GeantV – Coprocessors and Accelerators

Philippe Canal (FNAL) and Sofia Vallecorsa (CERN)
for the GeantV development team

Outline

- Introduction: motivation
- GeantV on GPU
- GeantV on Intel Xeon Phi
- Summary and plans

Motivation

- Modern HPC systems are being implemented as a mix of multi-cores CPUs and special purpose accelerators.
  - Largest share of world supercomputers use GPU.
  - Thiane-2 consists of 16,000 nodes: 2 Intel Xeon + 3 Xeon Phi (KNC) coprocessors
- Producers move towards hybrid systems
  - AMD APUs: processor and GPU are on the same chip
  - Intel just released the new Intel Xeon Phi (Knights Landing)

Motivation

- Exceptional raw power wrt simple CPUs
  - High energy efficiency
- Massively parallel architecture
- Substantial performance challenges for developers, including scalability, software tuning, and programming issues…

GeantV

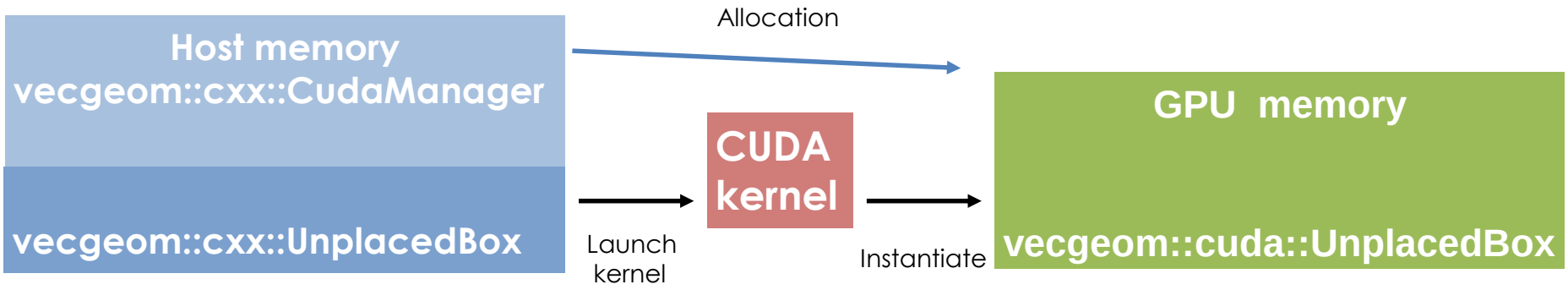
- easily portable to different architectures
- simple to maintain
- as little code duplication as possible

GeantV on GPUs

- VecGeom
- Physics
- Full GeantV transport

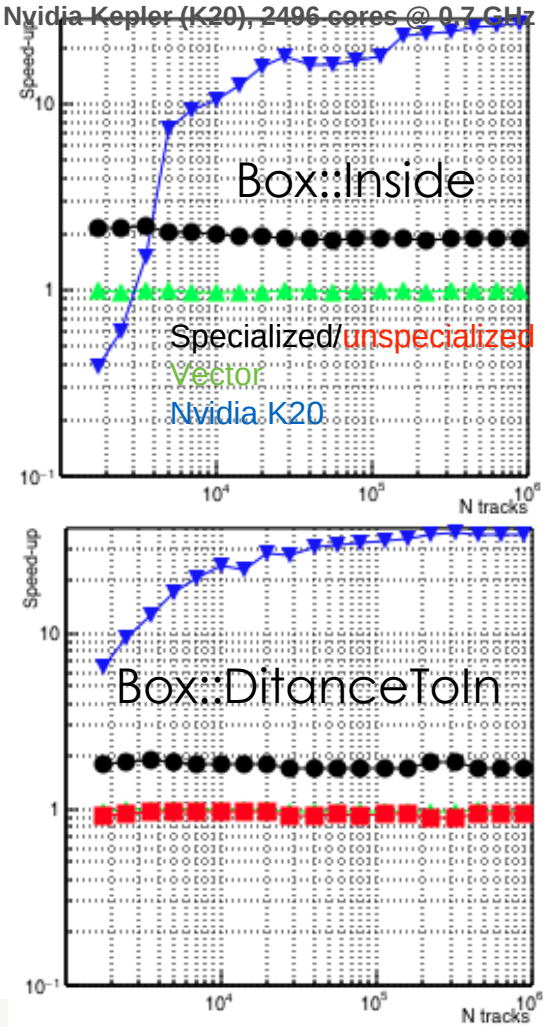
VecGeom on GPU

- Use same code for CPU and GPU (code abstraction & backends)
- Same classes live in 2 namespace: vecgeom::cxx and vecgeom::cuda
- Geometries are created in host memory and then synchronised to GPU memory



VecGeom Performance

- Asynchronous data transfer
- Providing constant throughput hides transfer latency
- Saturate device by running a single kernel on large containers or dynamically scheduling smaller containers.
- Measured kernel performance for BOX shape navigation methods
  - Scale with input size and number of floating point operations
  - Prove we can run successfully same CPU code
  - Working on optimisation



Tabulated physics on GPU

- Sampled interactions and tabulated cross sections as a quick tool for realistic showers
- Implement minor code changes
  - Replace stl containers and remove static members from __global__ or __device__ code
- Implement physics serialization to pass physics to device
  - On host: read physics in, stream to a buffer, copy to device
  - On device: rebuild physics tables and run kernels

