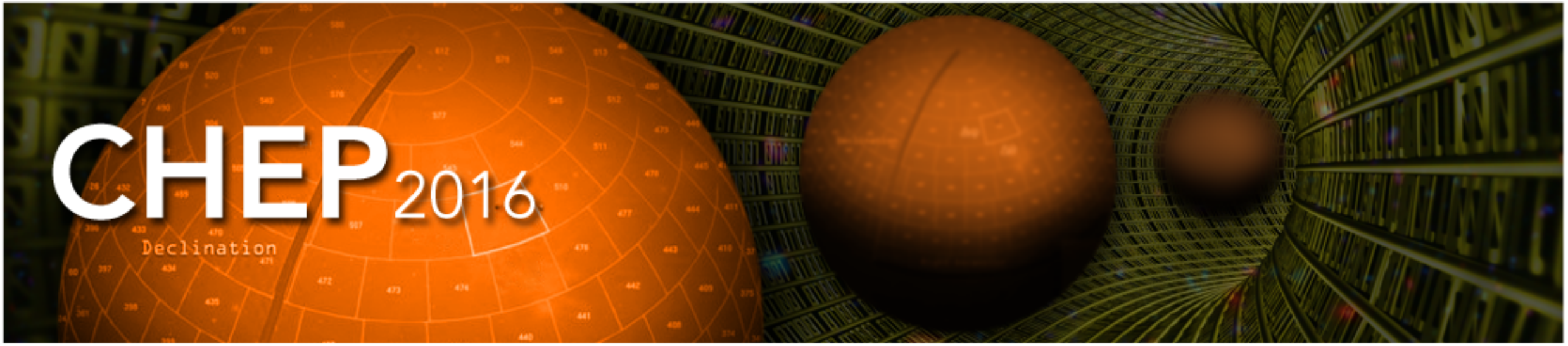


22nd International Conference on Computing in High Energy and Nuclear Physics, Hosted by SLAC and LBNL, Fall 2016

Performance of GeantV EM Physics Models

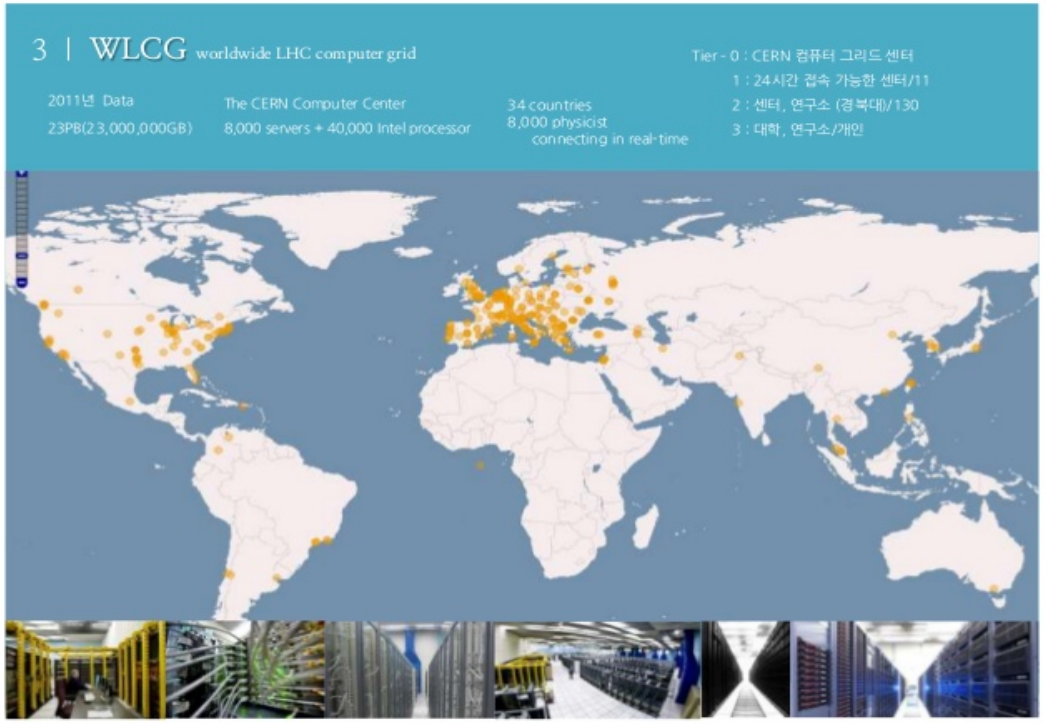
Soon Yung Jun (Fermilab)
for the GeantV Development Team

Oct. 11, 2016
CHEP16@San Francisco

Contents

Past – Death of Heroes

- Doom of vector (machines) in HEP more than two decades ago
  - Many HEP applications are memory-bound with many branches and poorly scalable
- Era of Pentium: Microprocessors ruled HEP computing
  - Experimental HEP programs have relied on high throughput computing (Grid)



Present – New Hardware Landscape

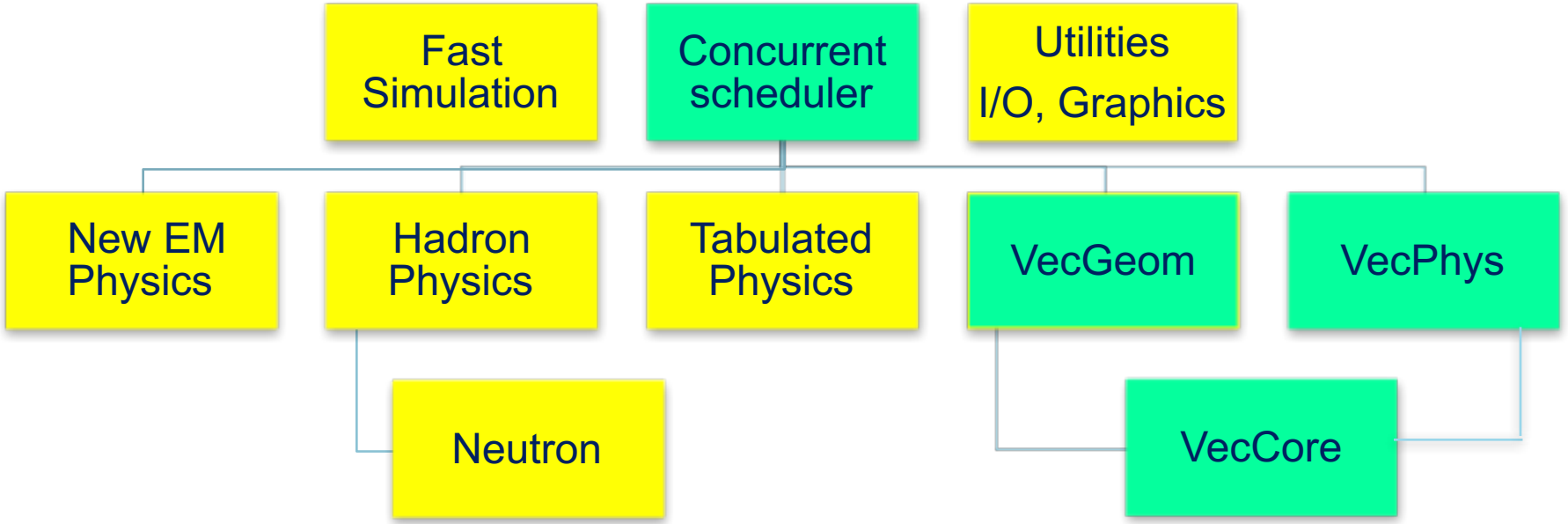
- Parallelism is omnipresent
- Where is HEP standing for utilizing vectors (SIMD), instruction level parallelism (ILP), hardware threading, and so on?

