Electromagnetic Physics Models for Parallel Computing Architectures

Soon Yung Jun (Fermilab)

for 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), P.Canal (FNAL), F.Carminati (CERN), L.Duhem (intel), D.Elvira (FNAL), A.Gheata (CERN), M.Gheata (CERN), I.Goulas (CERN), R.Iope (UNESP), S.Y.Jun (FNAL), 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)

17th International Workshop on Advanced Computing and Analysis Techniques in Physics Research

January 18 – 22, 2016

UTFSM, Valparaíso (Chile)

Contents

- Introduction
- Challenges and Goals
- Considerations for Implementation
  - Algorithms
  - Performance
  - Portability
  - Other consideration
- Preliminary Validation and Performance
- Summary

Introduction

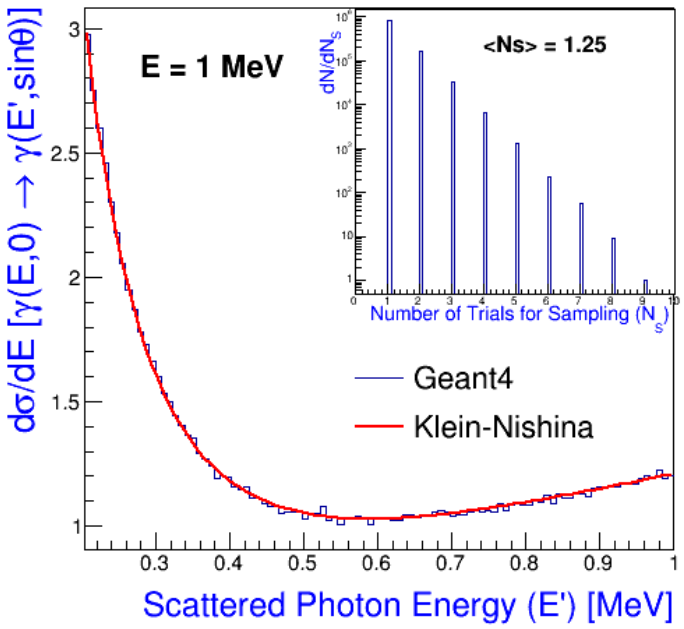
- Challenges and opportunities for HEP computing
  - ever-increasing computing power for future HEP programs
  - evolving hardware architectures and software technologies
  - promoting use of common resources and solutions to tackle next generation hardware (Forum for Computational Excellence) and advancing HEP community software (HEP Software Foundation)
- GeantV as next-generation detector simulation
  - simulation is one of the most CPU-intensive tasks in HEP computing and is largely experiment independent
  - concurrent framework for modern architectures, parallelized geometry and particle transport algorithms (track-level-parallelism)
  - see “GeantV: from CPU to accelerators”, by Andrei Gheata
- EM physics models for parallel computing architectures
  - 80% of CPU time is spent on electrons and photons in Geant4 for typical collider experiments, of which ~30% is on EM processes

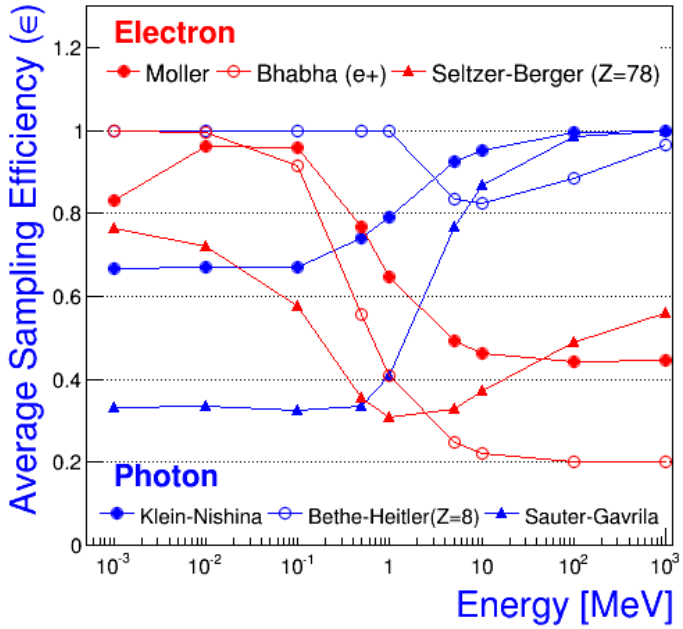
Challenges and Goals

- Characteristics of conventional HEP detector simulation
  - instructions are not reused often (many methods needed for each track, each with several branches)
  - memory bounded applications (cross section data, physics tables, geometry lookups, magnetic fields) – caching/locality
- No magical single solution for real world problems
  - particle transport in matter is an integral process (complexity)
  - explore every corner of technologies (diversity)
- Goal: write EM models, to deal with multiple tracks, to be accurate, fast and "portable"
  - maximum throughput for given resources
  - exploit both SIMD (vector pipeline) and SIMT (accelerators)
  - common source code between scalar, vector and accelerator (GPU, Xeon Phi)

Example of Algorithm Consideration: Sampling Methods

- Geant4 typically uses composition and rejection methods
  - accurate outcomes (ex. Compton at $E\gamma$ = 1MeV)
  - average sampling efficiency (ε) for photons and electrons

