GeantV Prototype
an update

Philippe Canal
for the GeantV project

F.Carminati (CERN), A.Gheata (CERN), S.Wenzel (CERN), M.Bandieramonte (Catania), J.de Fine Licht (CERN), R.Brun (CERN), A.Mohanty (BARC), A.Bhattacharyya (BARC), R.Seghal (BARC), Ph.Canal (FNAL), D.Elvira (FNAL), M.Novak (CERN), J.Apostolakis (CERN), G.Lima (FNAL), S.Jun (FNAL), O.Shadura (CERN), W.Pokorski (CERN), T.Nikitina (CERN), G.Cosmo (CERN), A.Ribon (CERN), G.Folger (CERN), G.Bitzes (openlab), G.Biro (Budapest U), L.Durhem (intel), H.Kim (Korea U)
Disclaimer

- Most of the material in these slides is from my colleagues
- I have added a few tweaks and all the mistakes
GeantV: motivations

(even if you are familiar with them)

- Performance of our code scales with clock cycle (hence is stagnant!)
- Needs will increase more than tenfold and the budget will be constant at best
- HEP code needs to exploit new architectures and to team with other disciplines to share the optimization effort
  - Data & instruction locality and vectorisation
- Portability, better physics and optimization will be the targets
- Simulation can lead the way to show how to exploit today's CPU's resources more effectively in complex applications
- Seeking ways to write code portable between CPU with vector units or not and accelerators (GPU, Xeon Phi)
With some simplification...
With some simplification...

Good Old Way
ETA: 3*Time/EV
With some simplification...

Brand New World
ETA: 3*Time/EV
With some simplification...

Brand New World
ETA: 3*Time/EV

Of course one should not forget cache effects, memory footprint and so on... still these are second order effects
With some simplification...

G4MT: CPU time for 100EV on 16 cores is the same as sequential.

RAM (reduced footprint) is reduced.

To reduce CPU time the flow of work (instructions, data) must change!

Brand New World
ETA: 3*Time/EV

Of course one should not forget cache effects, memory footprint and so on… still these are second order effects.
With some simplification...

Challenges

- Use CPU's resources to the max:
  - instruction, L1 & L2 caches, vector instructions, ILP
- Reuse instructions and data => deal with multiple tracks / events at a time
  - Current HEP code scores 0.8 IPC!
  - Fat CPUs can deliver > 4 results per clock (float & integer, ..)
- Create portable code!

G4MT: CPU time for 100EV on 16 cores is the same as sequential.

RAM (reduced footprint) is reduced.

To reduce CPU time the flow of work (instructions, data) must change!

Brand New World
ETA: 3*Time/EV

Of course one should not forget cache effects, memory footprint and so on... still these are second order effects.
What do we want to do?

- Develop an all-particle transport simulation software with
  - Improved state-of-the-art physics
  - A performance between 2 and 5 times greater than Geant4
  - Full simulation and various options for fast simulation
  - Portable on different architectures, including accelerators (GPUs and Xeon Phi’s)
  - Understand the limiting factors for a one-order-of-magnitude (10x) improvement
The initial ideas sounded easy
R&D directions

- Data structures, SOA types
- Concurrency libraries
- Steering code
- Base classes, interfaces
- Management and configuration
- Testing, benchmarking, development tools

- Locality by geometry or physics
- Dispatch efficient vectors
- Manage concurrency and resources
- Schedule transportation, user code, I/O based on queues
  - Optimize model parameters

- Next generation geometry modeling
- Template specialized algorithms
- Re-usability of inlined “codelets”
- CPU/GPU transparent support
- Support for vectorization

- Transforming existing G4 algorithms into “kernels”
- Support for vectorization
- Fast tabulated physics
- Template specializations
- Support for user fast simulation models

- GeantV kernel

- Scheduler

geant.web.cern.ch
The Scheduler

INPUT VECTORS OF PARTICLES

Generator
GeantV scheduler
Monitoring
Triggers, alarms

Particles

Geometry
Physics
Fast transport

ACTIONS: inject, Vector/single, prioritize, digitize, garbage collect
Logical volume trigger
Particle type, energy trigger
Geometry region, particle type, energy