Commodity CPU	SIMD	ILP	Threading
MAX	8 DP	4+	4+
Typical HEP	1	~0.5	1

- Challenges for High-luminosity LHC and future experimental HEP programs
  - Ever-increasing demand for computing power, especially for simulation
  - Disruptive hardware changes
- Opportunities
  - New architectures (wider vector, many-integrated, massively-many cores)
  - High performance computing (HPC) + High throughput computing (HTC)

Future - Return of Vectors

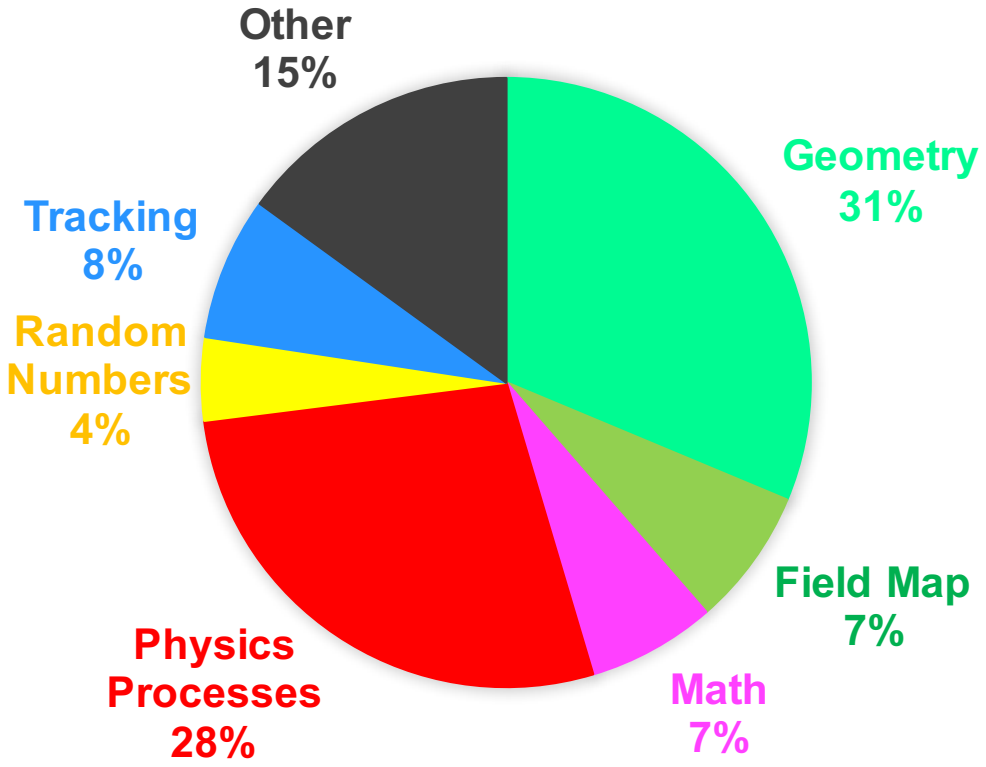
- GeantV: new demonstrator in HEP Detector Simulation
  - Track-level-parallelism to leverage vectors and threads
  - Locality and ILP (vector pipeline)
  - Portable codes for the variety of computing models

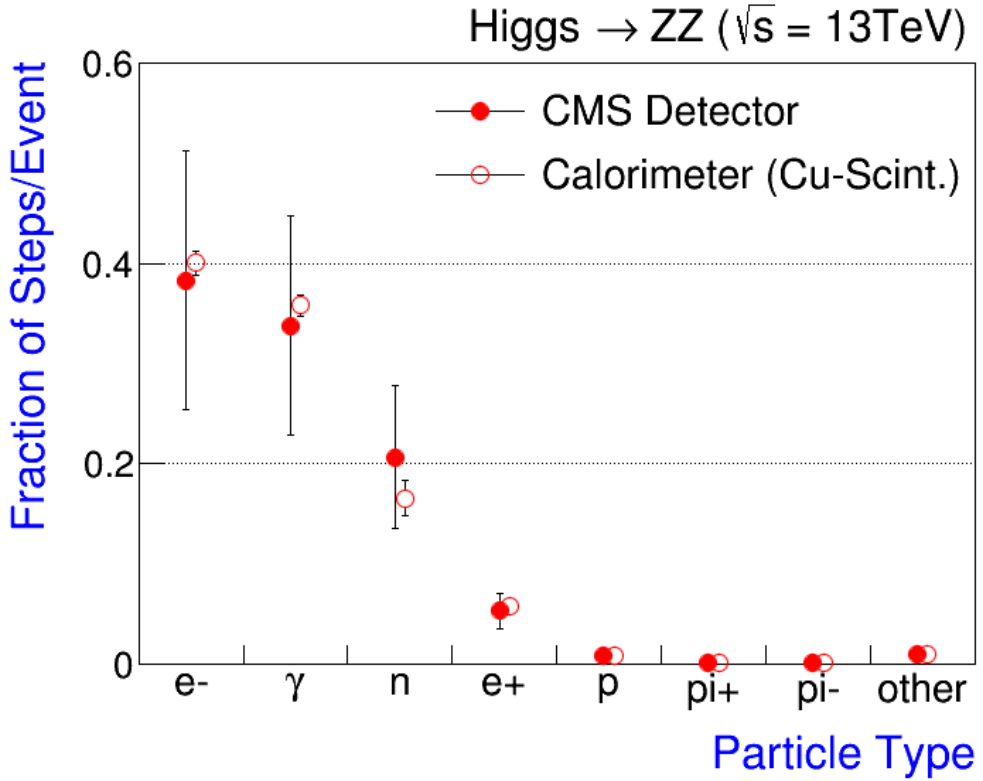


- GeantV EM physics: confluent-parallel paths
  - Develop new improved algorithms from the ground-up (New EM)
  - Vectorize EM physics models explicitly for SIMD/SIMT (VecPhys → this talk)

First Target - EM Physics

- A standalone CMS (geometry and a magnetic field map) simulation benchmark
  - pythia pp → H → ZZ (Z→ all decays) @ $\sqrt{s}$ = 14 TeV and Geant4 10.2
- CPU allocation for physics: ~ 40% (physics processes + pRNG + math library)
- $e^{\pm}$ and γ major consumers of physics (~80%) ➔ **~30%** of the total CPU time