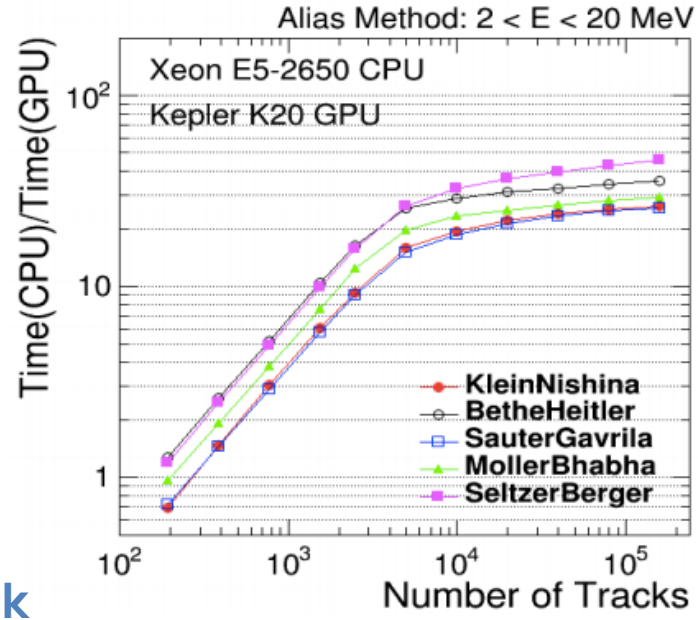
EM physics

- Vectorized EM physics kernels

PID	Process	Example Model	Secondaries
γ	Compton Scattering	Klein-Nishina	e^-
	Pair Production	Bethe-Heitler	e^-e^+
	Photoelectric Effect	Sauter-Gavrila	e^-
e^-	Ionization	Moller-Bhabha	e^-
	Bremsstrahlung	Seltzer-Berger	γ
	Multiple Scattering	Urban-WenzelVI	–
e^+	Annihilation	Heilter	$\gamma\gamma$

- Input energy range: E [2MeV:20MeV] with exp(-E) spectrum

K20 (2496 cores @ 0.7GHz, 26 blocks - 196 threads) + Xeon E5 (1 core, 2.6GHz)



Details in Soon's talk

GeantV transport: GPU schema

- Transport is managed by the CoprocessorBroker
  - Adapts baskets to the coprocessor
    - Gathers data in large enough chunks (e.g. 4096 tracks on K20) (at the cost of basket locality!)
  - Transfers data to and from coprocessor
  - Executes kernels
- Implicit vectorization: one thread per track
- Cost of data transfer is mitigated by overlapping kernel execution and data transfer
  - Split-up GPU's work and submit it through streams

GPU status

- Have working coprocessor broker with ‘geometry only’ kernel
- Tabulated physics code can be easily transferred to device.
- Vectorized EM physics kernel run on GPU

GPU plans

- Incorporate all physics code into CUDA Kernel
- Run the full prototype and understand performance issues
  - Latency, cost of data transfer
  - Basket size and composition
- Currently no customization of the algorithm for GPU
  - Room for kernel improvement and scheduling parameters tuning

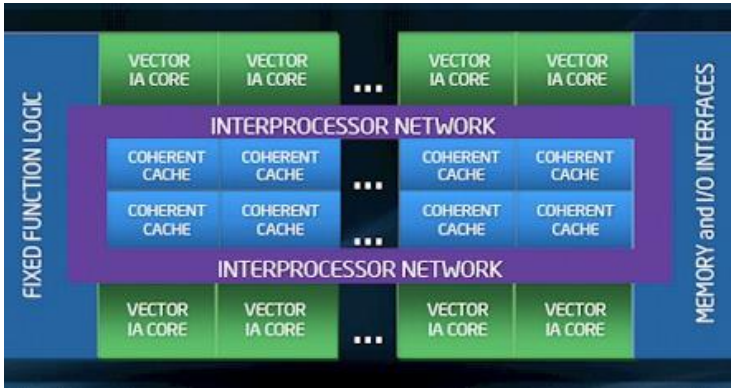
GeantV on Intel Xeon Phi

- VecGeom
- Full GeantV transport

First generation Intel Xeon Phi (KNC)

Many Integrated Core (MIC)

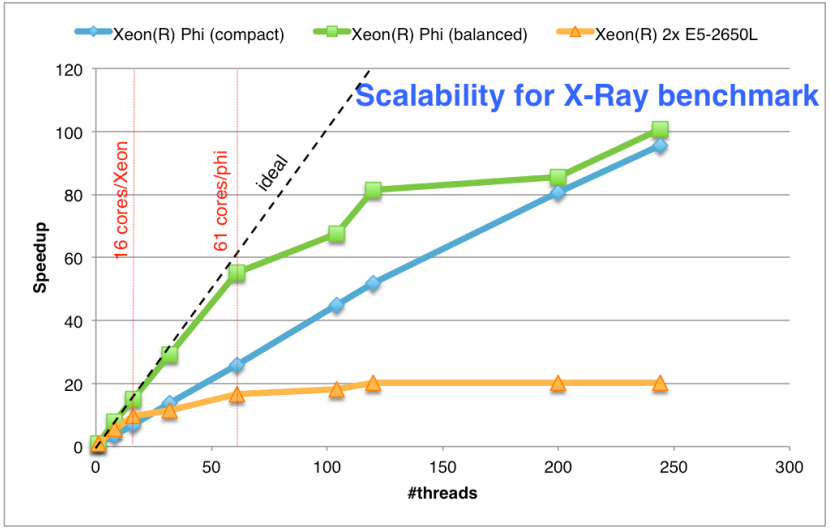
- Essentially a co-processor (stripped down Linux OS)
- 50+ “simple” 1.1 GHz cores in-order execution
  - 4 hardware threads cover latency issues
  - 512-bit registers give computing power
- Cache-coherency protocol



- 512k L2 cache per core (25MB on card), bidirectional ring network for L2
- Fast (GDDR5) memory on card

