http://www.geant4.org

Thoughts on longer term strategy for G4 in ATLAS

My personal view

Andrea Dotti (adotti@slac.stanford.edu) ; SD/EPP/Computing

US ATLAS Physics Support, Software and Computing Technical Planning Meeting 2016

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Outlook

Current limitations to increase throughput and a list of todos

Longer term ideas

Experience so far

SLAC

Strong interest in multi-threading (event level) parallelism (via MT and MPI), including testing at NERSC on Knights Landing

Conclusions:

- Very good scaling and memory consumption well under control with O(100) threads
- KL much easier to use w.r.t. KC, x3 “single thread” speedup confirmed
- With very large worker counts -O(200k)- we see I/O blocking further scaling (preliminary tests done by Tom LeCompte @ MIRA)
  - Is this MIRA specific? I do not think so (tbc)
  - Even cout/cerr must be kept at barely minimum
  - **Need to address Geant4 *DBs* access** (currently based on lots of small files, lazily accessed by threads): will be discussed for Geant4 2017 work-plan

More short term possibilities

SLAC

In the (near) future we should address two Geant4 specific tasks to speedup ATLAS full-sim:

- Fast Hadronic CrossSection (almost there)
- Event Biasing, e.g. russian roulette

Both (plus DBs rethinking) are steps toward a more efficient ATLAS full-sim, but are not the final step

What is next? What are some of the ideas we are thinking of?

Geant4 Flexibility: positive aspect

SLAC

Geant4 design is very much appreciated for its extendibility, allowing to increase its usability writing your new classes

Examples:

- Use of G4 outside of HEP: medical, space, material science. DNA physics and chemistry module, SEE in semi-conductors, Crystalline structures
- Possible to replace a module with an alternative implementation: VecGeom from Geant-V being tested to become the standard geometry primitives library. Same with the Goudsmit-Saunderson MSC model

Geant4 Flexibility: negative aspect

SLAC

Geant4 OO design (‘90s): modules are tightly coupled together, difficult to isolate one component from the others.
The problem is seen by the packaging and library structure (e.g. libG4processes.so x10 larger than any other libG4*.so)

- Being addressed by a future G4 version

Examples:

- Physics Lists library requires all physics modules even if not used in application
- MSC requires *connection* with geometry navigation

Clearly an *issue* quite common in all our code-base

Two concrete examples

SLAC

Retrieve hadronic cross-section for interaction in ID

- Required using a dedicated application to setup and interrogate physics models

Use of final-state generation from G4 physics in fast-simulation

- Requires creating few instances of *fake* G4 objects (steps, tracks) to trick G4 kernel to perform a fake "event loop" just to force a single interaction

These has been addressed using in clever ways existing API, but this is clearly not simple nor easy

An alternative

Since it is very difficult to imagine a complete rewrite of Geant4 physics modules without a very large effort (see, e.g. https://goo.gl/xY7hQa , Ch8@https://arxiv.org/pdf/1603.00886v1.pdf)

Imagine GeantX (or for what matters any other simulation engine of your choice) in which:

- Physics domains (and geometry, navigation B-Field) are independent modules
  - **only lightly coupled**
  - design is not based on OO inheritance, but on how data are exchange (e.g. definition of their input or their output)
- Maybe it is not the *most optimized code* for a given technology, but I think can bring huge benefits
  - re-use the same physics module in different engines (Geant4, Geant-V, FastSimulation), **drastically reducing the development and validation requirements**
  - increasing code re-use with non-HEP domains (e.g. MCNP developers, medical community)
- It will be easier to implement specialized codes to address particular problems (very different paradigms that can mix together)
  - e.g. a GNN fast alternative library to hadronic interactions ("if primary==proton && energy<10GeV ; then UseGNN()")
- It will be easier to cherry pick in ISF what is needed, and interchange the Geant4 (Geant-V) engine with something else without redoing validation from scratch

It can be done

Example from outside our community:

- Scipy rich and powerful set of modules, what they have in common is that data are usually exchange as numpy.array not so much more

Example from inside our community:

- Again VecGeom library, well self-contained design that allows to be used in several products

Increase independence of modules

SLAC

At its core a radiation transport simulation physics algorithm is a *black box* that:

- accepts an input: a 4-momentum and a material {A,Z}
- returns a vector of 4-momenta

Possible work-plan:

- identify a critical physics component that is responsible for a substantial CPU fraction of ATLAS simulation
- develop a stand-alone phsyics library and well-defined interfaces towards the simulation engines of interest (Geant4 and Geant-V, ISF)
- SLAC/G4-team will use low-Energy neutron as a play ground for this idea
  - already in the work plan for 2017/2018
  - in strong collaboration with Geant-V: goal is a single library for both G4 and GV
  - we had successful experience w/ MPEXS: EM (and DNA) GPU code for medical physics, x200 faster then single-core G4. Missing interfaces (but back-ported to CPU as a *proof of principle*)
- Other physics modules can also be tried, butt the moment require adequate allocation of resources