EM Physics and Components of VecPhys

- EM shower: $\{e^{\pm}, \gamma\}$ cascades ← $\{\Sigma, \frac{d\sigma}{dx}\}$

  Σ ($d\sigma$): macroscopic (differential) cross section

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	$\gamma\gamma$

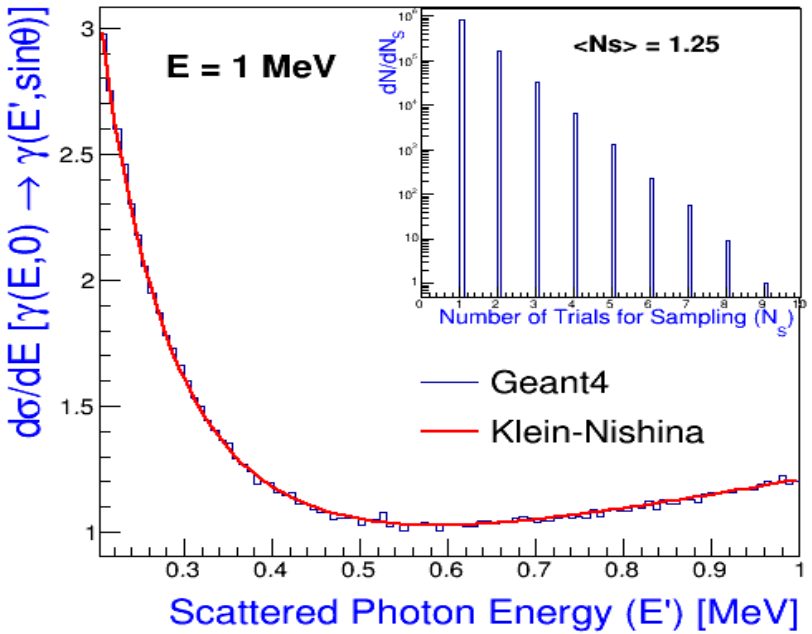
- 1) Determine distance before interaction

$$d = -\frac{\log(u)}{\Sigma} \qquad u(0,1) = \text{random numbers}$$

  - Calculation on-the-fly (vectorizable, costly)
  - Look-up cross-section table (**gather**)

- 2) Choose an interaction (process)
  - Based on relative weights of processes involved to Σ → conditional **random** decision
- 3) Simulate interaction (model): $d\sigma/dx$

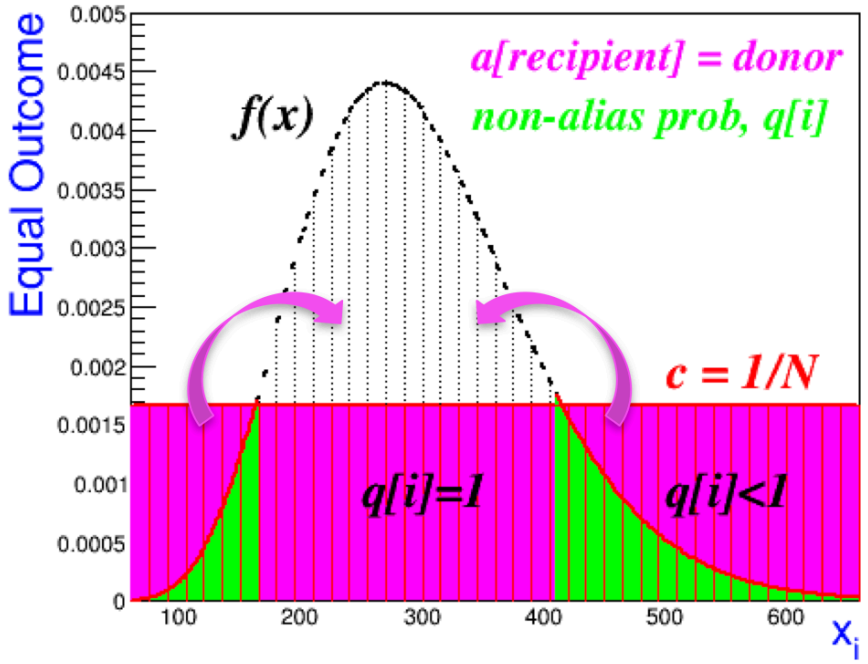
Compton Scattering: Composition and rejection

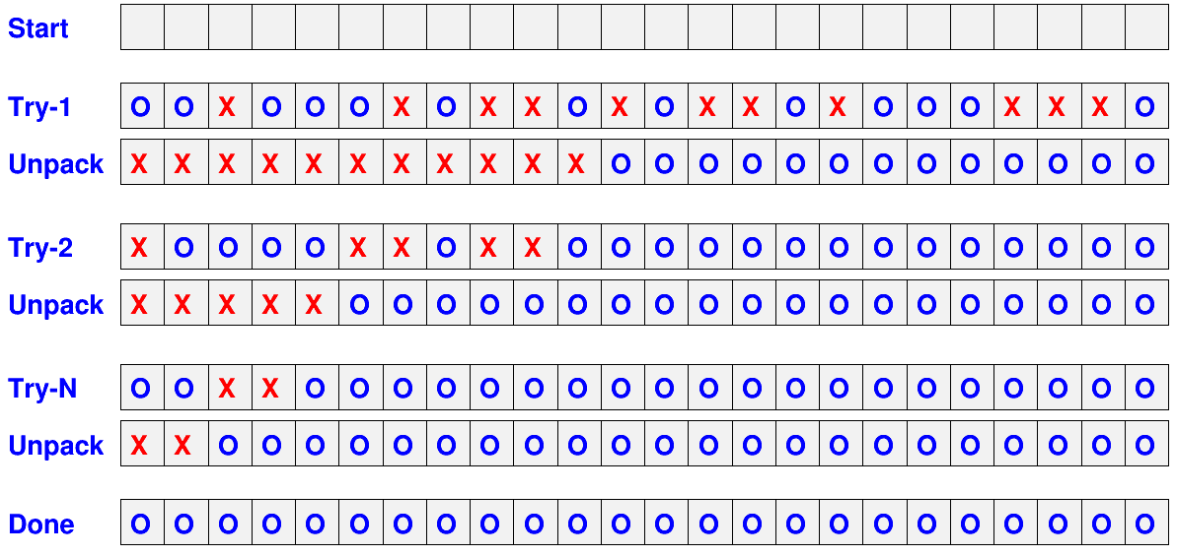


  - Involve conditional branches: not effectively vectorizable → **algorithmic change**

Example of Algorithm Consideration: Sampling Method

- Replace composition and rejection methods by an alternative sampling method
  1. Alias Method [1]
     - Recast a N-discrete p.d.f to N equal probable events, each with likelihood 1/N = c
     - Reproduce the original p.d.f by **one trial sampling** using alias and non-alias probability

