HEP-ASCR R&D Effort Towards Geant4 Re-engineering

Boyana Norris, Paul Ruth, Robert Fowler, Pedro Diniz, Robert Lucas, Makoto Asai, Andrea Dotti, Richard Mount, Philippe Canal, Daniel Elvira, Soon Yung Jun, James B. Kowalkowski, Marc F. Paterno, Robert M. Roser

Fermilab ANL renci RESEARCH ENGAGEMENT INNOVATION

U.S. DEPARTMENT OF ENERGY

SLAC NATIONAL ACCELERATOR LABORATORY

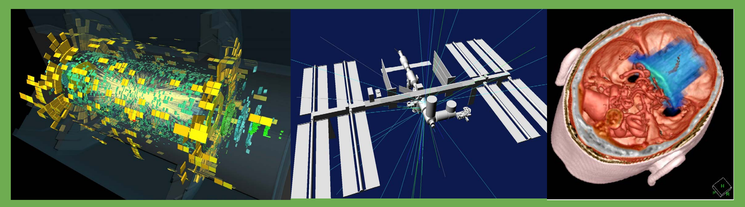
USC Viterbi School of Engineering University of Southern California

Slow path only:

Geant4

Geant4

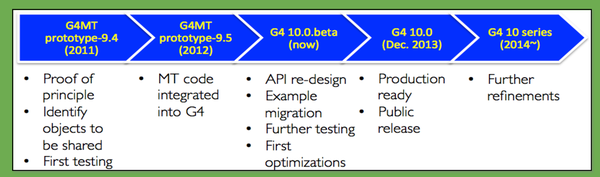
Geant4 is a toolkit for the simulation of particles passing through and interacting with matter. Its areas of application include high energy, nuclear and accelerator physics, as well as studies in space science, material science, medicine and biology, and also in security and industrial applications. In high energy physics, the Geant4 toolkit fulfills a critical need for the simulation of detectors at the LHC and at other existing and future experiments and facilities.

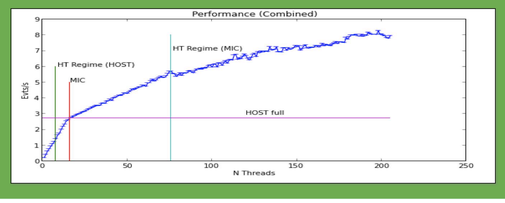


Geant4 is maintained and further developed by an international collaboration, which consists of more than 100 physicists and computer scientists. U.S. involvement in the development of Geant4 has been substantial since its early stages, and has increased with time. Several key Geant4 functionalities, including core framework, hadronic physics and visualization, are lead by the SLAC team, while major contributions in the key areas of hadronic physics and computing performance are made by the Fermilab team.

Geant4 version 10

High energy physics research as we know it today would not be possible without simulations. The massive production of event samples similar to those expected in the real experiment is an integral part of the process to design, build, operate highly complex accelerators and detectors and analyze the physics results. Thus Geant4-based simulation is currently the largest consumer of LHC compute cycles. In recent years, space and medicine have become significant user domains, with applications ranging from instrument and detector response verification to radiation dose and shielding optimization and analysis of biological effects. To address such an increasing demand of high statistic simulations, the Geant4 Collaboration released a new version of Geant4 (Geant4 version 10.0) in December 2013, which enables the use of multi-core CPUs and coprocessors in multi-threaded mode.

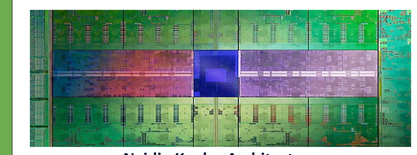




Preliminary scalability test of Geant4 version 10-pre-beta with full CMS detector geometry on 8-core Intel Xeon CPU and one Intel Xeon Phi coprocessor (60 core).
From left to right, CPU, CPU in hyper-thread, coprocessor, and coprocessor in hyper-thread.

HEP-ASCR R&D Effort

The process of re-engineering Geant4 has to be started and should target recently emerging new computing hardware such as many-core coprocessors and GPUs. Using these new architectures efficiently requires developing and leveraging massive parallelization, complex memory hierarchy, and deep vectorization capability.
FNAL, SLAC, UNC, USC and ANL joined forces to launch an R&D effort to investigate the possible evolution of the software infrastructure and numerical algorithms of the Geant4 toolkit to utilize these emerging technologies.