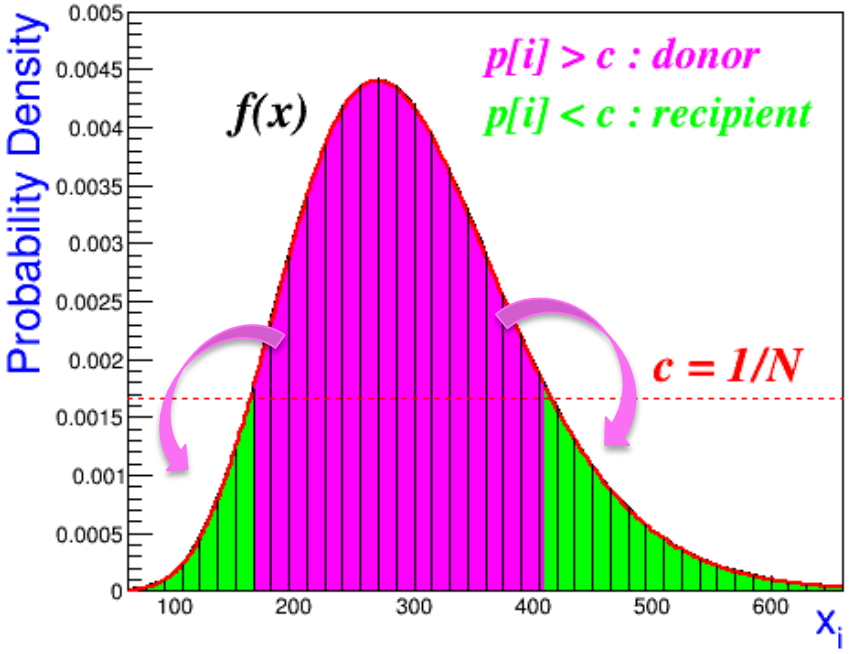


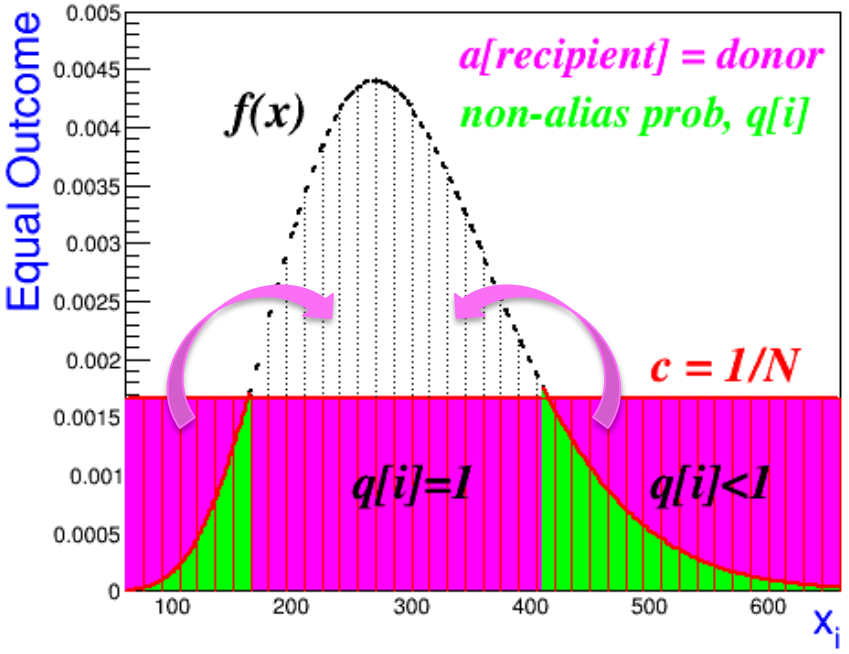


  - intrinsic problem for SIMD/SIMT: conditional branch (do-while)
  - probability to have the same number of trials for all n-vector operands (or n-threads) < $O(\varepsilon^n)$ (branch or thread divergence)
- Explore alternative methods to identify strengths (ability to vectorize), validity and limitations

Alias Sampling Method for N Discrete Outcomes

- Recast a N-discrete p.d.f to N equal probable events, each with likelihood 1/N = c (A. J. Walker, 1974) – effectively vectorizable
- Reproduce the original distribution by one trial sampling:
  - alias, a[recipient] = donor
  - non-alias probability, q[i] = probability not to take alias
  - for any random (xi), accept if rand(0,1) < q[i] or take the alias





Implementation Details

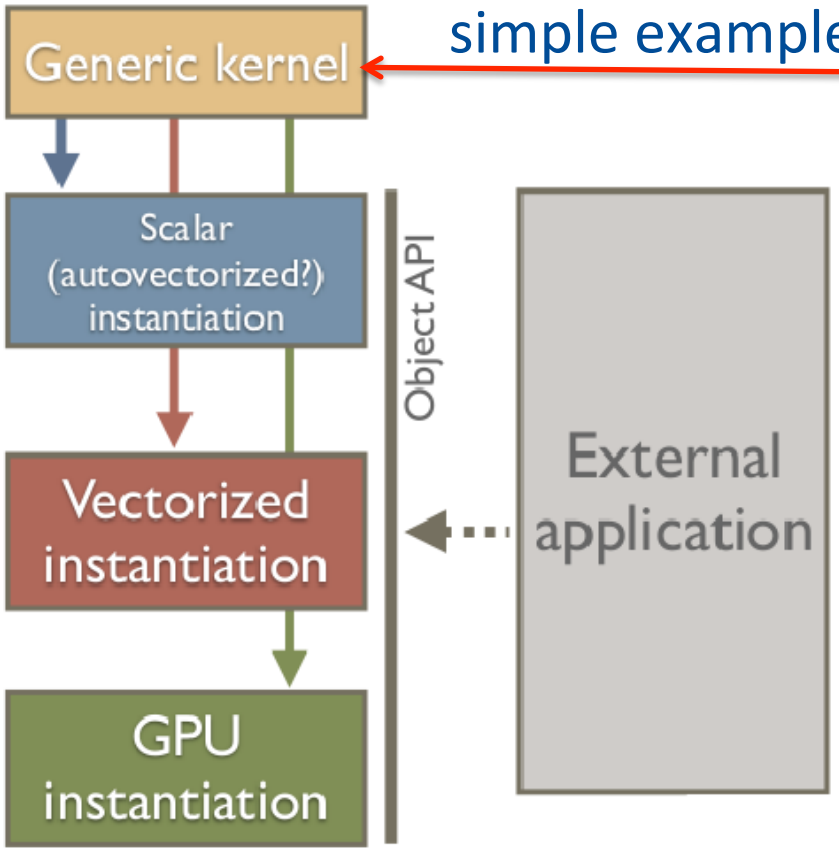
- Alias table: {a, q, p} [input-energy-bin][sampling-variable-bin]
  - equal log-division for input-energy bins
  - use cross section weighted bin positions (on the sampling axis)
  - for atomic dependent models, build tables for necessary elements
- EM physics models studied and size of sampling tables

Particle	Process	Model	Secondaries	Sampling Bin	Table Size
γ	Compton	Klein-Nishina	e^-	linear	100×200
	Conversion	Bethe-Heitler	e^-e^+	linear	$100\times100\times16$
	Photoelectric	Sauter-Gavrila	e^-	linear	100×200
e^-	Ionization	Moller-Bhabha	e^-	linear	100×100
	Bremsstrahlung	Seltzer-Berger	γ	log	$100\times100\times16$

- Sampling
  - table is chosen randomly from the ones below/above input-E
  - uniform sampling within a bin or a linear interpolation using p.d.f

Consideration for Portability: Scalar, Vector, CUDA

- Common interface to different architectures
  - generic kernel using abstract types (by backend)
  - template specialization
  - Technique pioneered in VecGeom (see also A. Gheata's talk)



simple example

```
inline namespace VECPHYS_IMPL_NAMESPACE { //cxx, cuda, ...

template<class Backend>
VECPHYS_CUDA_HEADER_BOTH
typename Backend::Double_t
GUAliasSampler::
SampleX(typename Backend::Double_t rangeSampled,
        typename Backend::Double_t probNA,
        typename Backend::Index_t  aliasInd,
        typename Backend::Index_t  icol,
        typename Backend::Double_t fraction) const
{
  typedef typename Backend::Bool_t   Bool_t;
  typedef typename Backend::Index_t  Index_t;
  typedef typename Backend::Double_t Double_t;

  Double_t r1 = UniformRandom<Backend>(ftate,fThreadId);

  Bool_t useDirect = (r1 <= probNA);
  Double_t binSampled = rangeSampled * fInverseBinSampled;

  Index_t      icolDist = icol;
  MaskedAssign(!useDirect, aliasInd, &icolDist); //if-else

  Double_t xd = icolDist*binSampled;
  Double_t xu = xd + binSampled;

  Double_t x = (1 - fraction) * xd + fraction* xu;

  return x;
}

} // end namespace impl
```