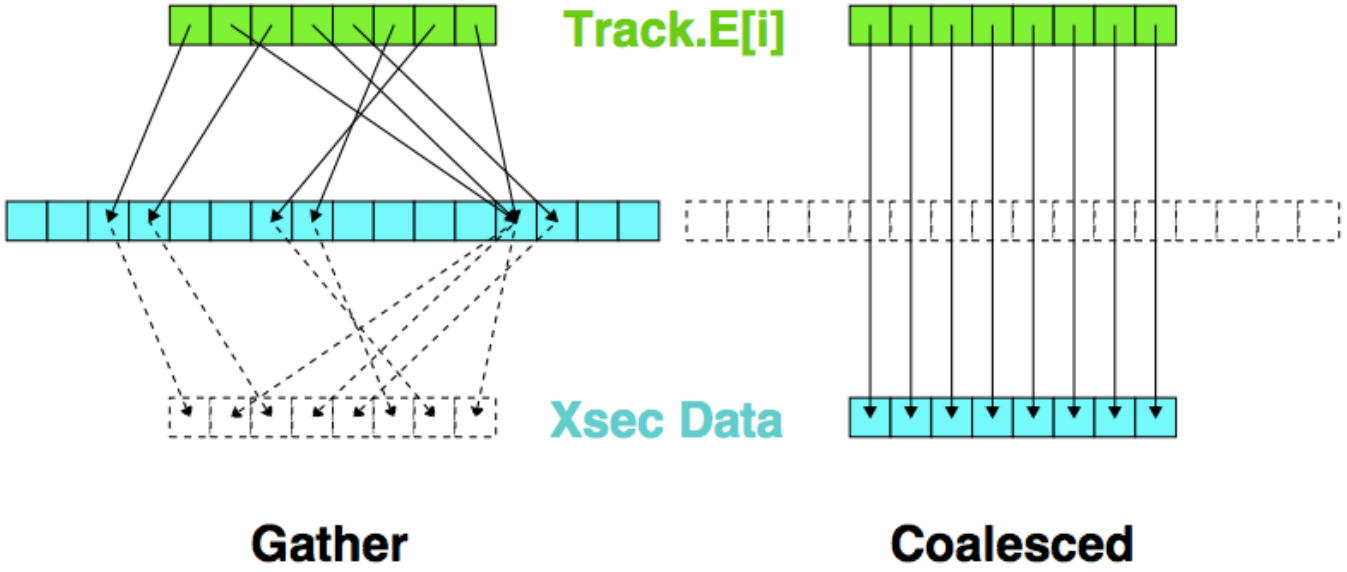


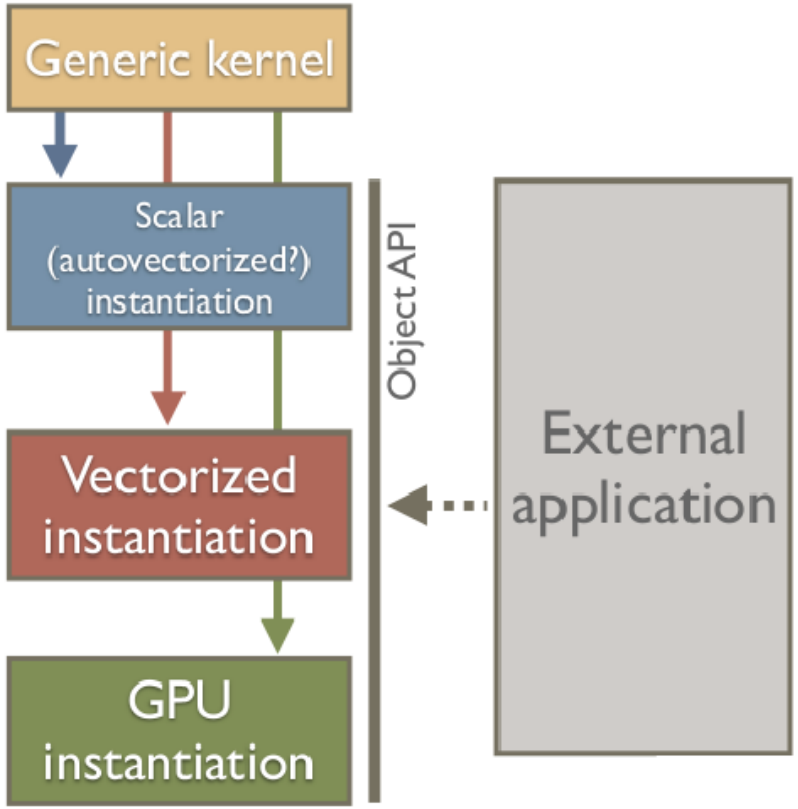


  2. Iterative shuffling: repeat **try and unpack** (overhead: reorganizing data)
  3. Combination of vector operations and scalar loops (overhead: Amdahl's)

Design and Performance consideration

- Implement algorithms based on generic and portable components for SIMD/SIMT
  - Template specialization and backend approach using VecCore [2]
  - Static polymorphism for interfaces
  - Data-centric (vector-friendly) code structure
- Other common techniques
  - Mask, gather/scatter, SoA data organization





Performance Measurement

- External vector libraries used (vector backend)
  - Vc: portable, zero-overhead SIMD library for C++ [3]
  - UME::SIMD: explicit vector library [4]
- Speedup for unit tests
  - SIMD = Time(Scalar)/Time(Vector)
  - SIMT = Time(Host)/Time(GPU) with Scalar
- Measurement:
  - Input energy range: E [2MeV:20MeV] with exp(-E) spectrum
  - Average time for n-repetitions as the number of tracks
  - Note that performance of Geant4 depends on the energy range

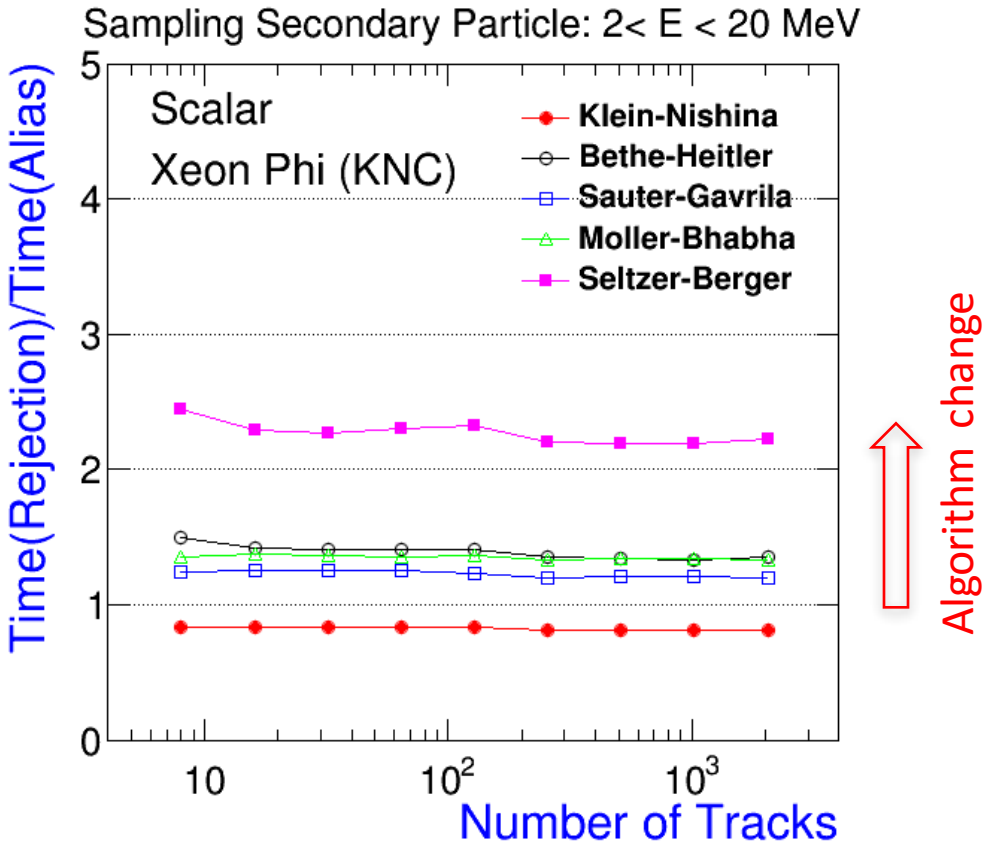
- Verification of simulation results: see the talk by M. Bandieramonte, "Validation of EM Physics Models for Parallel Computing Architectures in the GeantV project" in Session 2.7