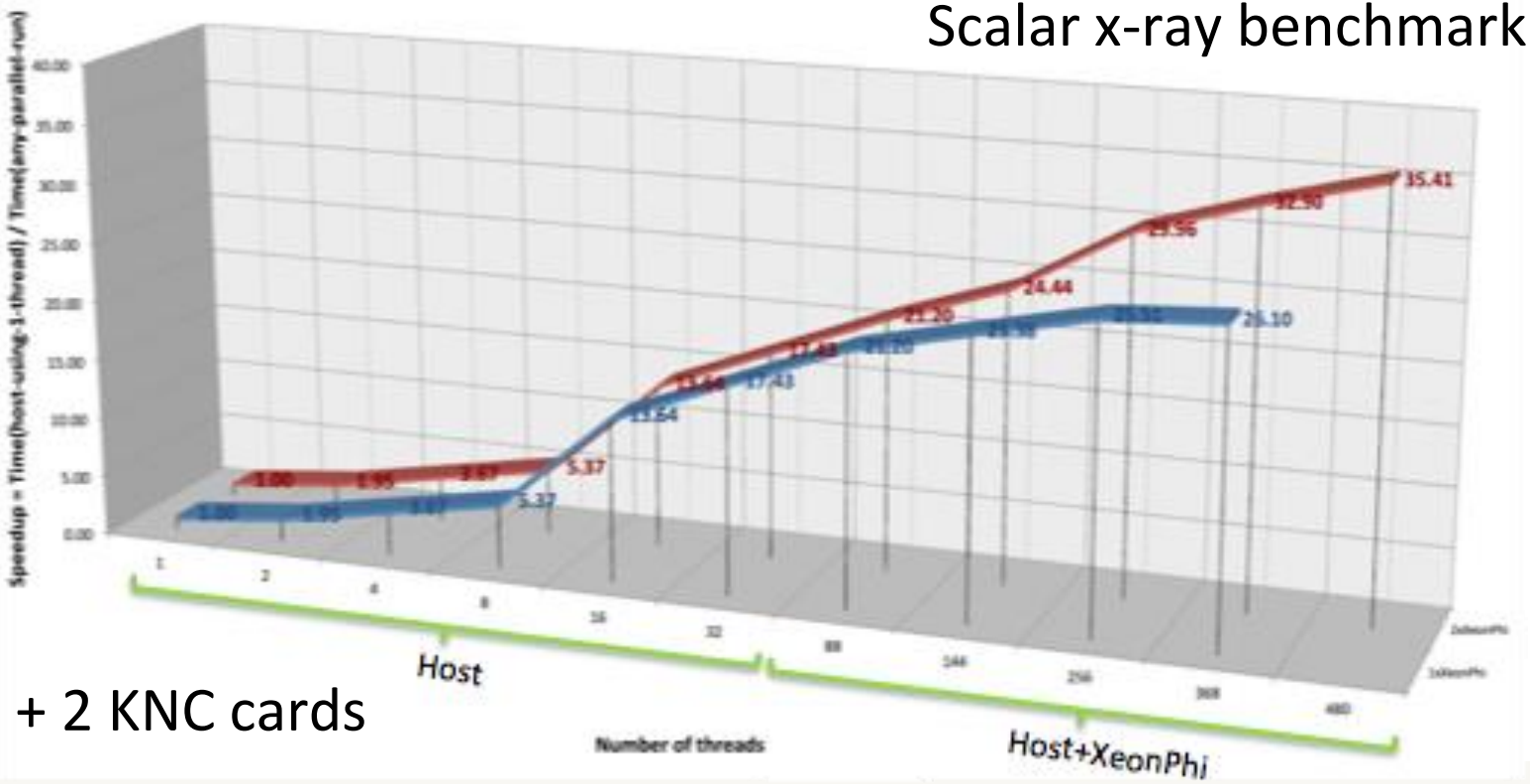
A powerful testbed

- To obtain good results on KNC need to saturate the cores and fill vector registers
- First detailed studies on scalability, memory bandwidth, vectorization and throughput for large number of threads
- First tests on basket strategy
- Run in native and offload mode



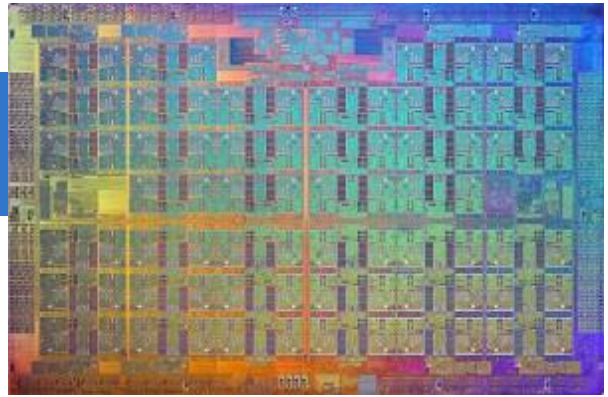
Offloading to KNC

Scalar x-ray benchmark



Intel Xeon + 2 KNC cards

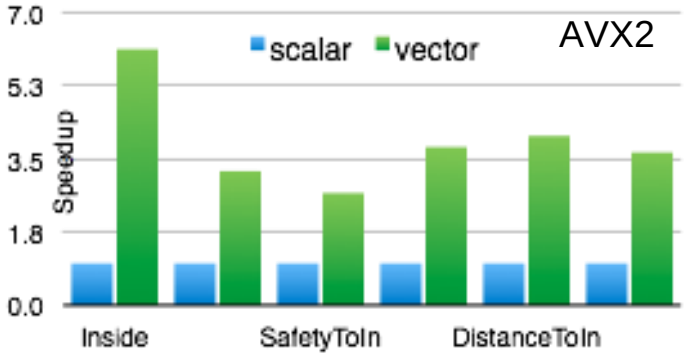
New Intel Xeon Phi (KNL)

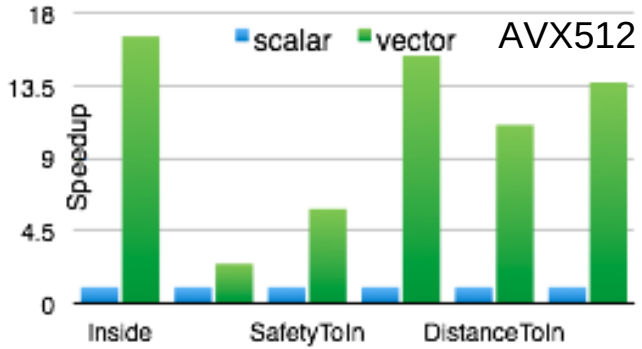


- Bootable host processor
- 60+ cores (2 VPUs)
- 3+ TFLOP/s DP, 6+ TFLOP/s SP (KNC: 1 TFLOP/s DP, 2 TFLOP/s SP)
- Direct access to <384 GiB DDR4 RAM (Up to 90 GB/s bandwidth)
- Up to 16 GiB MCDRAM (.5x DDR4 bandwidth)
- Binary compatible with Intel Xeon
- AVX512 instructions (including masks, gather/scatter)

