Introduction

High energy physics experiments such as the ones at the Large Hadron Collider (LHC) at CERN have been using so far most of their worldwide distributed CPU budget – in the range of half a million CPU-years equivalent – to simulate the transport through matter and the effects produced by particles generated in the initial collisions. These simulations are fundamental for understanding both the detector performance and the physics outcome of such an experiment.

The most computing-intensive components of such simulations are the geometry modeling, handling navigation in setups containing millions of objects, and physics, embedding state of the art knowledge of physics models.

Geometry redesign for vectorization

VecGeom is a complete geometry modeler evolved from legacy geometry libraries (Geant4, USolids, ROOT TGeo). It introduces a many-particle API besides the standard scalar one, and relies on templated based abstraction to enable both platform/architecture specific optimizations and vector/scalar API polymorphism.

The X-Ray benchmark: Can we harness the Phi for detector simulations?

Scalability of the basketizer behaves better using OMP balanced

• Approaches well the ideal curve up to native cores count
• Expected performance degradation as more threads are allocated

The balanced model converges towards the compact model as all the thread slots are filled

• It’s worth to run Xeon Phi saturated for our application

Rethinking particle transport to leverage vectorization

The scope of the project is the development of a community supported, open-source, next generation particle transport code for HEP (High Energy Physics) integrating both detailed and fast simulation physics models and transport algorithms, optimized for the emerging parallel and vector architectures.

- CERN/FNAL/BARC/Unicit-DACT joint project since 2013
- Two Intel®PCCs (CERN via openlab & UNESP)

• Group particles by locality into vectors (baskets)
• Invoke geometry to determine particle position
• Invoke physics models to predict stochastically a process location (interaction with detector material, decays, …)
• Validate proposed physics step against geometry
• Propagate vector of tracks and regroup baskets

Backends and interfaces

Long-term maintainability of the code implies writing one single version of each algorithm and specializing it for the different platforms/technologies using template programming and low level optimized libraries.

- A Xeon®Phi MCiev backend based on intrinsics is in production, inheriting from F64vec class, allowing also to run in offload mode
- A general vectorized backend is implemented using the Vc library (c-examples/uni-frankfurt.de/projects/vc)
- Backends exist for scalar, CUDA, CLX+, and Vc and can be extended to platform/library dependent implementations

Profiling with Intel Performance Tools

The performance tools were extensively used to understand the current performance of GeantV. Below is an illustration of the VTune outputs for the X-Ray benchmark done on a Xeon® Phi.

Offloading simulations on the KNC

We have tested the functionality of running GeantV tasks (scalar X-Ray benchmark) in offload mode, in a heterogeneous environment having one host and 2 Xeon® Phi cards. This was a preliminary performance measurement before enabling vectorization in our benchmark. The offload was split among the host and 2 Xeon Phi cards, demonstrating good scalability.

Good vectorization intensity, thread activity and core usage for the X-Ray basketized benchmark on a Xeon Phi (61 core COPRQ-7132 P)

We have compared the scalar Haswell performance for GeantV navigation in full CMS (one of the major LHC experiments) geometry. Left, real time for the simulation of 10 gp events at 77eV using the new VecGeom package instead of the existing ROOT geometry. Right side, the resident memory of the full application after completing the navigation states.

Realistic (basket) case: Fill baskets per geometry volume as particles are assembled (as in GeantV)