Performance of EM Physics Models on Intel KNC

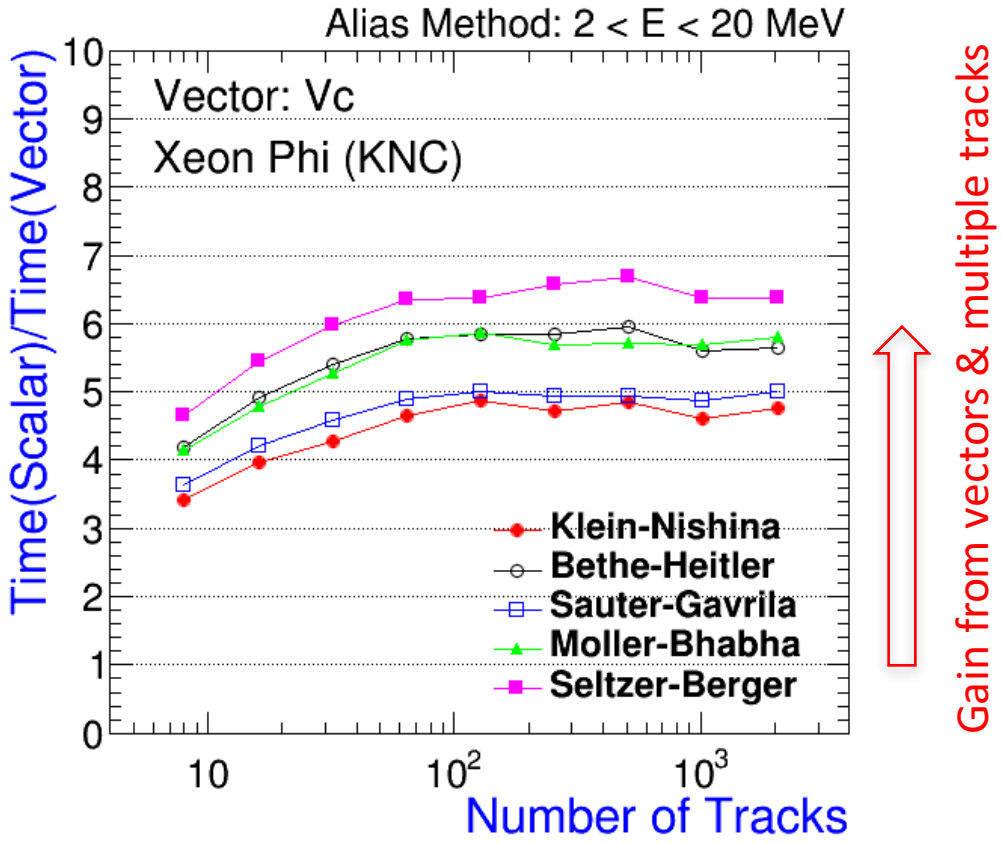
KNC (Xeon Phi 5110P 60 cores @1.013 GHz): **MIC** (**8 vectors** for double precision)

Simulation of interactions (sampling with alias methods)

- Scalar: Alias method



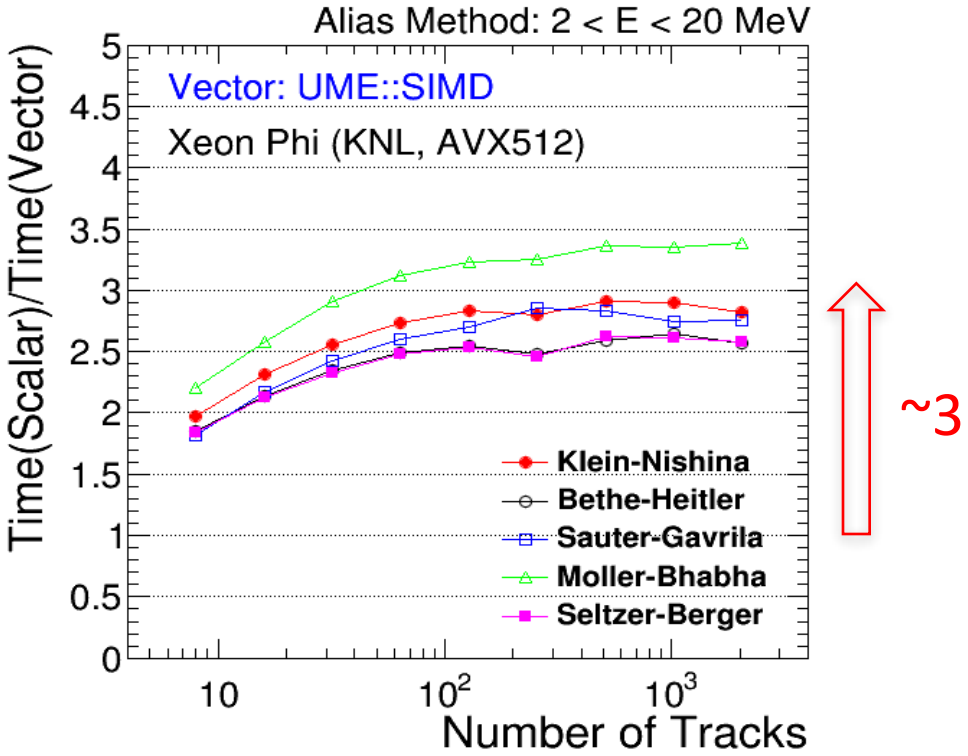
- Vector: Vc Backend + MIC



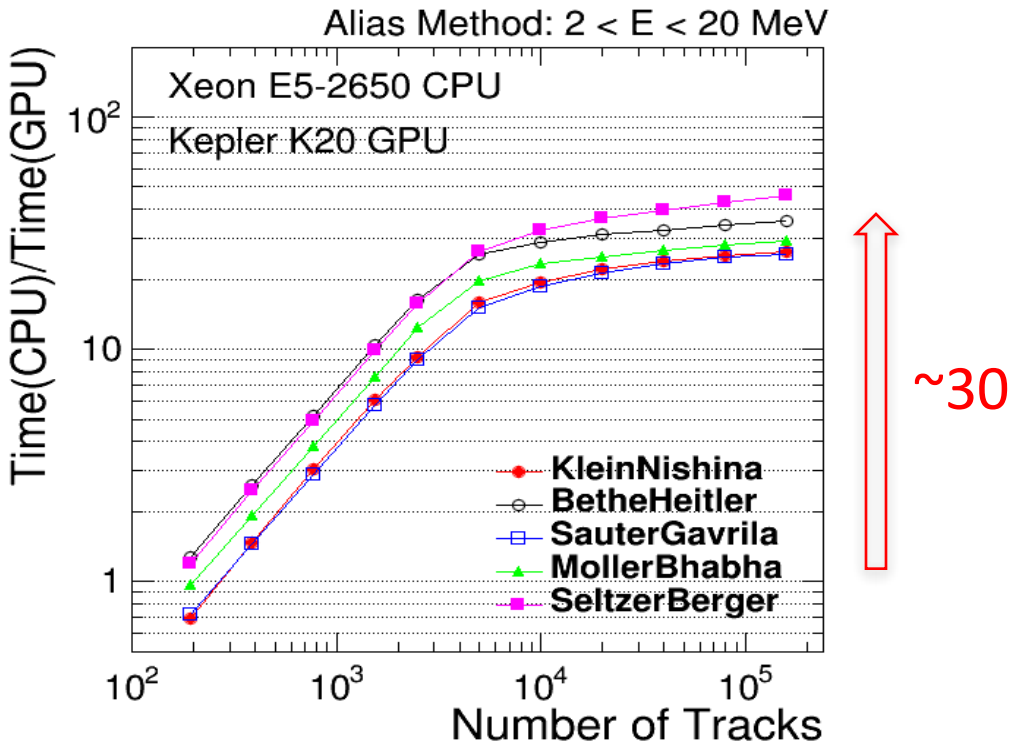
Performance of EM Physics Models on Intel KNL and NVidia Kepler (K20)