Other Considerations

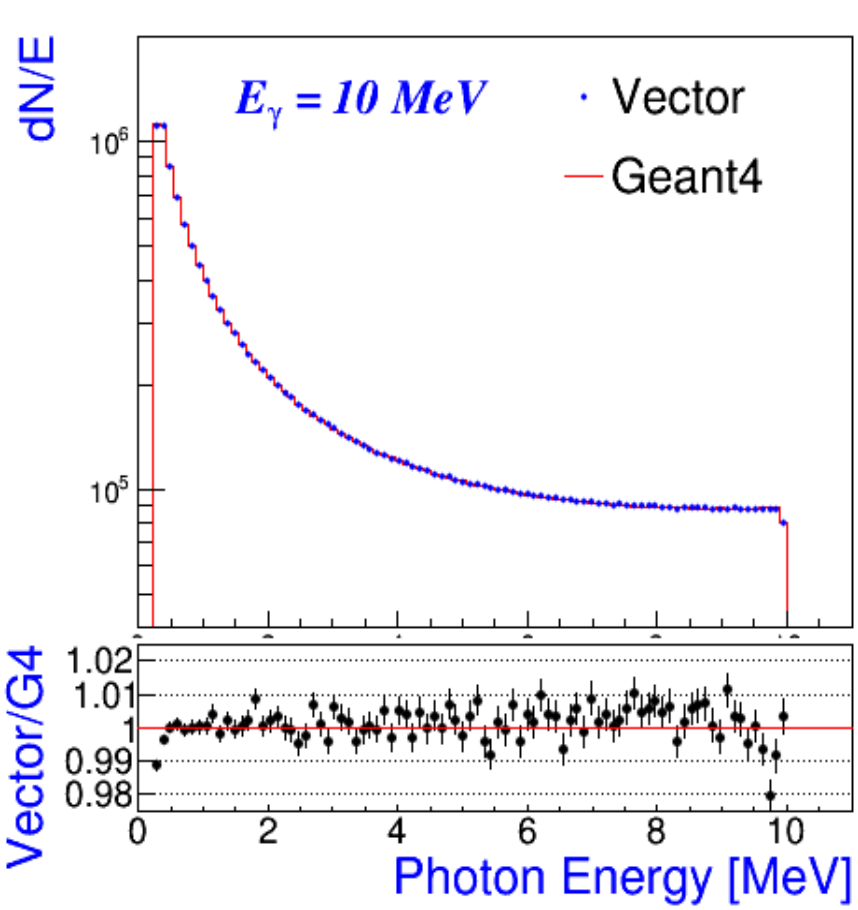
- Static polymorphism using CRTP (curiously recurring template pattern)
  - virtual is not allowed for template<backend>
  - avoid virtual layers (use early binding)
- Core techniques and patterns for vectorization: Vc library
  - conditional branches: mask
  - use gather to fetch data and place it contiguously
- Data structure
  - SoA (for vector) and AoS (for scalar, cuda)
  - Pre-sorting tracks by energy to apply different algorithms across different energy regions (also may reduce latencies) - optional
- Random number generation
  - Vc random numbers
  - Cuda pRNG library (CURAND)

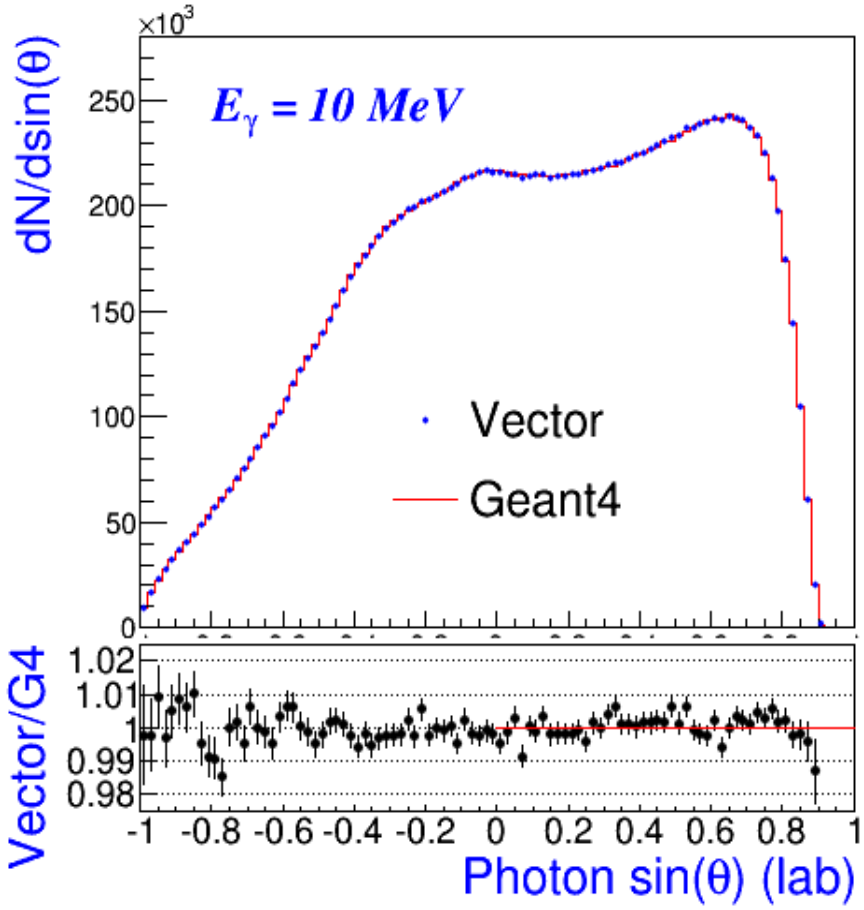
Preliminary Validation and Performance Evaluation

- Physics validation
  - simulate interactions at fixed input energy points
  - compare results (final states of interaction) w.r.t Geant4 (code extracted from Geant4)
- Performance measurement
  - 10 experiments, 100 repetitions per experiment
  - time for simulating N secondary interactions with a exponentially falling input energy spectrum in the range;
    - input energy range = [2MeV:20MeV] with 16 Z-elements
    - also tested in E [10keV:1MeV], E [10MeV:1GeV], E [10GeV:1TeV]
    - note that performance of Geant4 depends on the energy range
  - sample and store secondary particles, and update primary tracks
  - speedup = scalar/vector or scalar/GPU as the number of input particles (tracks) for the same task

Preliminary Validation: Alias Sampling Method vs. Geant4

- Compton: scattered photon energy and angular distributions with input photon energy = 10 MeV (Vector)