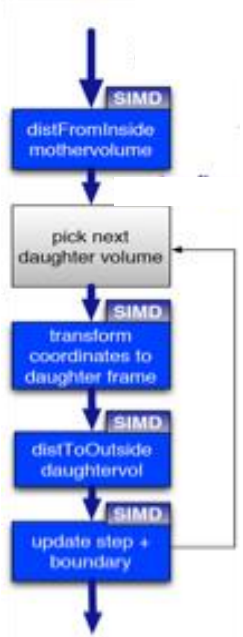
Geometry: shape benchmarks

- Test standard shape benchmarks
- Measure vector/scalar speed-up for AVX2 and AVX512 using UME::SIMD backend
- Super-linear speedup for some methods
  - Compiler and algorithms effects



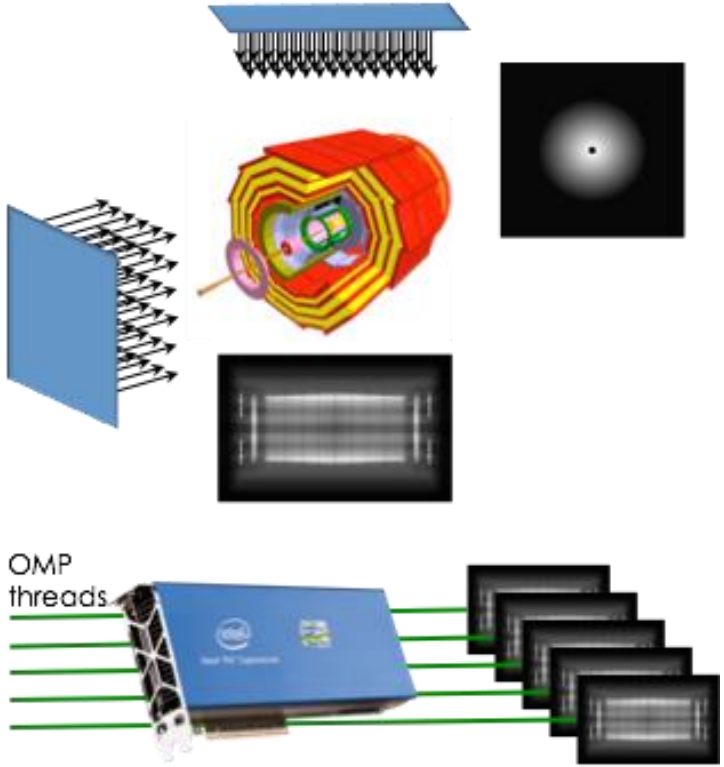


Intel® Xeon Phi™ CPU 7210 @ 1.30GHz, 64 cores



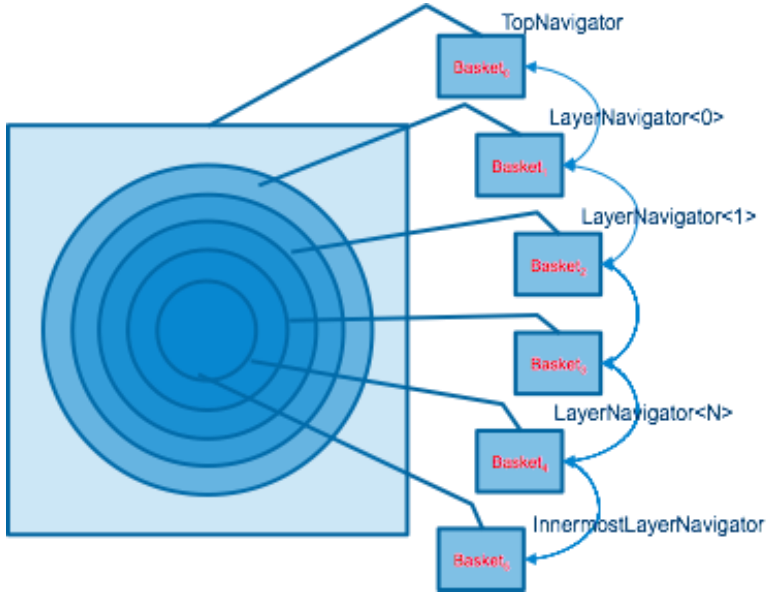
Geometry: navigation benchmark

- X-Ray scan of a simple toy detector geometry
  - Concentric set of tubes emulating a tracker
- Trace one ray per pixel and reconstruct the image
- Test the global navigation algorithm
- Stress vector API + basket transport tracing multiple identical tracks through the same grid
- Test OMP parallelism producing multiple identical images



Geometry: Navigation benchmark

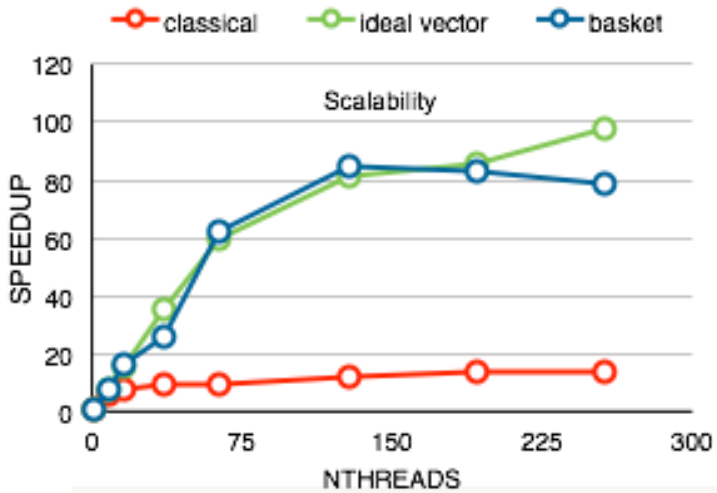
- Basket approach uses one specialized navigator per volume in vectorized mode
  - Crossing a layer “feeds” the next basket
  - Vectorization enforced by API, (UME::SIMD backend for AVX512)
- Geant4/TGeo approach uses a navigator per thread and works in scalar mode
- Compare also to an ideal “vector case” without any re-shuffling