- KNL (Xeon Phi 7120 64 cores @1.3GHz): **AVX512** (**8 vectors** for double precision)
- K20 GPU (2496 cores @ 0.7GHz with blocks=26, threads) + Xeon E5 (1 cores, 2.6GHz)

- Vector: UME::SIMD backend+ AVX512



- CUDA: Scalar backend



- Note: performance of UME::SIMD is underestimated as a sequential pRNG is used for the backend
- Maximum potential speedup(1-CPU core/2496-GPU cores) would be ~ 300 = 2496*(0.7/2.6)*(Float/Double~0.5)

Summary

- Demonstrated feasibility of implementing electromagnetic physics processes and models for SIMD/SIMT architectures with common source codes
  - Implemented vectorized algorithms for multiple sampling methods
- Evaluated computing performance on vector CPU and accelerators
  - Tests demonstrate vector gains from SIMD of about 3.3-6.5 on KNC for 8-64 tracks
  - Performance potential of x30 on GPU, but requires $> 10^4$ tracks per process
- Outlook / Ongoing
  - Complete vector/GPU EM physics processes and models
  - Integration in full GeantV simulation and optimization

The GeantV Development Team

G.Amadio (UNESP), A.Ananya (CERN), J.Apostolakis (CERN) , A.Arora (CERN), M.Bandieramonte (CERN), A.Bhattacharyya (BARC), C.Bianchini (UNESP), R.Brun (CERN), Ph.Canal (FNAL), F.Carminati (CERN), L.Duhem (intel), D.Elvira (FNAL), A.Gheata (CERN), M.Gheata (CERN), I.Goulas (CERN), F.Hariri (CERN), R.Iope (UNESP), S.Y.Jun (FNAL), H.Kumawat (BARC), G.Lima (FNAL), A.Mohanty (BARC), T.Nikitina (CERN), M.Novak (CERN), W.Pokorski (CERN), A.Ribon (CERN), R.Sehgal (BARC), O.Shadura (CERN), S.Vallecorsa (CERN), S.Wenzel (CERN), Y.Zhang (CERN)

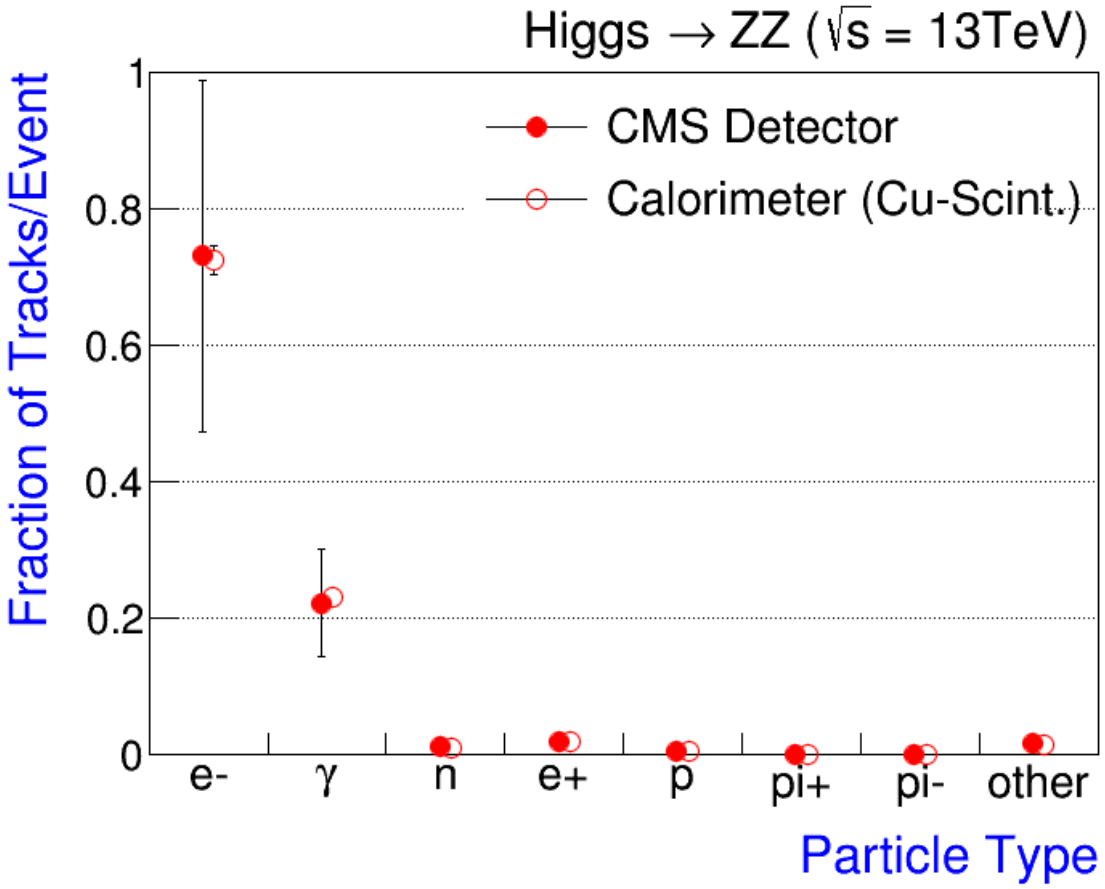
References

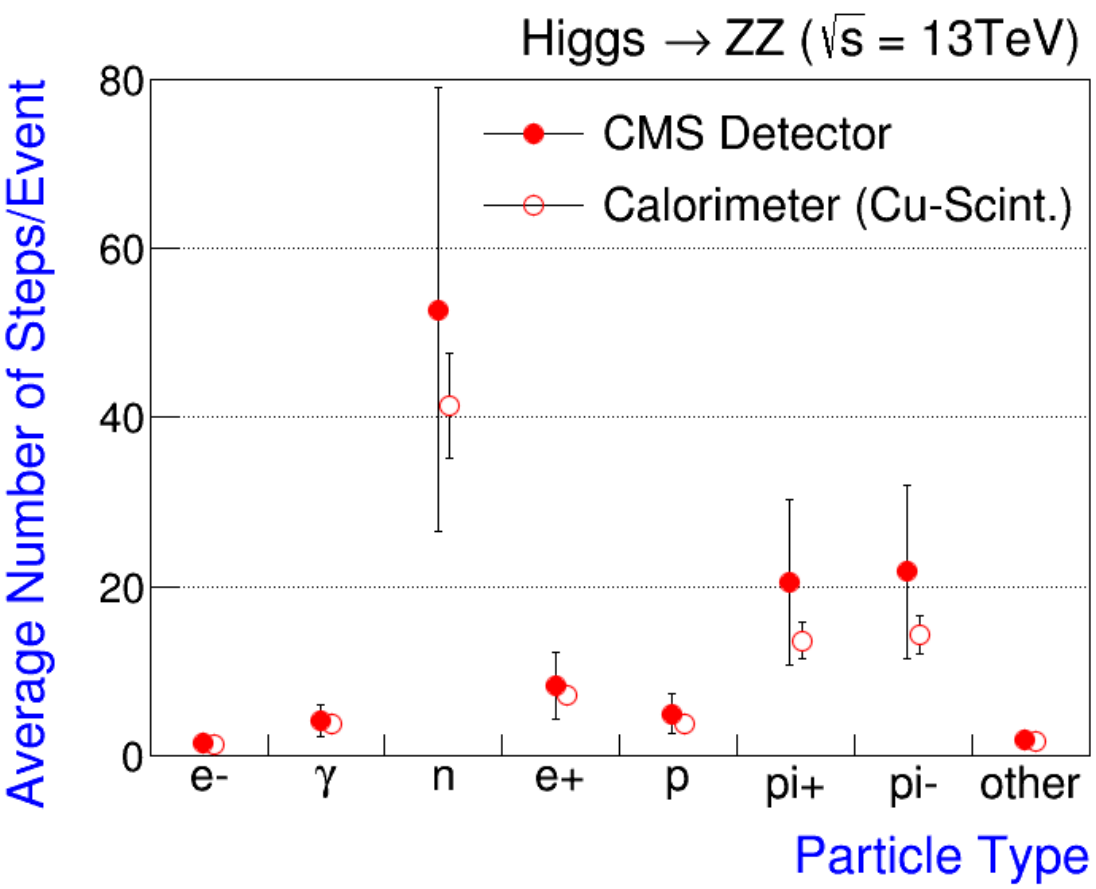
1. Walker A J, *ACM Trans. Math Software*. 3 **3**, 253-256 (1977)
2. Apostolakis J *et al.*, *J. Phys.: Conf. Ser.* 608 (2015) 012023
3. Vc, http://compeng.uni-frankfurt.de/?vc or https://github.com/VcDevel/Vc
4. UME::SIMD, https://bitbucket.org/edanor/umesimd