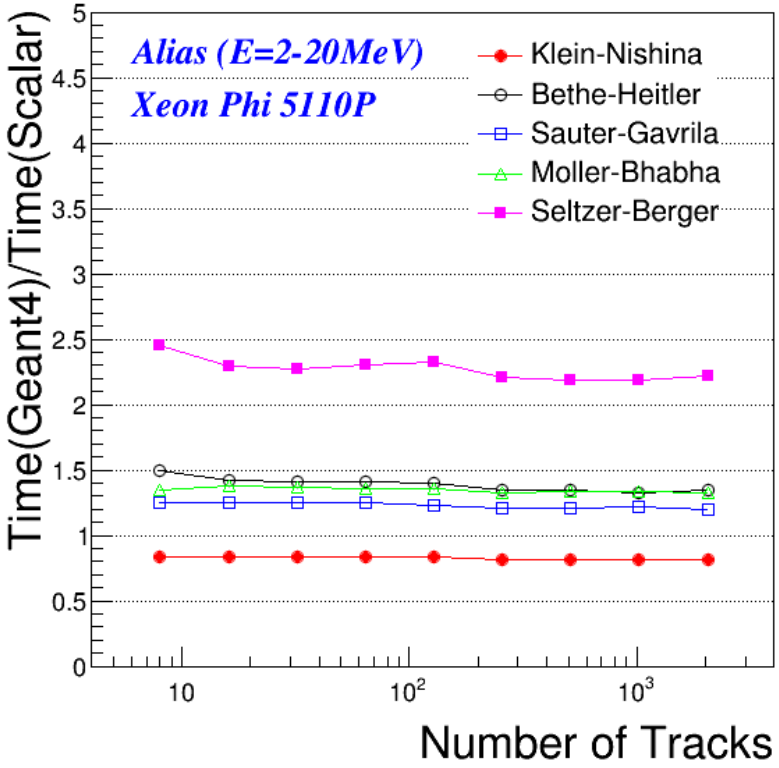


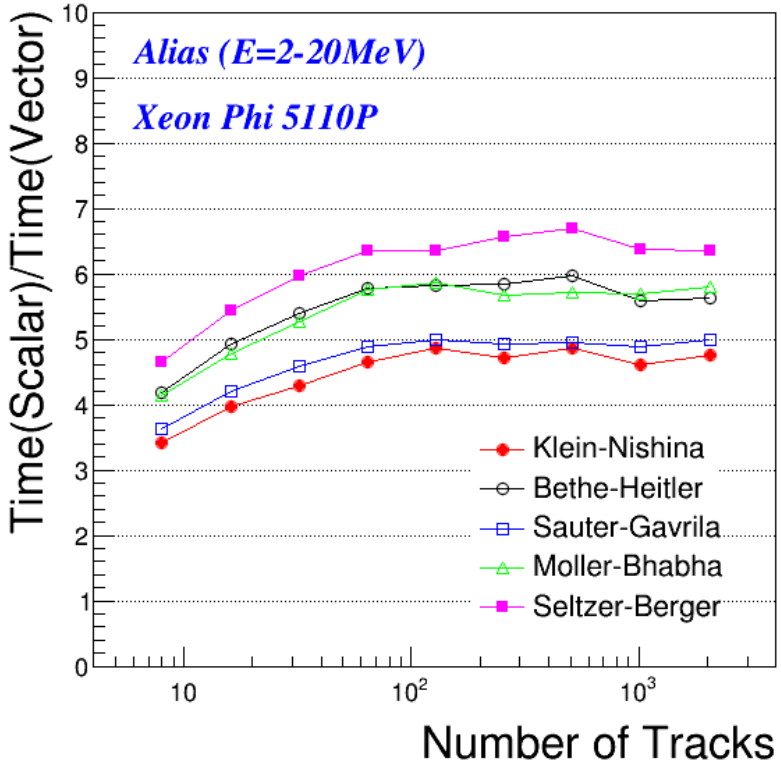


1%-level agreement up to 100 MeV with alias table size [100,200]

Preliminary Performance: Alias Sampling Method

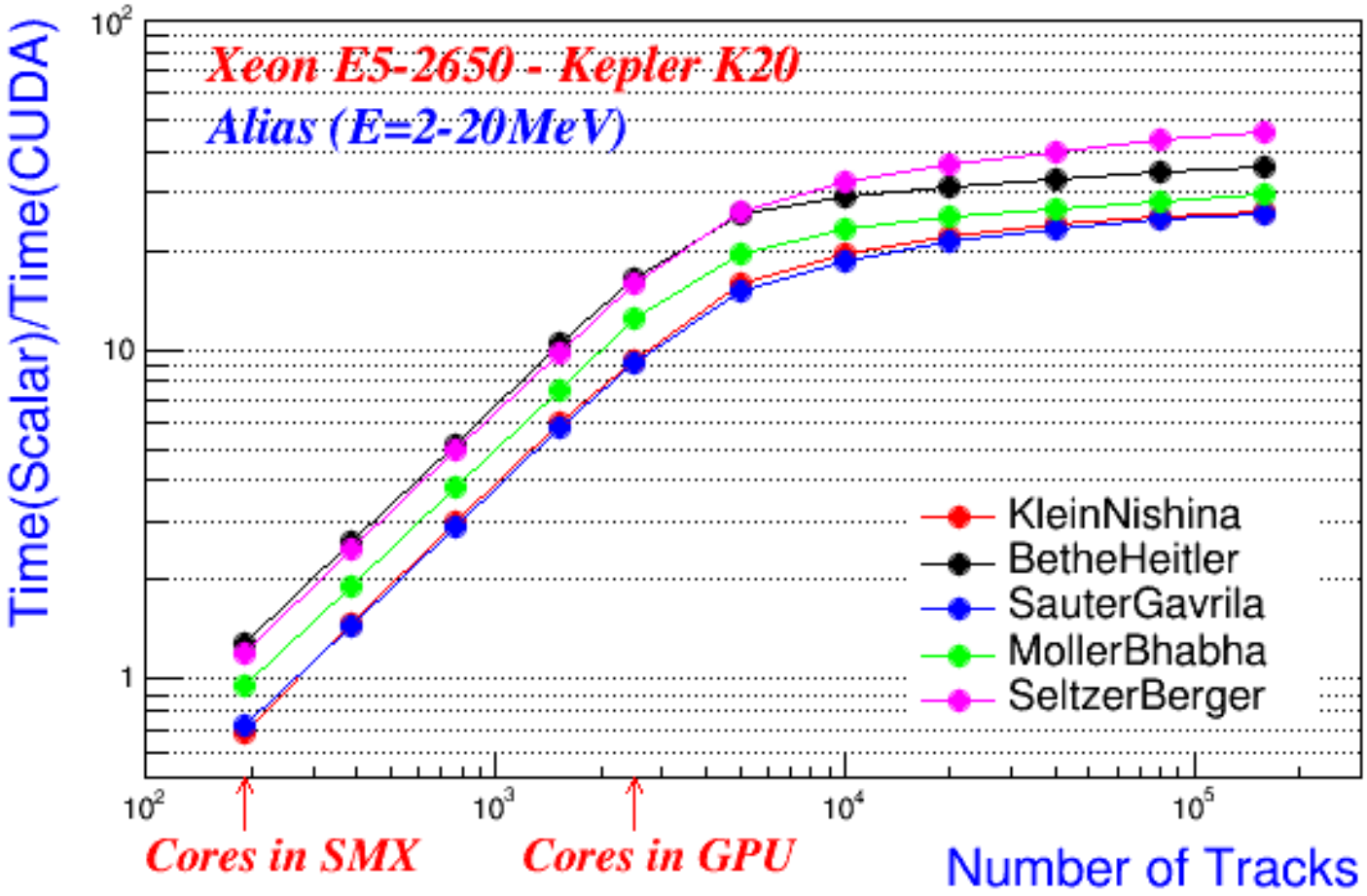
- Vector performance
  - input particle energy: 2-20 MeV (valid range for all models)
  - using 16 elements (random for each track)
  - MIC (Intel Xeon Phi 5110P 60 cores @ 1.053 GHz) - 8 vector pipelines for double precision – see also SSE/AVX in backup