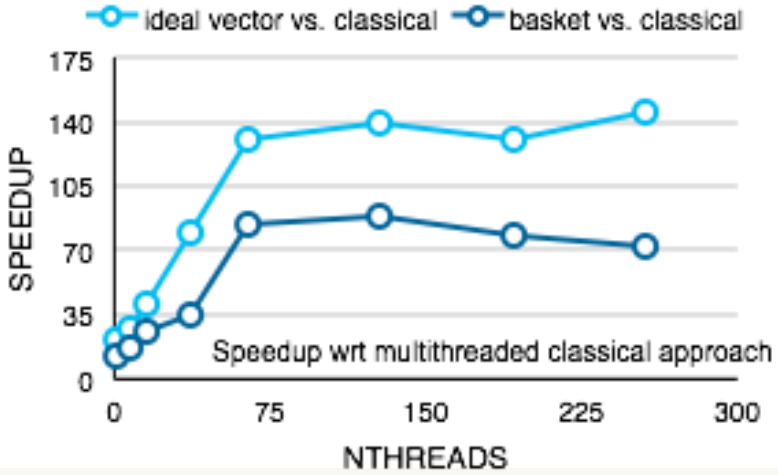


Scalability

- Scalability reaches ~100x for the ideal and basket versions
- Simplified case to test navigation approach

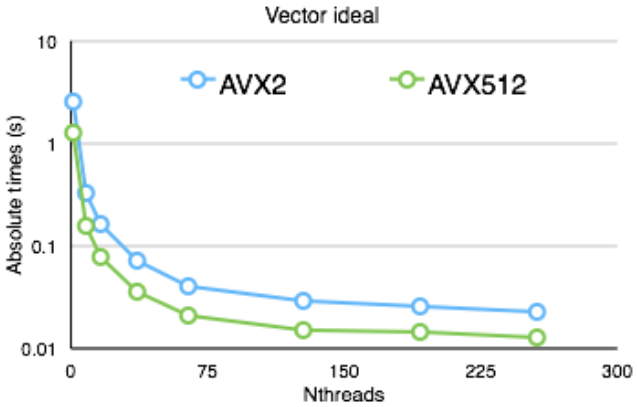
Intel® Xeon Phi™ CPU 7210 @ 1.30GHz, 64 cores

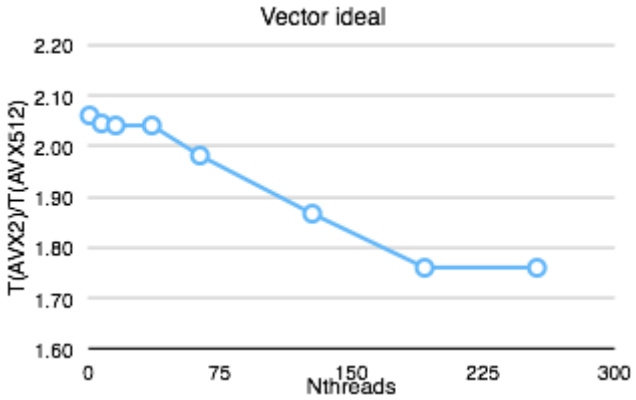




Vectorization

- High vectorization intensity achieved for both ideal and basketized cases
- Run AVX2 and AVX512 builds on KNL
  - AVX512 brings an extra factor of ~2 in speedup



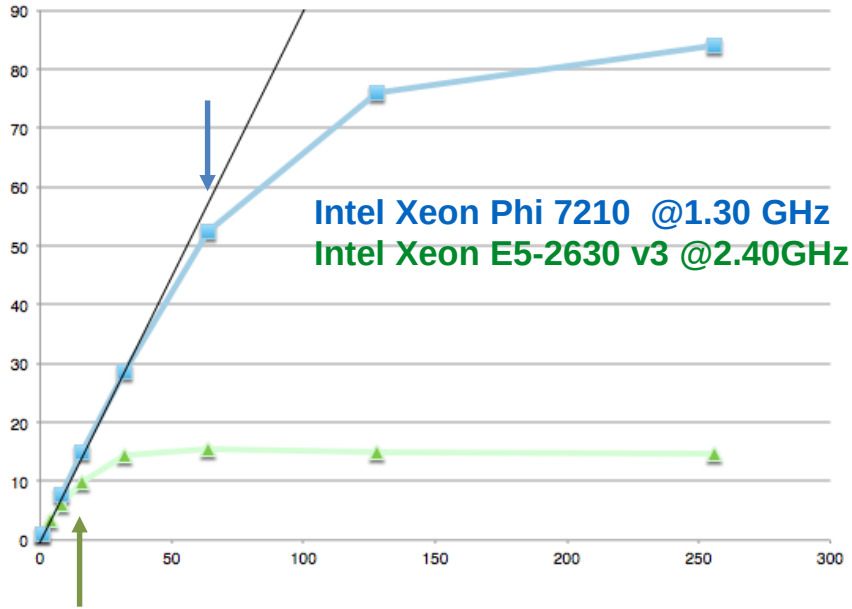


GeantV prototype

- First GeantV full transport benchmark on KNL
  - Tabulated physics
  - Simplified detector geometry
  - Test track transport and basketization procedure

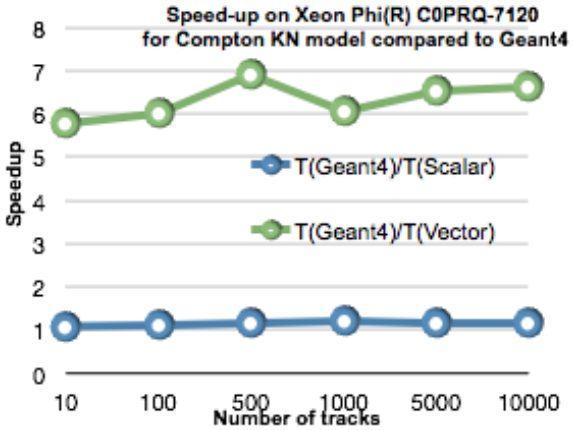
Good scalability up to the number of physical cores

Details in Andrei's talk

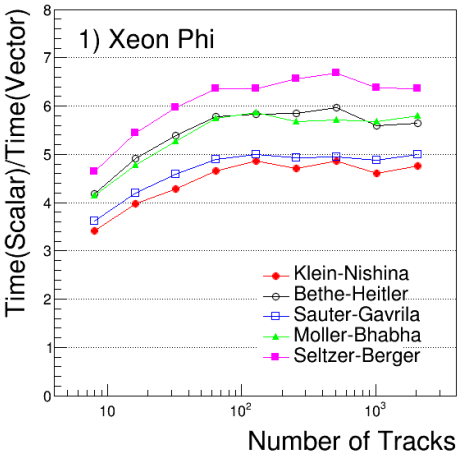


What about physics?

