RECENT UPDATES IN THE GEANT4 HADRONIC FRAMEWORK

V. Ivanchenko CERN & Tomsk State University, Russia
18 March 2019
Motivation

G4DynamicParticle and G4Track

G4HadronicInteraction

String models

Destruction of physics at exit

Hadronic physics list components

What is not achieved

Plans for the June release
Motivation

- Hadronic framework was stable for the very long time
  - Some software solution looks not only obsolete but even ridiculous
- There are new elements of work plan for 2020
  - Hadronic interactions of b-, c- baryons and mesons
- Geant4 10,6 is expected to be used CMS for Run-3
  - Analysis of CPU performance of 10.6 shows that improvements introduced in 10.6 affect total CPU less than expected
    - We suspect, that some components of Geant4 kernel are not optimized
- Critical analysis of the framework is required
  - Beginning of the year is the best time
  - Git allows do things technically
G4DynamicParticle and G4Track

- This are container classes used in run time at each step of a particle in all interactions
  - In several discussions with Mihaly Novak and Gabriele Cosmo we identified that containers are used in not optimal way
  - There were “if” operators, thread_local variables

- What is done:
  - G4Track always has valid pointer to G4DynamicParticle, no check is needed
  - PDG code for optical photon “-22” – no check of the string for particle name is anymore required
  - Non-effective G4VelocityTable is not used anymore
  - New method G4DynamicParticle::GetBeta()
  - Removed majority of “if” from inline methods
G4HadronicInteraction

- This is the base class for hadronic models
  - Register/deregister mechanism allows keeping all pointers of all interactions for a given thread in G4HadronicInteractionRegister
  - BuildPhysicsTable(const G4ParticleDefinitio&) is called before run for all models

- Majority of components of the hadronic framework are now inherited from G4HadronicInteraction base class
  - developer should not delete these classes but left this for the G4HadronicInteractionRegister
String models

■ Small clean-up
  - Some useless methods are removed
  - Use public, protected and private consistently in the base and derived classes
  - Delete standard operators in the same way
  - Use C++11 keywords

■ Simplified instantiation of FTF and QGS model
  - Pre-compound is the default transport model

■ Builders now do not instantiate Lund fragmentation and Participants
  - G4ExcitedStringDecay by default uses G4LundStringFragmentation
Destruction of physics at exit

■ Why we need full destruction of physics at exit?
  - Users have trouble using debug tools like valgrind
  - Users may have problem in their application code when destruct Geant4
  - Developers have problem to identify memory leaks

■ The most important recent updates:
  - G4ParticleTable and G4ProcessTable classes are responsible for registration and destruction of particles and processes
  - Provided correct destruction of HP and AllHP models and cross sections

■ Recommendations for further developments
  - cross sections, models, and processes should be instantiated via pointers not be part of any other objects
  - no private destruction of these objects is allowed
  - we should not use G4THREAD_LOCAL data members in any new place both in hadronic classes and in Physics Lists constructors
Hadronic physics list components

- Thread local variables are removed from builders and physics constructors
  - hadron_inelastic, ions, stopping, decay

- Requirement to have a possibility defining of cross sections on top of any Physics Lists requires redesign of hadron inelastic constructors
  - We should subdivide process/models builders and cross section assigning
    - Cross sections better to taken out of builders?
What is not achieved

- We do not properly destruct processes of type
  - fTransportation
  - fParameterisation
  - fParallel
- Template pattern G4QGSModel<G4Participants> is not removed
- G4ThreadLocalSingleton does not delete itself and cross section factory does not delete itself
- I would propose to close the campaign of improving of the design
  - Only continue to measure current CPU performance
Plans for the June release

- Introduction of hadron elastic and inelastic interactions of b- and c- baryons and mesons
  - Will increase number of line of identical code if we will follow the current design of builders
  - Proposed class `G4HadParticles` with methods to access separately to hyperons, b-, c- particles, and stable particles

- Extend `G4HadParameters` with options enabling/disabling components of Physics Lists for b-, c- particles

- Implementation of a flexible choice of hadron elastic may be done in the style of EM adding extensions HADX, HADY, HADZ to the name of Physics Lists
  - Implemented in `G4PhysListFactory`

- Provide `G4PARTICLEXS3.0` with extensions planned for 2020