Nvidia Kepler Architecture

1) Performance analysis of the current Geant4 toolkit and its typical applications,
2) Reorganization of loops in the algorithms to make better use of vectorization
3) Prototyping GPU-based code
4) Studying automated code transformation for GPUs.

Investigations

CrossSection Optimizations

Surrogate Model for Common Cases
- **Fast path calculation for selected particles, material and physics processes**
- **Load predefined file (current) or generate on the fly (long-term goal)**
- **Fit unbiased bounded error curves (fixed and variable spacing)**

Particle/Material Pairs
- **50% of calls in ~10 particle/material pairs**
- **90% of calls in ~40 particle/material pairs**
- **Observed ~4k pairs**

Particle/Material/Process Triples
- **50% of cycles in ~10 triples**
- **90% of cycles in ~85 triples**
- **Observed ~18k triples**

Caching tens of pairs can speedup nearly all of the calls.

Example surrogate model for one triple:
Particle: Neutron
Material: materials_StainlessSteel
Process: G4Neutron InelasticXS

Slow path only:

	Cycles	Calls	Cycles/Call
Slow Path	6,133,110,476	6,278,517	977

Fast path with lazy computation of slow path:

	Cycles	Calls	Cycles/Call
Slow Path	223,362,860	94,876	2,354
Fast Path	1,059,541,332	5,887,001	179
Total	1,282,904,192	5,981,877	214

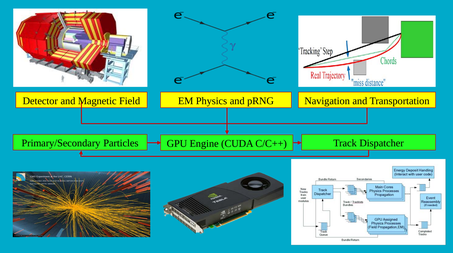
Potential ~5x speed up of all cross section calculation

Result Caching
- **Multiple calls to GetCrossSection with exactly the same particle, material, process, and energy**
- **Results in same cross section result**

Optimization (implemented):
- **Cache recent cross section for particle, material, process triple**
- **17% of calls would benefit from this cache**
- **29% of GetCrossSection cycles are from these calls**

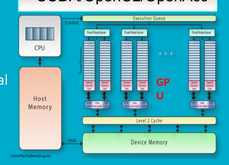
Measured 1.8% performance increase for real input

Prototype Running On GPU



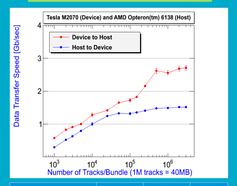
- ***Geant4 HEP applications***
  - event-level parallelism
  - highly sequential
  - memory intensive

CUDA/OpenCL/OpenAcc



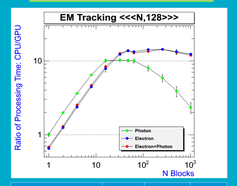
- **GPU-Vector**
  - track-level parallelism
  - vectorized track dispatcher
  - coalesced memory access
  - work balance (CPU-GPU)

Data Transfer



e⁻ <<<32,128>>>	CPU(ms)	GPU(ms)	Gain
Bremsstralung	2099	104	20
Ionization	558	25	22
Multiple Scattering	1034	185	6
Electron Kernel	751	61	12

Performance



γ <<<32,128>>>	CPU(ms)	GPU(ms)	Gain
Compton Scattering	51	6	8
Photo Electric Effect	70	6	12
Pair Production	50	10	5
Photon Kernel	71	7	10

Geant4 GPU applications/prototypes have been improved by:

•Dividing particle tracking calculations into small kernels and optimizing the kernels for GPU architecture. A physics process counter kernel was re-engineered to use parallel tree-wise reduction of counters, which produced

•On Fermi, up to 17x kernel speedup (633us -> 37us) with 2.1% whole-app execution time reduction

•On Kepler, up to 3x kernel speedup (84.88us -> 27.83us) with 0.17% whole-app execution time improvement

•Using GPU texture memory for RK4 electromagnetic field calculations for an improved capture of data locality of a particle moving through the magnetic field. This produced 1.038x kernel speedup (166.41ms -> 160.27ms) with 1% whole-app execution time reduction on Kepler.

Other performance optimization techniques are being pursued such as GPU thread grid optimization and further small kernel extraction and tuning.