- Working on vector/accelerator friendly physics code
- Started with electromagnetic processes
  - Good performance gains in Compton scattering



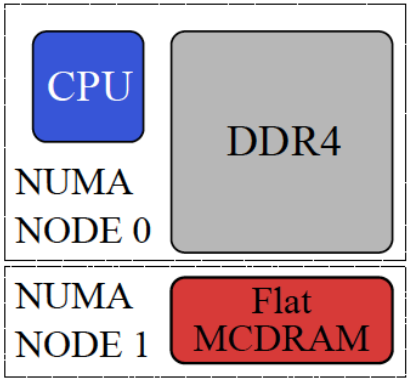
Intel Xeon Phi 5110P 60 cores @ 1.053 GHz



KNL memory modes

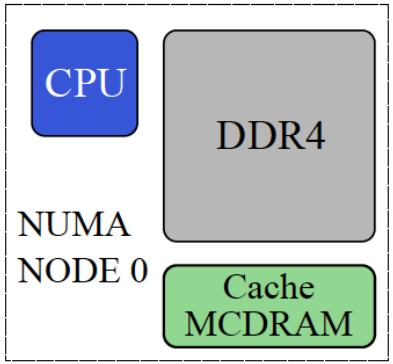
Flat mode

- MCDRAM treated as a NUMA node
- Users control what goes to MCDRAM



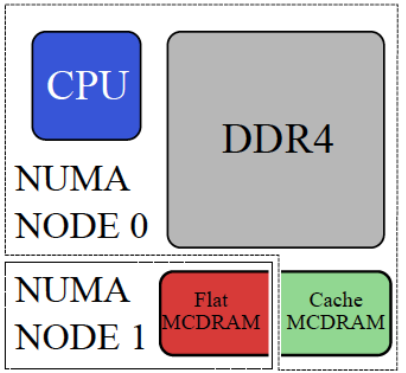
Cache mode

- MCDRAM treated as Last Level Cache
- MCDRAM is used automatically



Hybrid mode

- Combination of Flat and Cache
- Ratio chosen in BIOS

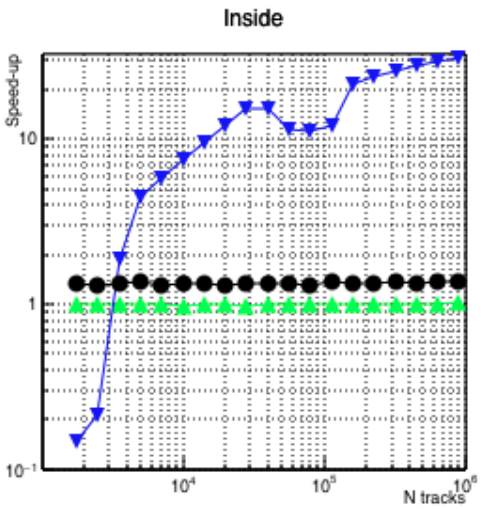


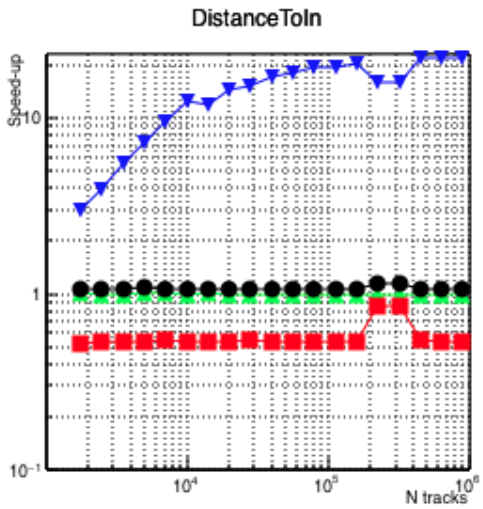
Summary & plans

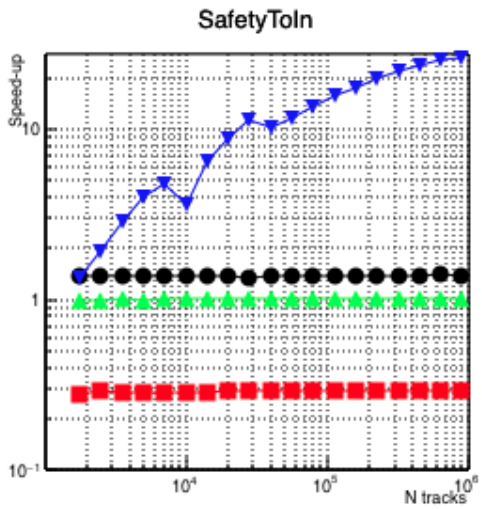
- Intel Xeon Phi is a powerful but complicated system
  - We have good results in terms of vectorisation and scalability
- Successfully run geometry on KNL
- Now optimising the whole prototype for a full realistic detector geometry benchmark
- Investigating effect of memory configuration

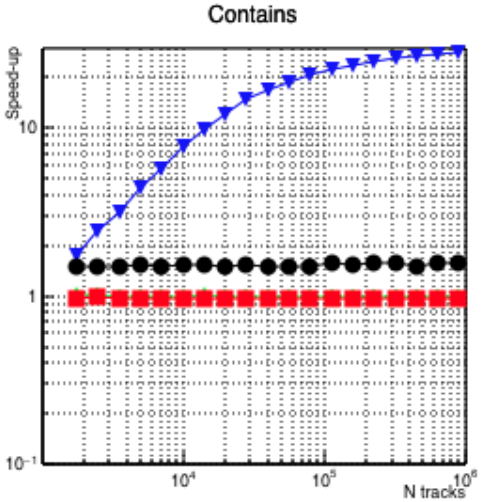
Thank you!