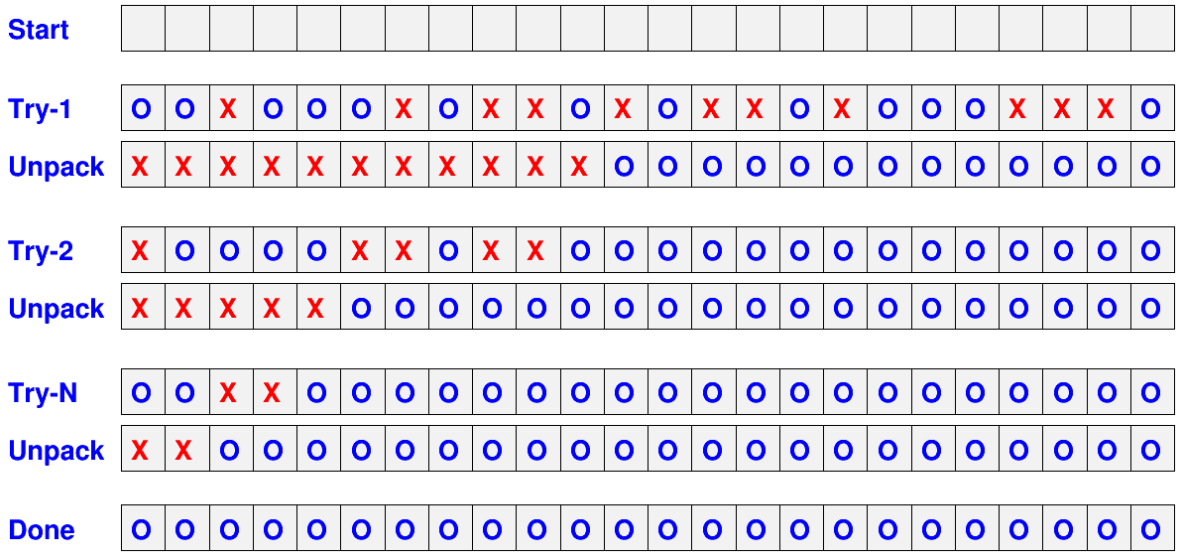
Preliminary Performance: Alias Sampling Method – GPU

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



Alternative Sampling Methods

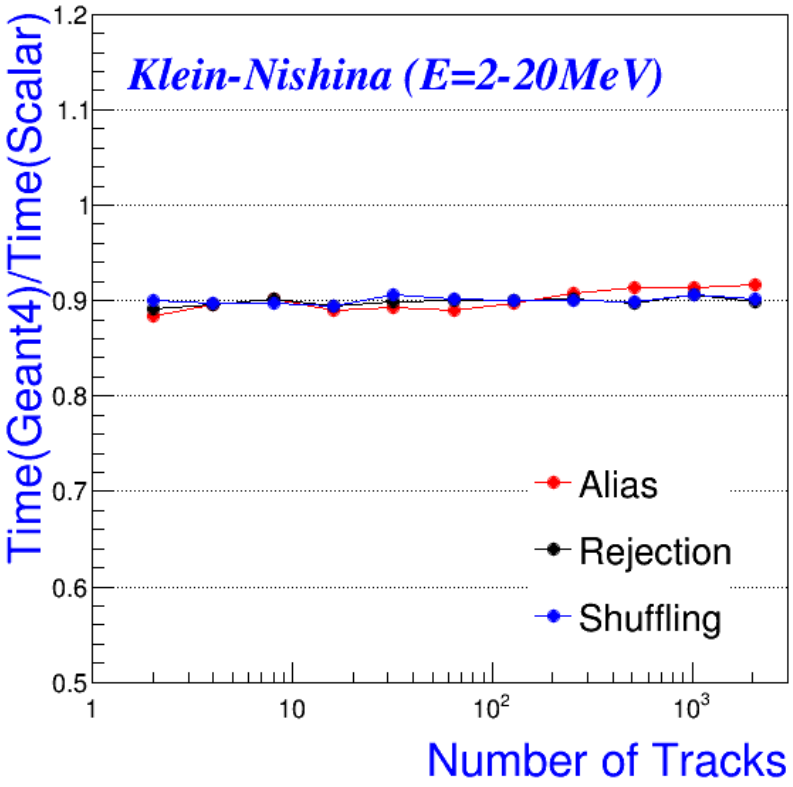
- Limitation of the discrete alias sampling method
  - the alias method with a finite bin size is subject to have biased outcomes if p.d.f is neither near constant nor linear within a bin
  - Ex.: $d\sigma/d\epsilon[\gamma(E,0) \rightarrow \gamma'(E', \sin\theta)] \rightarrow \epsilon^{-1} = E/E'$ for small ε
- Alternative techniques using the composition and rejection
  - Parallel (vector) + Sequential (scalar) loop - (see backup)
  - Shuffling (repeat try and unpack, overhead for reorganizing data)

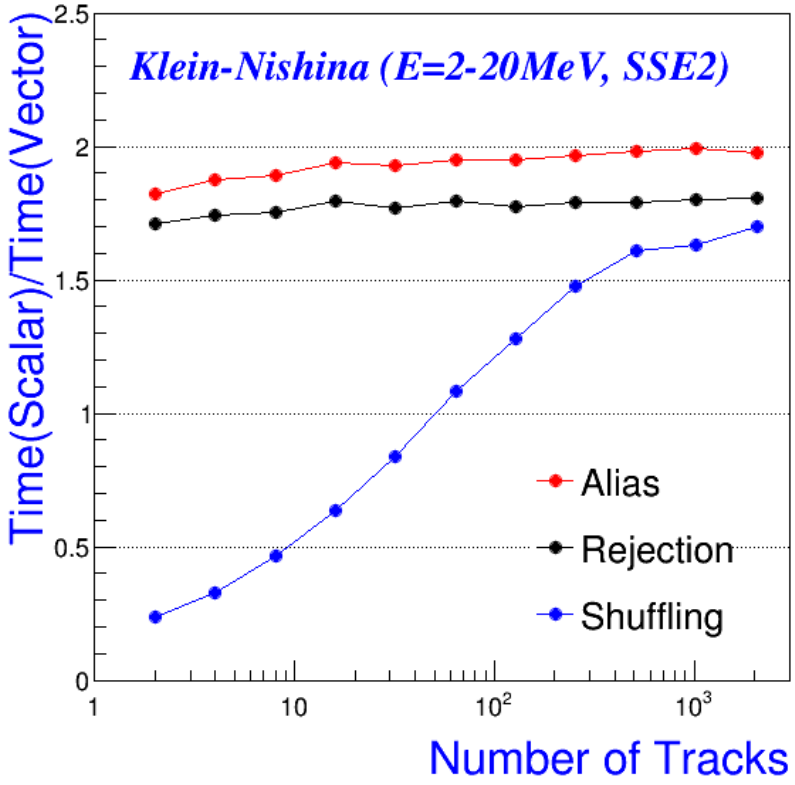


  - hybrid (mixture of different methods in the parameter space)

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]

Summary

- Demonstrated feasibility of implementing electromagnetic physics models for SIMD/SIMT architectures with common source code
- Integrated different methods (alias, adaptions of accept/reject)
- Validating physics results with Geant4 and evaluating computing performance
- Outlook
  - optimize further for both SIMD and SIMT
  - Integrate in the GeantV framework
  - measure performance in full simulation

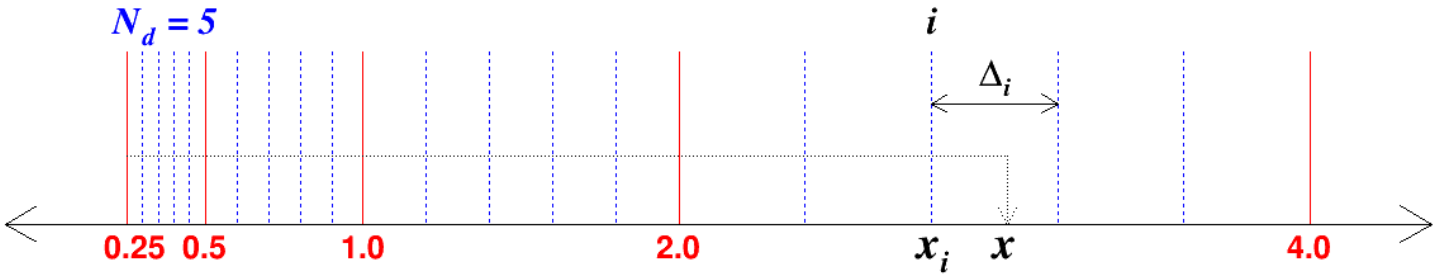
Backup Slides

Example of Performance Consideration: Binning Technique

- Construct bins (x_i) with a equal divisor (N_d) between the power of 2 from 2^{Nmin} to 2^{Nmax} for any integer N_{min} and N_{max}

$$x_i = [1. + (\frac{i\%N_d}{N_d})] \cdot 2^{\frac{i}{N_d}+\mathrm{N_{min}}}, \qquad i \in [0, (N_{max} - N_{min}) \cdot N_d + 1]$$