Backup Slides

Characteristics of HEP Particle Tracking

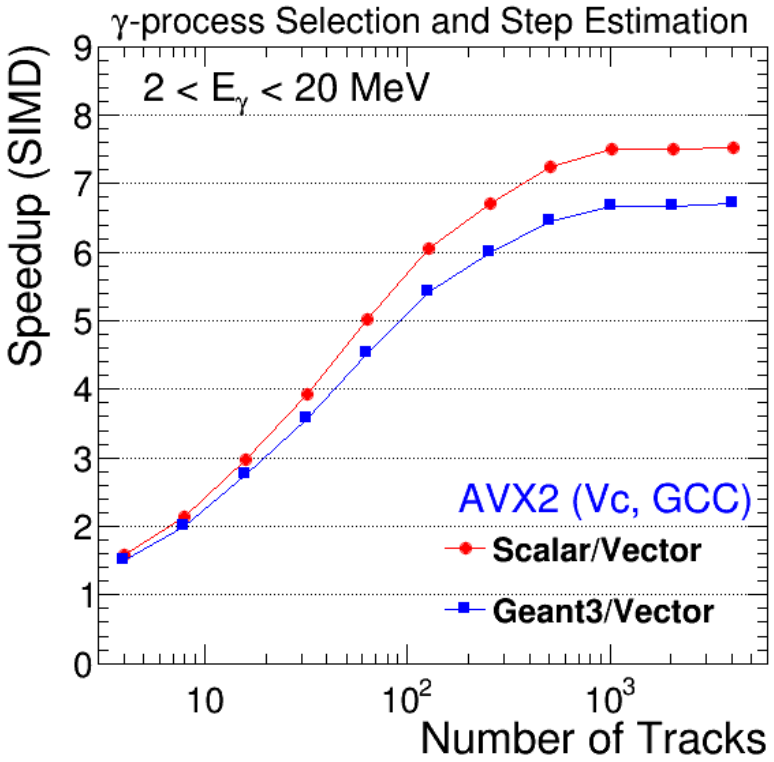
- Electrons are most populously produced
- Neutrons are long range in terms of the number of Geant4 steps

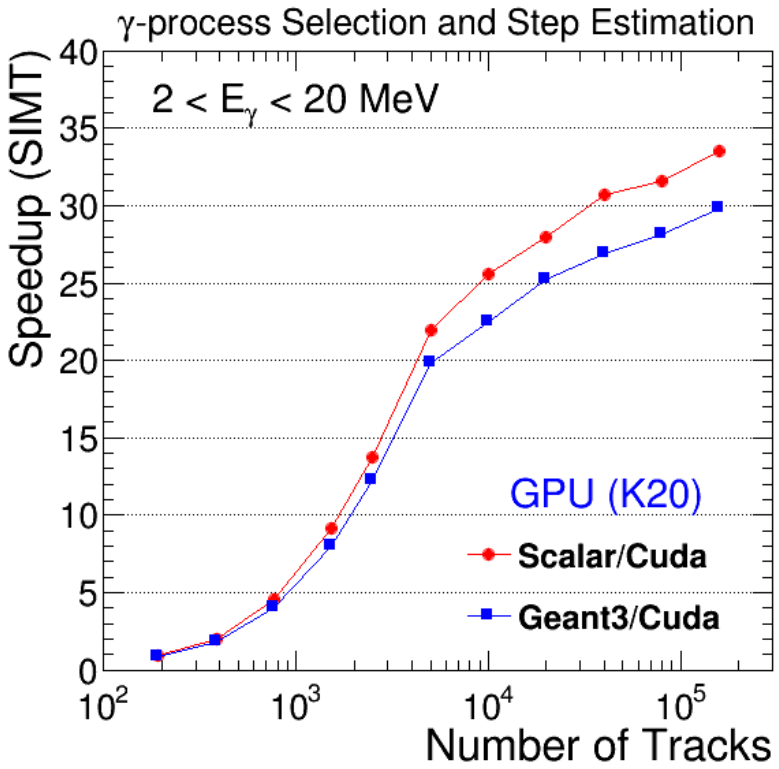




Preliminary Performance: AVX2 and CUDA

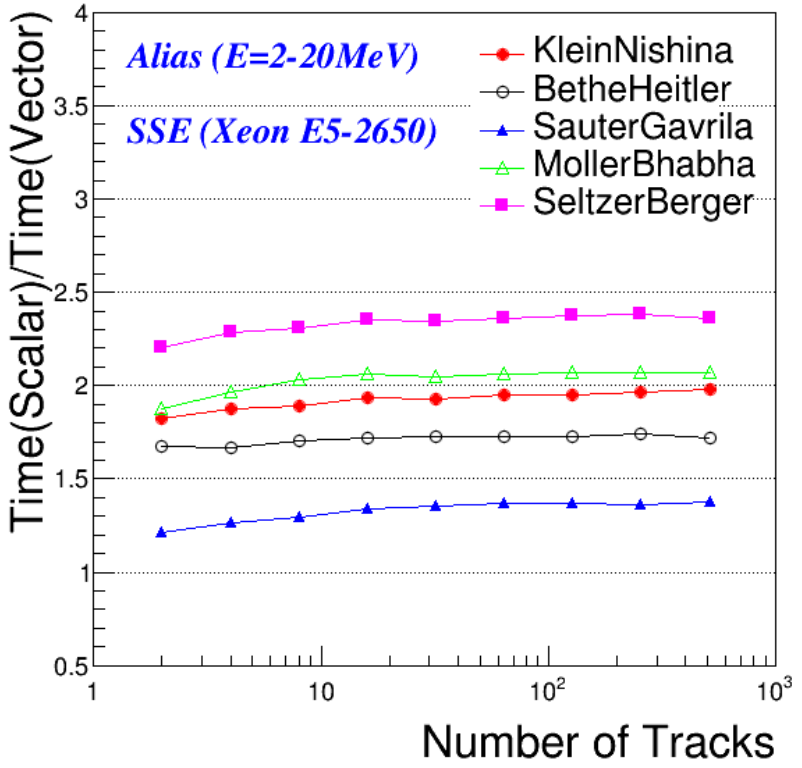
- Process selection and step length estimation with 3 photon processes
  - Scalar: backend implementation with the a.k.a. Geant3-tracking
  - Vector: Vc [2] with AVX2 on KNL
  - GPU: Cuda on NVidia Tesla K20

