as much as possible the infrastructure/process we used for profiling of Geant4 in the past (e.g. when doing code reviews or optimizing the code).
- The application is built/run on set of Fermilab US CMS nodes with CMSSW software installed.

Profiling Infrastructure

- The full infrastructure is a set of scripts which take as an input a Geant4 tarball, a version of the CMSSW release, a configuration file specifying simulation parameters and an event generator input file. The Geant4 tarball replaces the Geant4 version used by CMS for a given CMSSW release.
- The output is a set of performance related numbers/plots.
- The simulation package is build on an interactive US CMS build node and run on a Condor farm. The build and run parameters are recorded in a database for each run.
- As of now QGSP_BERT physics list & Z'->dijets Pythia event input file are used.

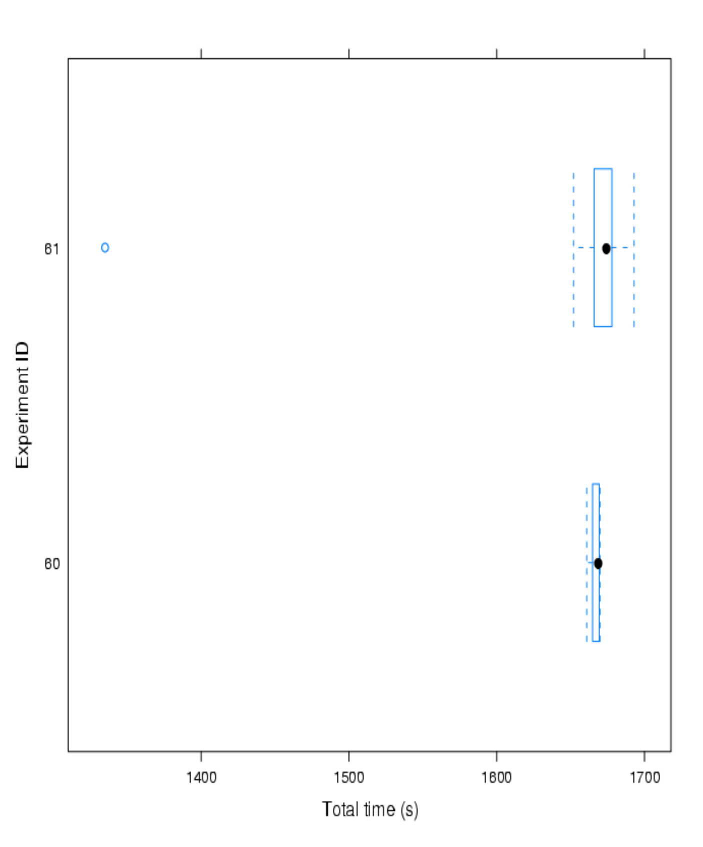
Profiling Infrastructure, cont'd

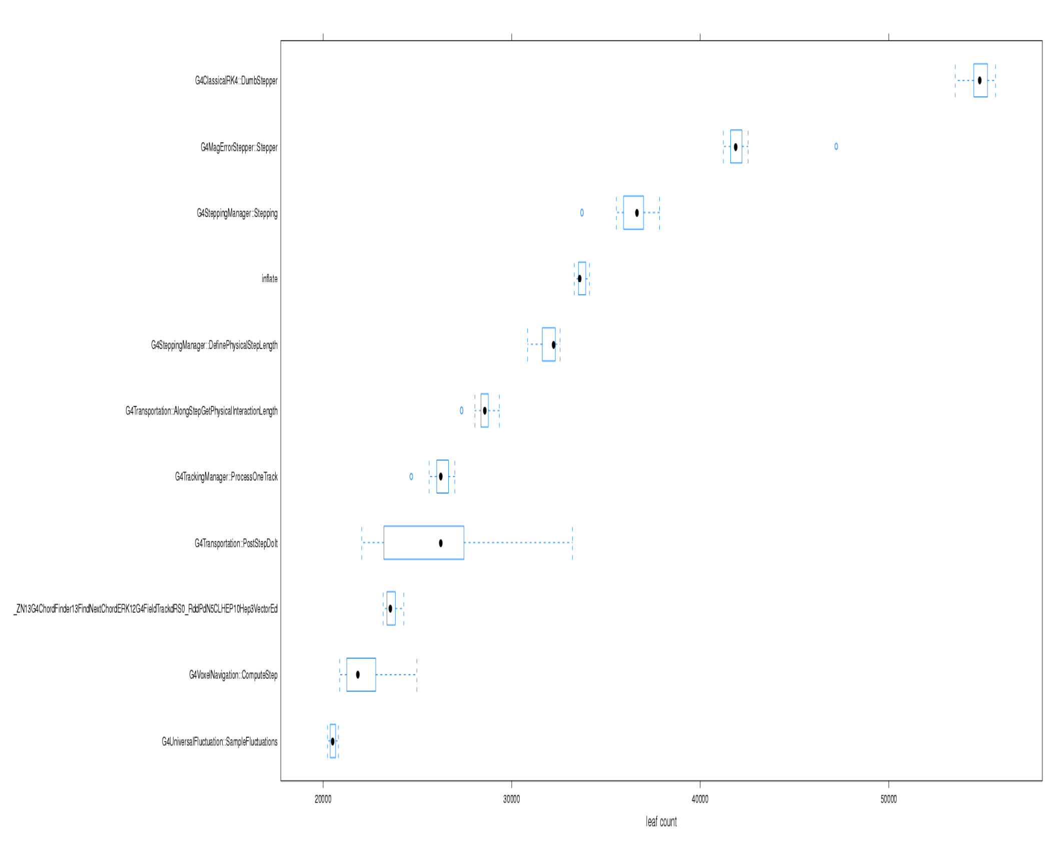
- The profiling itself is done using SimpleProfiler
  - C++ dynamic library collecting detailed call stack samples, using "libunwind" plus a set of auxiliary tools
    - SimpleProfiler is a part of the FAST package, a collection of tools for gathering, managing, and analyzing data about code performance. See: https://cdcvs.fnal.gov/redmine/projects/show/fast
- The data is uploaded into PerformanceDataBase
  - system (written using Ruby on Rails and MySQL) for recording performance run data and handling the assessment of the statistical significance of the timing differences.
- Timing and function call measurements are a result of a number of runs on the previously mentioned Condor farm and uploading the information into the database.

Work In Progress

- Migrating the system from the old to the new version of the SimpleProfiler
- Debugging situations where SimpleProfiler does not recognize call stack functions or when libunwind incorrectly processes call stack or crashes.
- Modifying the Performance Data Base to make classification of run and build configurations more convenient.
- Adding scripts for automatic performance data analysis and presentation including plots pertaining to comparisons between different collections of test runs.

Sample Plots


Experiment ID
Total time (s)


G4ClassicalRK4::DumbStepper
G4MagErrorStepper::Stepper
G4SteppingManager::Stepping
inflate
G4SteppingManager::DefinePhysicalStepLength
G4Transportation::AlongStepGetPhysicalInteractionLength
G4TrackingManager::ProcessOneTrack
G4Transportation::PostStepDoIt
_ZN13G4ChordFinder13FindNextChordERK12G4FieldTrackdRS0_RddPdN5CLHEP10Hep3VectorEd
G4VoxelNavigation::ComputeStep
G4UniversalFluctuation::SampleFluctuations
leaf count