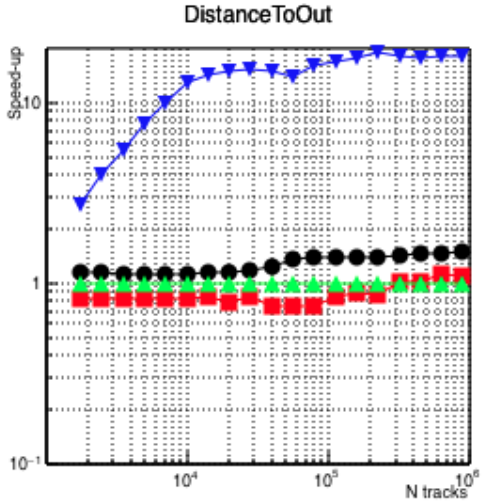
Cone:

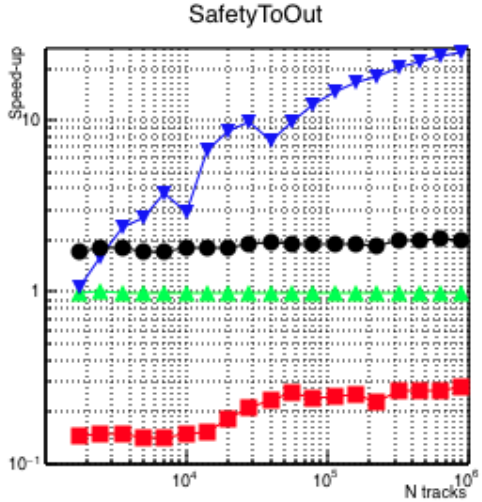

Inside
Speed-up
N tracks


DistanceToIn
Speed-up
N tracks

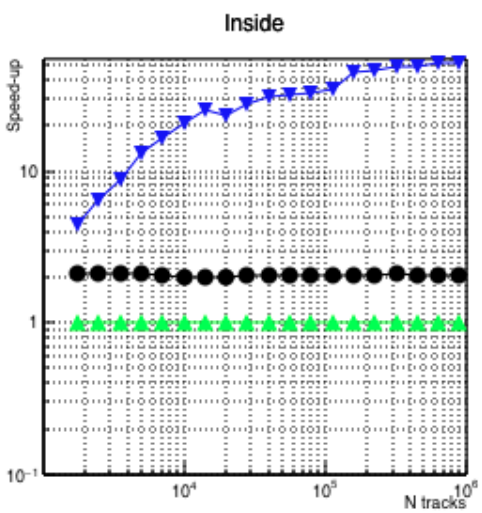

SafetyToIn
Speed-up
N tracks

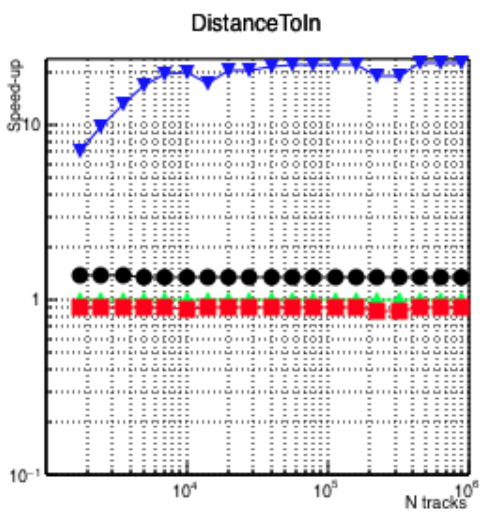

Contains
Speed-up
N tracks

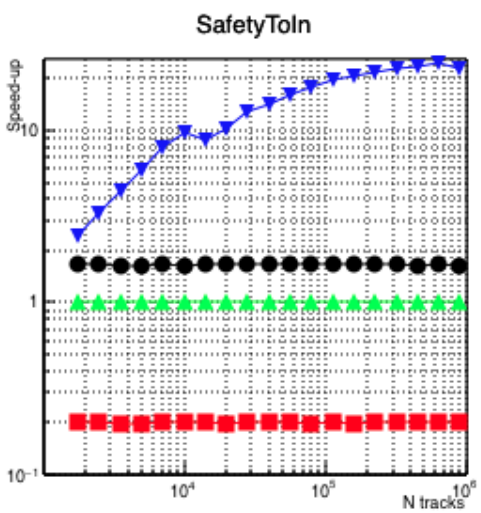

DistanceToOut
Speed-up
N tracks

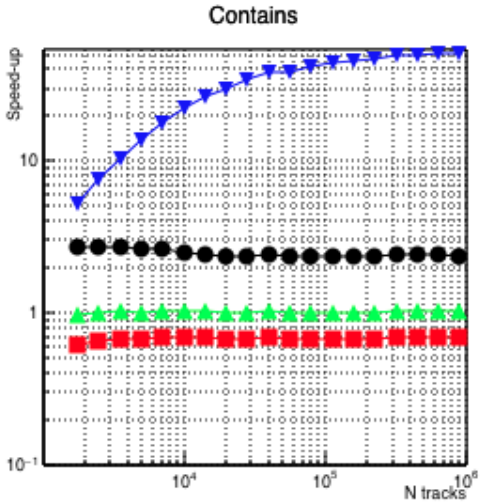

SafetyToOut
Speed-up
N tracks

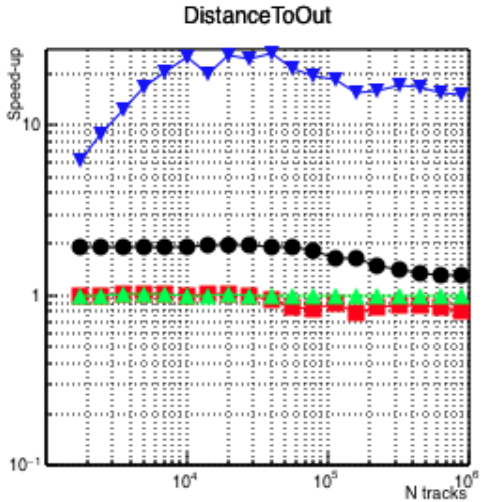
Trapezoid

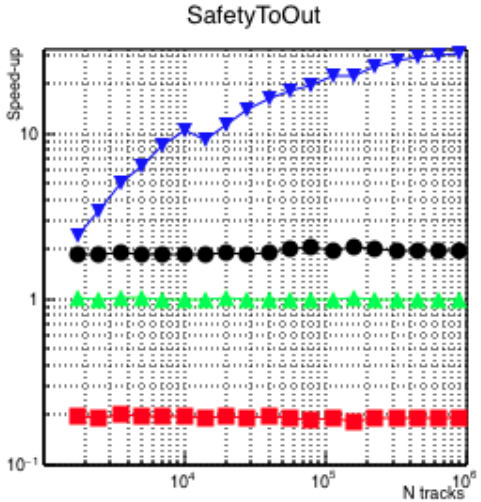

Inside
Speed-up
N tracks


DistanceToIn