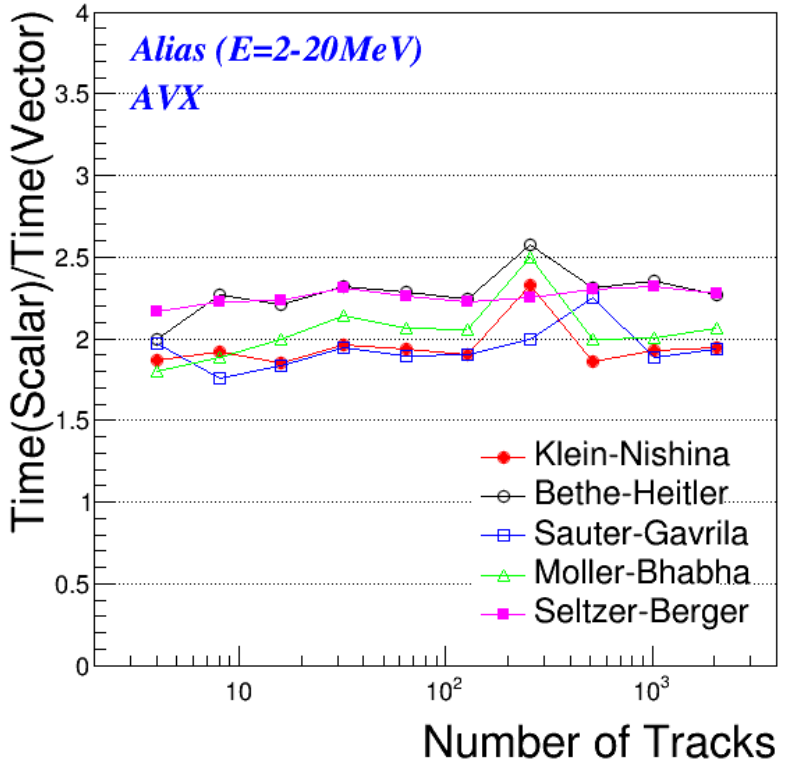




Preliminary Performance: Alias Sampling Method – Vector

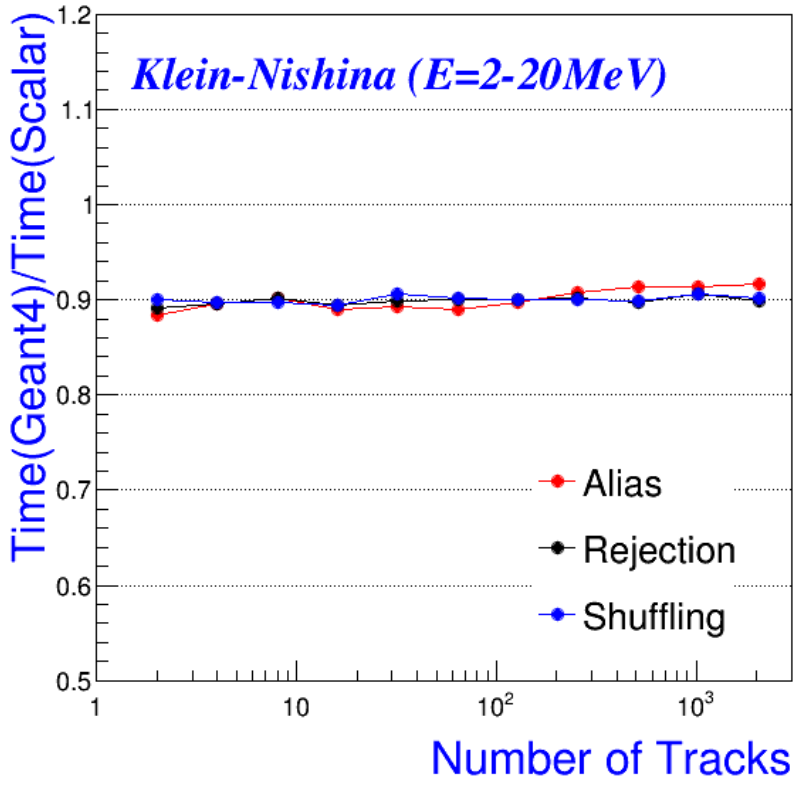
- Scalar/Vector
  - SSE (Intel Xeon E5 – 2650 @ 2.60 GHz) – SSE2
  - AVX (Intel Xeon E5 – 2620 @ 2.00 GHz)

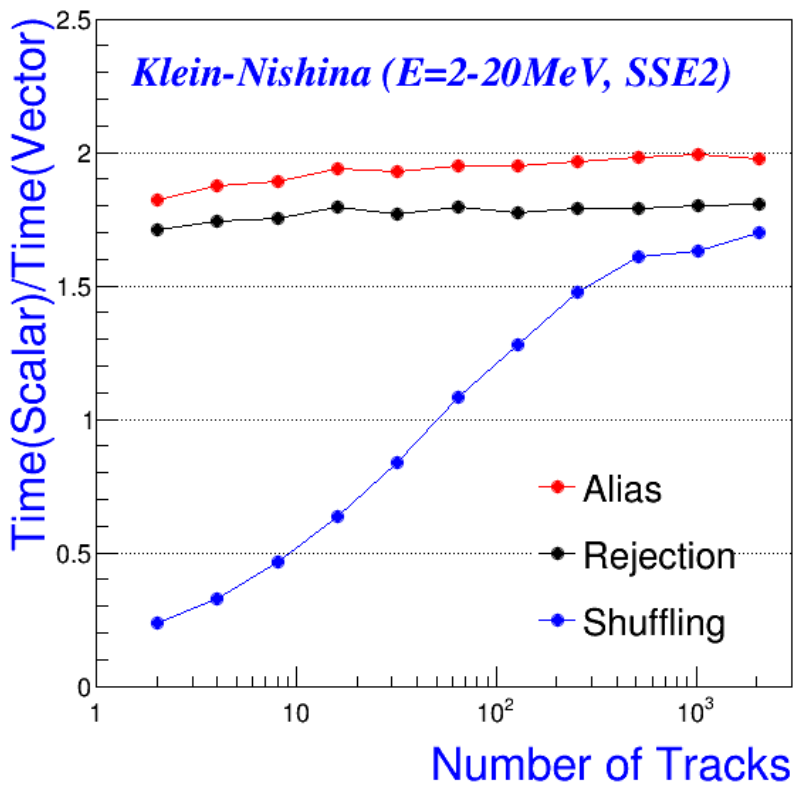




Performance: Alternative Sampling Method – Vector (SSE)

- Scalar/Vector
  - SSE (Intel Xeon E5 – 2650 @ 2.60 GHz) – SSE2





- Hybrid Compton for a small bucket of tracks
  - Alias [10keV,100MeV] + Rejection [100MeV,1TeV]

Performance: Alternative Sampling Methods - GPU

- GPU
  - GPU: Nvidia Kepler (K20), 2496 cores @ 0.7 GHz - <<<26,192>>>
  - Host: Intel Xeon E5 – 2650 @ 2.60 GHz