- For any x in [2^{Nmin}, 2^{Nmax}], the bin number (i) and the fraction (δ)



$$\begin{aligned} M &= \mathrm{frexp}(x, \&E) \\ i &= N_d \cdot (E - 1 - N_{min}) + \mathrm{floor}[(2M - 1) \times N_d] \\ \Delta_i &= [\mathrm{ldexp}(1., E - 1)]/N_d \\ \delta &= \frac{(x - x_i)}{\Delta_i}, \qquad \delta \in [0, 1) \end{aligned}$$

Random Number Generators

- Pseudo random number generator (pRNG)
  - Sequential: std::rand()
  - Vector: Vc Random_t
  - Cuda pRNG library (CURAND)
- Performance: 10K pRNG generation time in [ms]
  - CPU (Xeon x5650, 2.67GHz)

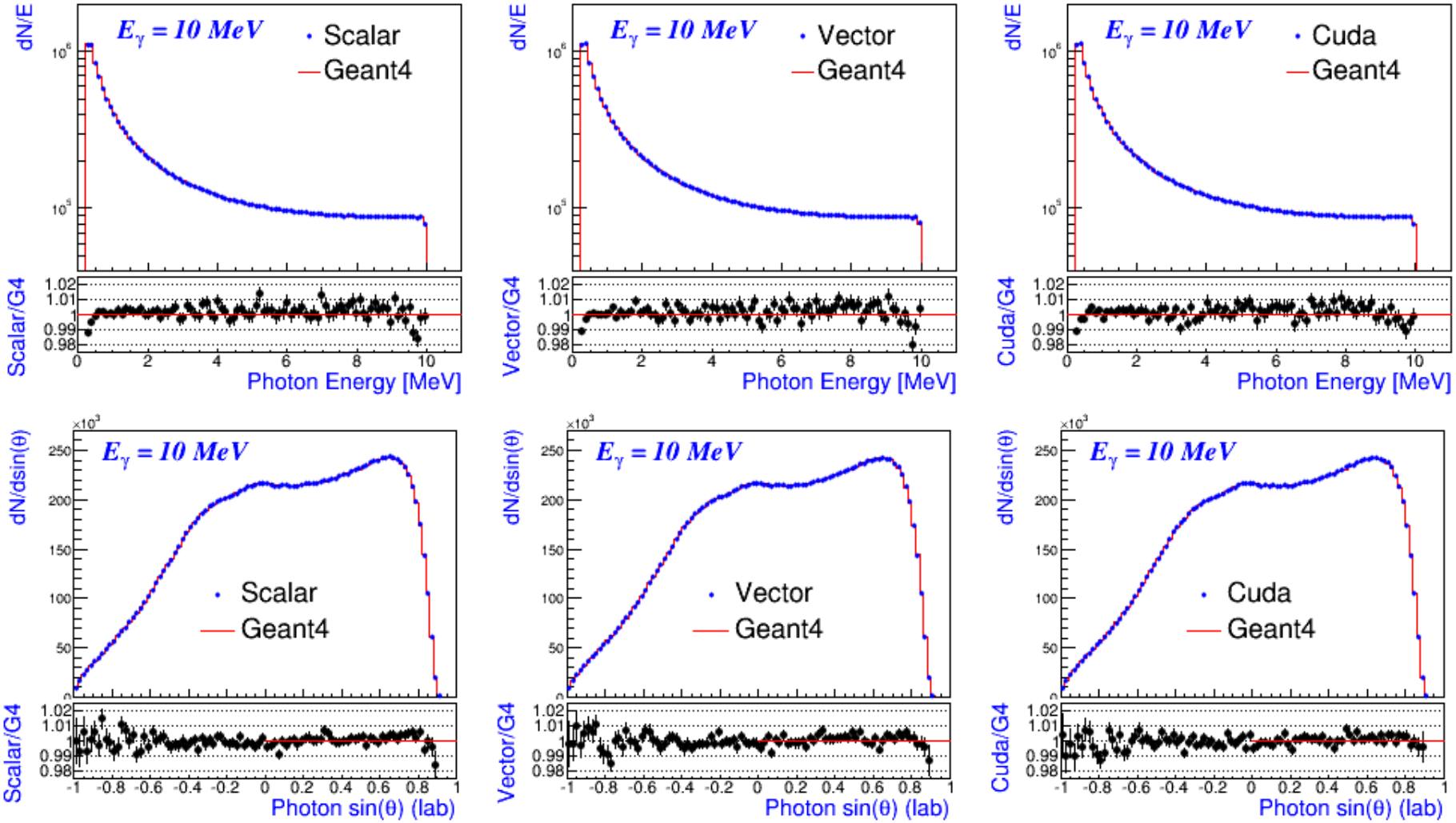
pRND generator	10K pRNG [ms]
Rand()	3.66
Vc random_t (SSE)	1.44

  - GPU (Nvidia K20M)

CURAND	Init States for 64 blocks [ms]	10K pRNG for 256 threads [ms]
XORWOW	4.12	7.92
MRG32k3a	5.02	21.88
MTG32	0.69	31.94

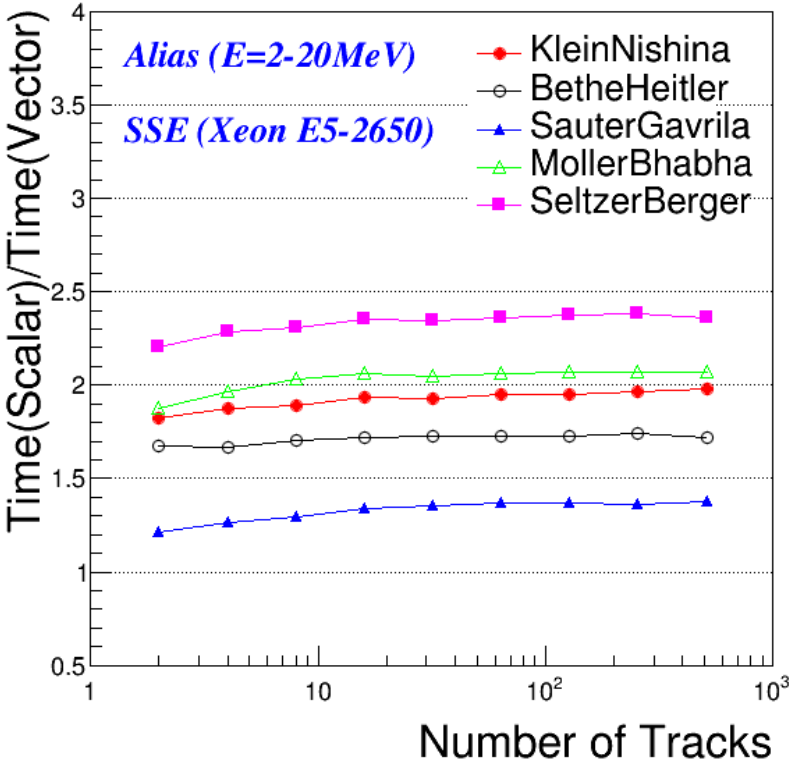
Preliminary Validation: Alias Sampling Method vs. Geant4