Speed-up
N tracks

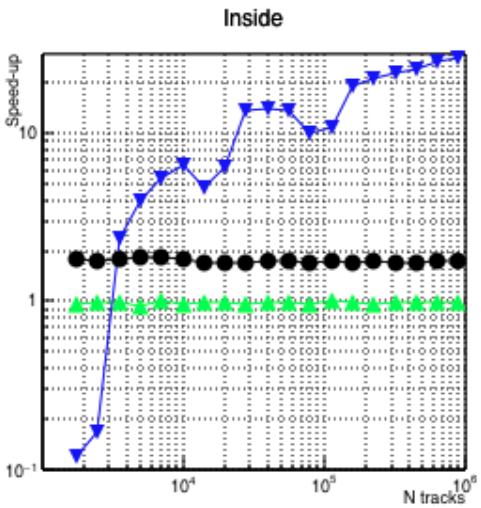

SafetyToIn
Speed-up
N tracks

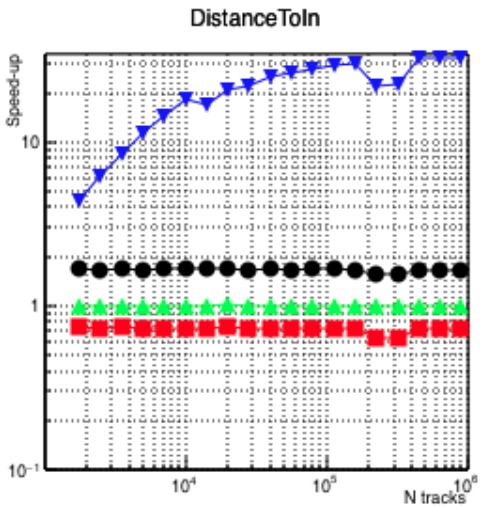

Contains
Speed-up
N tracks

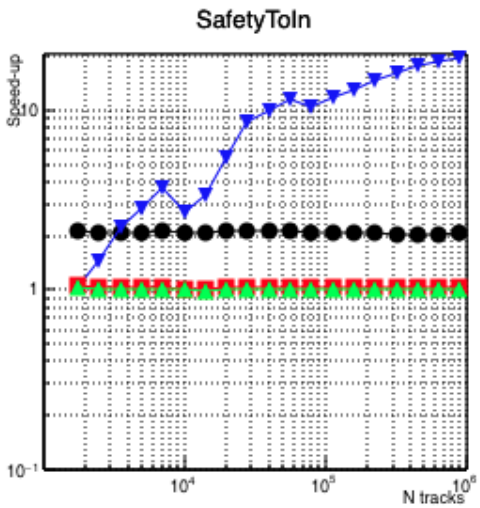

DistanceToOut
Speed-up
N tracks

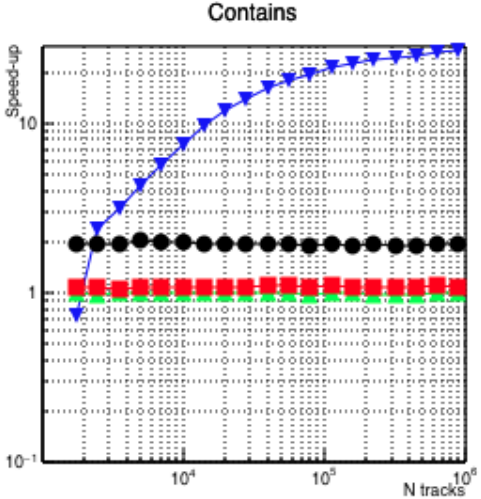

SafetyToOut
Speed-up
N tracks

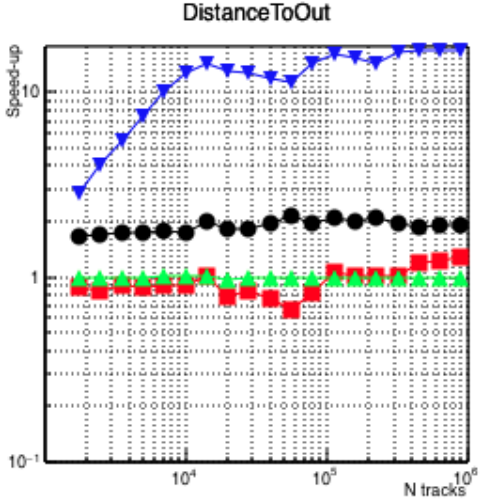
Tube

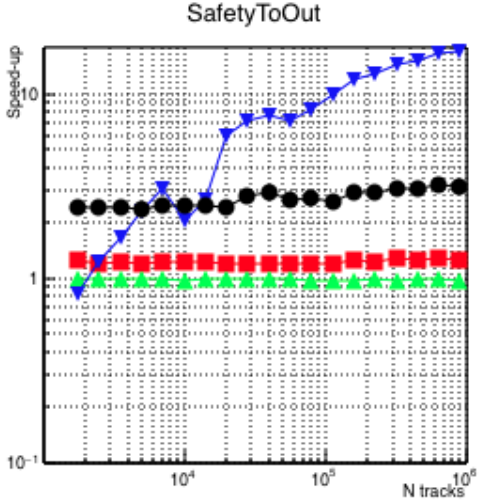

Inside
Speed-up
N tracks


DistanceToIn
Speed-up
N tracks


SafetyToIn
Speed-up
N tracks


Contains
Speed-up
N tracks


DistanceToOut
Speed-up
N tracks


SafetyToOut
Speed-up
N tracks