WORK QUEUE

FastSim stepper
User defined param.
Fill output vector

GPU broker

VecGeom navigator
Simplified geometry
Full geometry

Vector stepper

Step sampling
Filter neutrals
(Field)
Propagator
Step limiter
reshuffle

TabXsec manager
Tab. final state samples
Tab. Xsec

Physics sampler
Phys. Process post-step
Secondaries
(Vector) physics
Compute final state

TO SCHEDULER

TO GPU broker

TO SCHEDULER

THREADS

WORK QUEUE
Scheduling features

- Generator
- Scheduler
- Stepper

1...N$_{volumes}$

1...N$_{workers}$

Automatic threshold-based basket dispatch

Transport queue

- Empty
- Full
- Transported

Reuse of baskets

AddTrack

Work balancing: FIFO
Prioritize events: LIFO

On-demand event flushing mechanism

Push on threshold

Recycle

Priority
Challenges for vectorisation

- Tracks have to be copied to a receiver during rescheduling
- During transport, tracks stop leaving holes in the container
Challenges for vectorisation

- Tracks have to be copied to a receiver during rescheduling
- During transport, tracks stop leaving holes in the container
Scalability for MT is challenging

- Performance is constantly monitored
- Jenkins module run daily
- Allows detecting and fixing bottlenecks
- Amdahl still high due to criticality of basket-to-queue dispatching operations

1000 events with 100 tracks each, measured on a 24-core dual socket E5-2695 v2 @ 2.40GHz (IVB).
Queues in GeantV

- Mutex based `dcqueue`
  - In production as work queue, provides priority
- Mutex/atomic hybrid `priority_atomic`
  - Mutexed only in high concurrency regime, provides priority
- Atomic CAS (compare and swap) `mpmc_atomic`
  - In production for basketiser queues, replacing `dcqueue`
  - Circular buffer, no priority
- Array lockfree `carray_lockfree` (ported by Omar)
  - Another implementation of circular buffer queue
- Boost lock free queue `boost_lockfree` (ported by Omar)
  - Boost implementation of lock free queue
Performance

- Our current dcqueue is outperformed by all the others on all platforms
- We currently work at ~10 transactions/sec
- Lockfree queues are doing great on Mac compared to mutex-based ones (50x factor!)
- priority_atomic is the only current replacement for dcqueue (must provide priority)
- We can expect a factor of 2 queueing improvement on x86_64 linux
- Reducing Amdahl requires revisiting the basketizing model
A lot of parameters!

- Keep $N_{\text{buff}}$ in memory (from $N_{\text{total}}$ to be simulated)
  - As an event gets flushed, inject a new one
- The vector size is a major parameter of the model
  - Small vectors = inefficient vectorization, dispatching becomes an overhead
  - Large vectors = larger overheads for scatter/gather, more garbage collections
- Automatic adjustment of vector size per volume
GeantV & optimisation

- Optimize GeantV scheduler model
  - Use genetic algorithms to find the optimum in the parameter space for different setups
- Model chromosomes:
  - Thresholds for prioritizing events, basket size, number of threads
  - Threshold for switching to single track mode, size of event buffer
- Fitness function: minimize simulation time while keeping in predefined memory limits
- TMVA analysis will come next
“VecGeom” in action

• Geant-Vector prototype can run complete first particle-detector simulations using VecGeom (or with ROOT/TGeo)
• measured a total simulation runtime improvement of 40% going from ROOT/TGeo to VecGeom for small example

ExN03 example

Pb
Scintillator

Caching & SIMD overheads

Gain larger than pain!
"VecGeom" in action

- Geant-Vector prototype can run complete first particle-detector simulations using VecGeom (or with ROOT/TGeo)

- Measured a total simulation runtime improvement of 40% going from ROOT/TGeo to VecGeom for small example

Preliminary performance checks

- Simple example imported from GEANT4
- Novice examples
- Scintillator + absorber calorimeter
- 30 MeV to 30 GeV electrons, 100K primaries
- Physics reproduced, small differences to be investigated for the highest energy
- No energy dependence of performance gain
- Extension to (simple version of) CMS geometry soon possible

Andrei Gheata, ACAT 2014

ExN03 example

Pb Scintillator

Caching & SIMD - overheads

Gain larger than Pain!