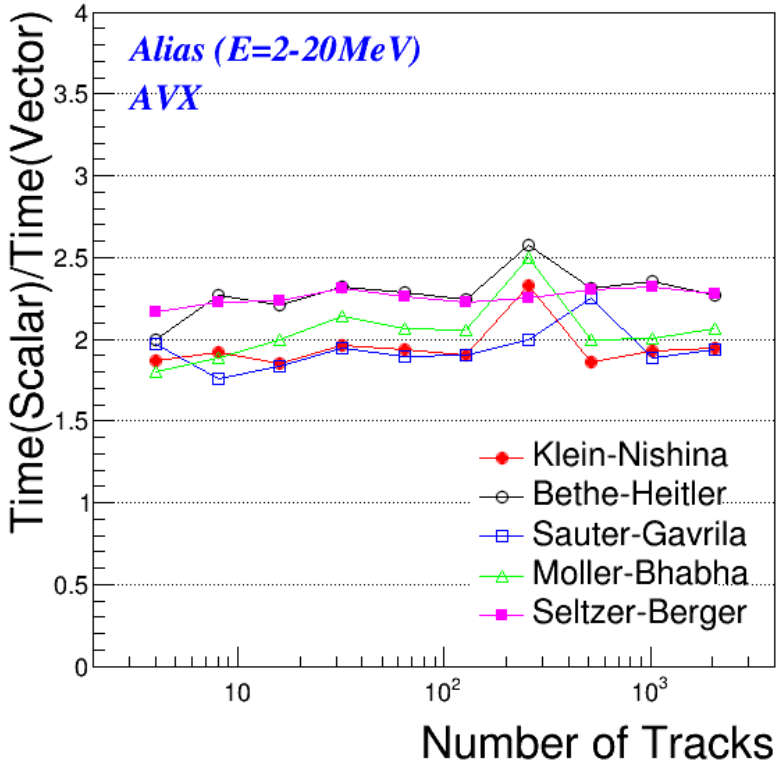
- Compton: scattered photon energy and angular distributions with input photon energy = 10 MeV (Scalar, Vector, Cuda)



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)





Limitation of Discrete Alias Sampling

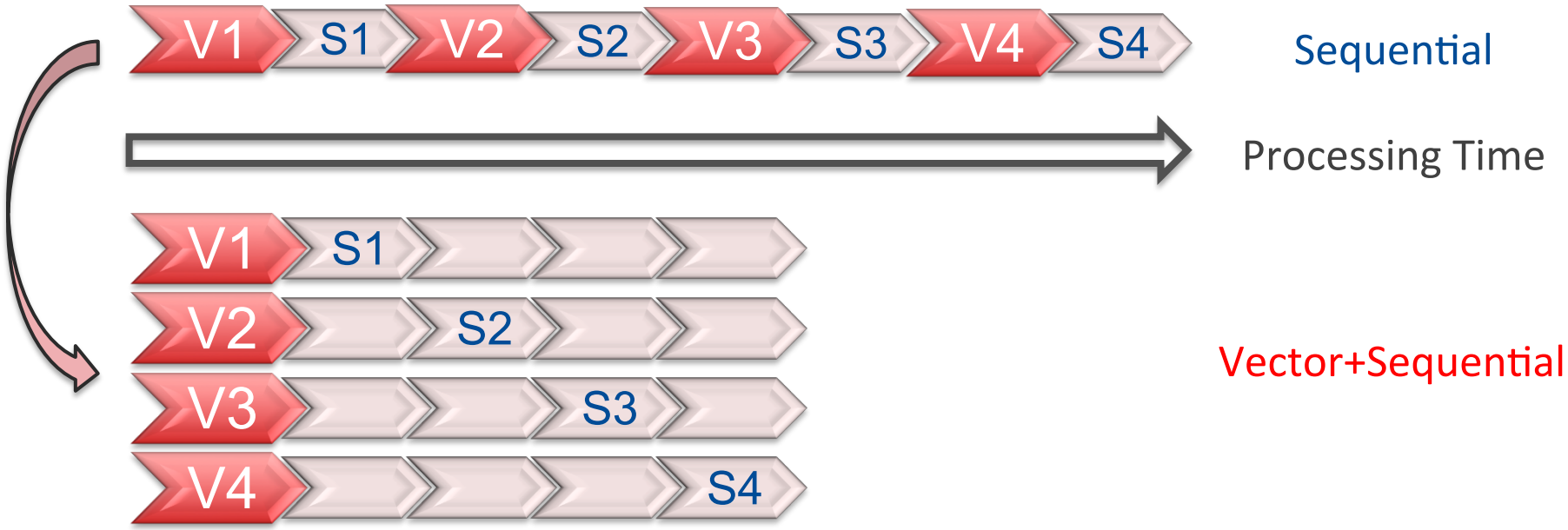
- The alias method with a finite bin size is subject to have biased outcomes if p.d.f. of sampling, f(x) is significantly non-linear within a bin
  - alias method may reproduce the average of f(x) within a bin (i.e, building alias tables at integrated averaged points and a interpolation within the sampling bin)
  - still no guarantee to sample g(f(x)) correctly even though <f(x)> is well reproduced
  - this is also true for the inverse cumulative approach
  - Example for Compton scattering: for large E and small E'

$$\frac{d\sigma}{d\epsilon}[\gamma(E,0) \rightarrow \gamma'(E', \sin\theta)] = \frac{X_o n \pi r_o^2 m_e}{E^2}[\frac{1}{\epsilon} + \epsilon][1 - \frac{\epsilon \sin^2\theta}{1+\epsilon^2}] \propto \frac{1}{E^2}\frac{1}{\epsilon}$$

