Performance of CMS Geant4 Simulation

V.Ivanchenko

17th Geant4 Workshop
10-14 September 2012
Chartres, France

Introduction

- CMS has integrated Geant4 9.5p01 in the developemt branch and faces visible slow down of simulation
- Urgent fixes are applied but CPU performance of 9.4 is not yet achieved
- This problem is general and is of concern for other LHC experiments and needs actions to improve the situation

CPU performance problem

- Significant CPU penalty have been identified after simple unit tests after 9.5p01 integration to CMSSW: depending on type of events up to 15%
  - Was confirmed by performance analysis of the FNAL group
- Analysis with igprof clearly identified a significant time spent in G4HadronicCrossSection class – GHEISHA cross section
  - The fix is in return back cash and to use G4Pow
  - The fix is in ref-07
- The private patch for CMSSW was provided using improved Geant4 classes included in ref-07
  - This reduces CPU penalty by factor 2
  - Still cross section methods are inside top list in the igprof report

Ttbar in Full CMS with Geant4 9.5p01

```
% total      Self  Function Before fix
   7.29    203.52  G4HadronCrossSections::CalcScatteringCrossSections(G4DynamicParticle const*, int, int) [38]
   4.81    134.17  G4CrossSectionDataStore::GetCrossSection(G4DynamicParticle const*, G4Element const*, G4Material con) [31]
   4.31    120.24  __ieee754_log [51]
   2.91     81.39  G4Mag_UsualEqRhs::EvaluateRhsGivenB(double const*, double const*, double*) const [72]
   2.13     59.59  G4Navigator::LocateGlobalPointAndSetup(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const*, bool, ) [46]
   1.85     51.57  __ieee754_exp [96]
   1.84     51.39  G4PolyconeSide::DistanceAway(CLHEP::Hep3Vector const&, bool, double&, double*) [85]
   1.84     51.30  _init [93]
   1.65     45.98  G4ClassicalRK4::DumbStepper(double const*, double const*, double, double*) [45]
   1.49     41.70  G4PhotoNuclearCrossSection::GetIsoCrossSection(G4DynamicParticle const*, int, int, G4Isotope const*, .. [104]
   1.47     40.97  __ieee754_atan2 [114]
   1.35     37.75  G4PhysicsVector::Value(double) [91]
```

```
 % total      Self  Function After fix of GHEISHA x-section
   5.46    348.91  G4CrossSectionDataStore::GetCrossSection(G4DynamicParticle const*, G4Element const*, G4Material *) [32]
   4.08    260.85  __ieee754_log [51]
   2.68    171.25  G4Mag_UsualEqRhs::EvaluateRhsGivenB(double const*, double const*, double*) const [73]
   2.57    164.13  G4HadronCrossSections::CalcScatteringCrossSections(G4DynamicParticle const*, int, int) [68]
   2.24    143.02  _init [80]
   2.06    131.57  G4Navigator::LocateGlobalPointAndSetup(CLHEP::Hep3Vector const&, CLHEP::Hep3Vector const*, bool) [45]
   1.84    117.50  __ieee754_exp [97]
   1.71    109.57  G4PolyconeSide::DistanceAway(CLHEP::Hep3Vector const&, bool, double&, double*) [85]
   1.60    102.28  __ieee754_atan2 [107]
   1.58    100.92  G4ClassicalRK4::DumbStepper(double const*, double const*, double, double*) [44]
   1.49     95.24  G4PhotoNuclearCrossSection::GetIsoCrossSection(G4DynamicParticle const*, int, int, G4Isotope const*,….) [105]
   1.47     94.02  G4CrossSectionDataStore::GetIsoCrossSection(G4DynamicParticle const*, int, int, G4Isotope const*….) [36]
```

Comments to IGPROF results for CMS

- At each step of a particle elastic and inelastic x-sections are computed
  - For GHEISHA (and some other) x-sections computation of elastic and inelastic are performed by call to the same private method CalcScatteringCrossSections
    - Usage of cash reduces number of such calls at least in 2 times
    - Usage of G4Pow reduce CPU required by this method
- After fix of GHEISHA x-section leading methods takes:
  - Geometry-navigation 14.6%
  - Hadronic cross sections take 12.8%
  - Math functions (EM, hadronics, geometry) 7.5%
  - EM physics takes 3.3%
  - Random generator 1.0 %

Recent FNAL Profiling Results

- Soon Yung Jun and Krzysztof Genser provided monthly report for ref-07
- About 5% CPU degradation in electron samples is observed
- Preliminary analysis of simple profiler results shows that extra methods appear in ref-07 in list of top CPU usage:
  - G4CrossSectionDataStore::GetCrossSection 2%
  - Electro-nuclear x-section 1.4%
  - G4ParticleChange::CheckIt 1.5%
- Further analysis is needed
- The Sunday report for ref-08 shows that the problem of EM CPU degradation disappears, what is the reason?

CPU Performance Problems for Geant4 9.5 and 9.6

- In CMS profiling cross sections take more time than in SimplifiedCalo
- After fix of GHEISHA x-section G4HadronCrossSectionDataStore cashes were added in ref-07, results are problematic but in ref-08 are suddenly improved
- Is ref-08 (current) situation final for 9.6 or we can do better?
- What else can be done for cross sections?
  - These classes are concentrated in one library, so all fixes are compact
- We need to understand which particle cross sections really take majority of CPU
  - Is it neutron and/or other cross sections?
  - Why gamma-nuclear and electro-nuclear take too much CPU?
  - Is it only hadronic problem or EM may compute cross sections faster alsoll?
- One possible improvement is to use G4Pow whenever it is possible
- This problem is essential for all LHC applications both for 9.5 and 9.6!