A speedup between 3 & 5 seems today really aim at one order of magnitude
Using GPU

• Can we use GPU just as we use worker threads on CPU now?
  • Initialise geometry, physics tables; same as on CPU - done
  • Pick-up CPU baskets, re-basketize GPU friendly – done
  • Asynchronous data transfer kernels and propagation kernels – partially done
  • Deliver back transported baskets – to do
• Prerequisites
  • Propagation code in GeantV to compile as kernel with NVCC — to do
  • Contiguous GeantTrack_v container to avoid gather/scatter towards GPU, refactoring non-POD navigation history
• CPU-GPU data exchange — starting now
  • Expecting issues in load balancing, latency, propagating action requests (e.g. garbage collection)
Other accelerators

- Xeon Phi
  - Keen interest from Intel
  - Some of the code already ported by Intel onto Phi
  - IPCC submitted, hope to get 2 FTE x 2 y
- AMD
  - Offer to pay a doctoral student
  - Identified a thesis director (Prof. D. Hill, Clermont-Ferrant university)
  - Looking for students
FastSim

- FCC studies are now being made with GEANT4 fast simulation option and ATLFAST parametrisations (Anna & Themis)
- As soon as this works, we will do the same with GeantV
- We may have our first customer in production!
GeantV/VecGeom Jenkins

- [https://geantbuild.cern.ch](https://geantbuild.cern.ch) available to geant-dev egroup
- SL5/SL6 (i686/x86_64), OS X (10.8/10.9), gcc4.8.1/4.9.1, clang and CUDA builds
- Nightly builds of GeantV and VecGeom with email plugin for notification users in e-group
- Code coverage
- Coding conventions (currently only Google, looking at clang-tidy)
- CTests with future CDash integration
- Code checks: Coverity, Cppcheck...
- Benchmarks/prototype checks (future possibility to store benchmarks in DB and build live plots)
Short term plans

• Run the prototype with the full (simplified) CMS geometry
• Develop simple event record for kinematics
• Target are x86 and NVIDIA
• Compare results with Geant4 FTP_BERT & Tabulated physics lists
• In the new year optimise the code and present the results at CHEP
2015 plans

• Study vectorisation of time consuming EM effects (Multiple scattering and ionisation)
• Port on Xeon Phi, ARM and AMD
  • Optimisation effort will depend on early results and vendors' help
• Introduce I/O for hits and study I/O parallelism
• Continue optimisation of Geometry, in particular with the introduction of fast voxelisation
• Develop *fast simulation* hooks / framework
• Perform optimisation of programme parameters
2015 plans

• Design & Install final development system

• Continuous integration, build and test system (jenkins, cdash, coverity…)

• Coding conventions (need to find the tool)

• Gitlab development model

• Validation & non-regression infrastructure
2015 plans

- Design of major subsystems
  - Electromagnetic
  - Hadronic
  - Scoring & Hits
  - I/O
  - Generator interface
  - Event model
Without forgetting…

- Documentation
- Coding conventions
- Some type and function naming which are confusing
- Support for AMD (OpenCL, sycl)
- Testing, testing, testing (standalone unit tests, shape stress tests, continuous integration)
- Extend benchmarks
- Continuous performance monitoring
- Issue tracking (bugs should be reported ... )
Longer term plans

- Development of improved, high-performance electromagnetic and hadronic packages
- Integrated fast / full simulation framework
- Low energy neutron integration
- Biased sampling
- …sky is the limit ;)

...
Synergies, synergies, synergies

- We are currently developing a \textit{vector signature} math library
- It would make sense to have it as a part of TMath
- Our Jenkins / git infrastructure could be merged with the one being developed for the group
- Work ongoing Oksana + Patricia
- VecGeom will be usable by GEANT4 soon behind the Usolid interface
- I think we should move one step beyond toward \texttt{ugeom}
- We have defined coding conventions, but we do not have a tool
Outlook

• Encouraging status
  • Expose parallelism, minimize contentions, real size application, stay architecture agnostic and portable
• Very large program of work
• Involvement of other communities would be very important
  • On the model of the collaboration FNAL - CERN
• Wish us good luck