Evolving Geant4 to cope with the new HEP computing challenges

A Gheata for the Geant4 Collaboration

4-8 Nov 2019, Adelaide
Context

• Increasing requirements for simulated samples towards HL-LHC
  • Full simulation still a bottleneck in many workflows
  • Speedup of several factors needed

• Potential of speedup in Geant4 on CPUs now better understood
  • Results from previous performance R&D studies
    • International R&D effort producing GeantV prototype and VecGeom
  • Restructuring and optimizations: more compact and streamlined code
  • Vectorizing some FP-intensive modules (e.g. field integration, multiple scattering)

• Accelerators and heterogenous computing become even more important
  • Making simulation code accelerator friendly becomes a necessity
  • Integration in experiment-driven parallel frameworks essential
Geant4 Task Force for R&Ds

• Created to promote & survey R&D on software architectural revisions
  • Including adaptations to emerging technologies and architectures beneficial to Geant4

• Provide communication/support among/for R&D activities

• Assess performance of different improvements and effort for integration
  • Make recommendations to the Geant4 SB

• Intended as catalyzer for short-cycle integration of performance developments
  • Coming from within and outside the collaboration
  • Regular meetings focused on ideas & follow-up of R&D activities
Performance: main directions

Parallelism
Concurrency model review - fine grain parallelism

Optimization
Faster physics/geometry algorithms - low level code optimizations

Restructuring
More compact code & data - simplified calling sequence - stateless - pipelines for heavy computation kernels

Heterogeneous computing
GPU friendly kernels

Experiments integration

Fast sim revisiting
Parameterizations - ML

A.Gheata - CHEP2019
Ongoing Investigations

- Code compactness, simplified calling sequence, optimizations
  - A large part of the GeantV speed-up coming from better fitting the instruction cache
  - More streamlined computation for HEP simulation hotspots
- More flexible parallelism model, accelerator friendly
  - Sub-event parallelism, task parallelism
  - Efficient track-level parallelism on warps
- Standardized and easy to use tools enabling fast simulation workflows
  - Parameterizations, but also generative models

Example of CPU μ-pipe for CMS EM shower simulation w/ Geant4

A.Gheata - CHEP2019
Vector pipelines in Geant4

Generalizing vectorization by passing vectors of data to functions rather than rely on inner loops.

• Idea originating from GeantV workflow, but generalized as templated API usable in any workflow
  • Using VecCore as vectorization library
• Prototyping the changes needed in Geant4 for such extension
  • Ongoing work for making Geant4 transport stateless
  • Aiming to prototype integration w/ FP-intensive modules
Optimizing Geant4 navigation using VecGeom

- A first implementation of a Geant4 navigation plugin, using VecGeom capabilities.
  - Allows to make use of the modular and extensible navigation acceleration structures of VecGeom
- Tests on full detector geometries remain to be done and development to be completed
  - Preliminary tests on simplified geometry very promising: 10-15% speedup vs. G4 native geom
- Related ongoing work: VecGeom navigation specialization for some volume topologies
Parallelism model, GPU exploration

Extension of Geant4 parallelism model to a task approach
GPU exploration work in general HEP simulation context
Task parallelism in Geant4

• Geant4 can benefit from having internally nested task-based parallelism
  • Making parallelism transparent to users (i.e no G4MTRunManager)
  • Better support for sub-event parallelism, eventually track-level parallelism

• Easier to expose simulation as a task
  • In relation with concurrent task based frameworks (e.g. CMSSW, Gaudi, …)

• First implementation of tasking support already available
  • gitlab.cern.ch/jmadsen/geant4-tasking
  • Based on standalone tasking library: github.com/jrmadsen/PTL
    • Native C++ features (future, promise, packaged_task, coroutines)
    • TBB backend available, PTL forwards task to TBB scheduler instead of internal

• Support for multiple task pools
  • E.g for off-loading work to coprocessors
Exploring GPU usage in full HEP simulation

• Geant Exascale Pilot Project – several collaborators from US
  • Goal to study and characterize architecture and performance to best use GPUs in general and Exascale facility in particular for full HEP simulation
    • Explore memory access, computation ordering, and CPU/GPU communication patterns
    • Avoid over-simplification
    • Reuse or leverage existing packages, not bound by backward compatibility
  • Strategies
    • Focus on NVidia compiler at first (later look at Kokkos and others)
    • Research way to increase instruction and data cache efficiency
  • Early technical ideas
    • Partial Static Polymorphism: allow upload/download of data to device without transformation
    • Separation Of State and Access and Functional Approach: allow significant data memory layout change without code change

• GPU-aware physics code restructuring being investigated
  • Kernels for EM shower physics “confined” to GPU, w/o user code calls
Fast simulation toolchains

Fast simulation techniques use approximations which typically trade accuracy by speed (e.g. shower libraries, parameterizations, sampling, ML)

Integration with Geant4-based full simulation workflows can be standardized
Automated tools for fast simulation

• Fast simulation is experiment dependent
  • custom procedures to extract parameterisations

• Ongoing work for streamlining the procedure
  ➢ Users come with their own setup
  ➢ Full simulation producing standard information (hits)
  ➢ Simplified, automatic and easy to use procedures to extract the parameters (e.g. fitting shower profiles, training networks)
  ➢ Plugged back in simulation via Geant4 fast sim hooks

• Goal: improve precision of fast simulation models

- Tools for extracting parameters of fast simulation models
- Tools for training and inference for generative models
Automating parameterisation for fast simulation

- Revisiting fast simulation hooks and models offered by Geant4 (e.g. GFlash).
- Work on automation of EM shower parametrisation - tuning of parameters for users’ geometry.
- More validation tools developed (for fast simulation, biasing - comparison to standard simulation)

![Graphs showing logarithmic plots](https://example.com/graphs.png)

- **green line** - GFlash parameters from arXiv:hep-ex/0001020
- **blue points** - from full simulation on Pb
- **red line** - fit to full simulation (e.g. no Z dependency)

\[
\langle \ln \alpha \rangle, \quad \sigma (\ln \mathcal{T})
\]

\[
dE(\vec{r}) = Ef(t)dtf(r)drf(\varphi)d\varphi
\]

A. Zaborowska (CERN)
Machine Learning Approaches

- Training on simulated data
  - Different calorimeter types: PbWO4, Pb/LAr, Pb/Sci, W/Scint

- Different architectures are tested
  - Autoregressive, VAE, GAN
  - Validation against MC truth

- Training/inference workflow aiming to integrate with Geant4 hooks
  - R&D done in EP-SFT & openlab (CERN)

https://openai.com/blog/generative-models/

I. Ifrim (CERN) talk and poster

A. Gheata - CHEP2019
Other optimizations and new features

• Magnetic field class refactoring
  • Runtime -> static polymorphism
  • Multi-particle driver integration

• Sub-event parallelism
  • Allow splitting events in vertices processed by separate threads
  • Better scaling and MT workload for large events

• Refactoring transportation
  • Splitting transportation process in “flavors” dealing separately with:
    • Charged/neutral particles, optical photons, parallel geometries or other specialized cases
  • Increasing code locality and decreasing branching
Conclusions

• Geant4 puts very high priority on performance related developments
  • Task force for R&D as catalyzer for development & integration
• Several performance improvement directions being followed
  • Structural changes & optimizations, but also revisiting fast sim workflows
• R&D on extending the parallelism model
  • More efficient integration with experiments parallel frameworks
  • Targeting also accelerators and HTC
• Aiming for accelerated prototyping/integration cycle