$$\epsilon = \frac{E'}{E} = \frac{m_e}{m_e + E(1-\cos\theta)}$$

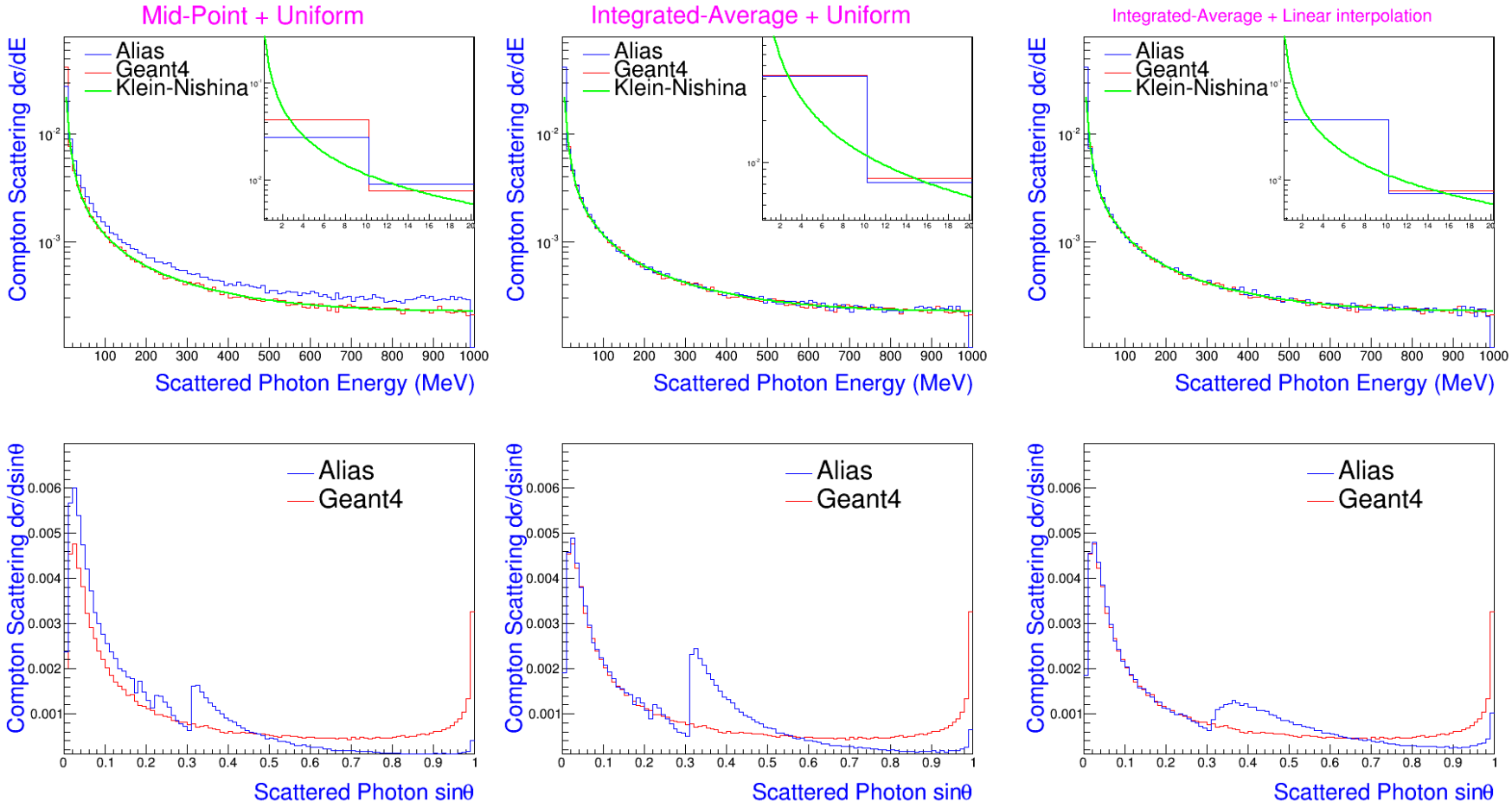
Vector + Sequential (Scalar) loop

- Sequential (scalar) operation for non-vectorizable (do-while) part of the composition and rejection method:
  - Amdahl's Speedup = 1/[(1-P) + P/N]
    where N = vector width for Vc and P = fraction of parallel code
  - Example with N = 4, V=vectorizable, S=sequential, P=V/(V+S)



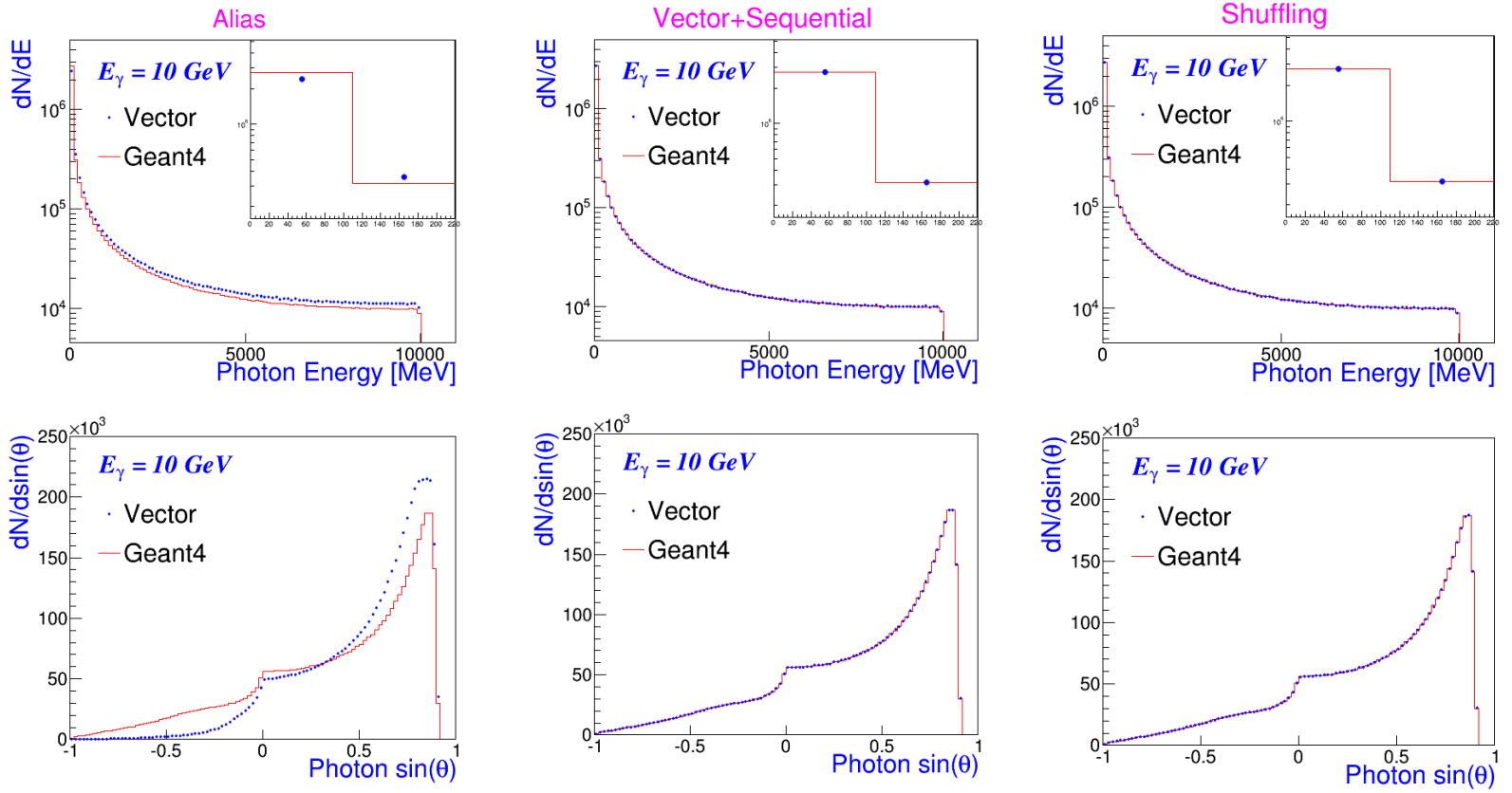
Limitation of Discrete Alias Sampling

- Compton scattering for Eγ = 1GeV
  - mid-point + uniform sampling
  - Integrated average + uniform sampling
  - Integrated average + linear sampling



Validation: Alternative Sampling Methods

- Compton scattering for input Eγ = 10 GeV: (Vector/Geant4)
  - Alias: Integrated average + linear sampling, table (100, 200)
  - Vector + Sequential
  - Shuffling method:



- Hybrid compton: alias (upto 100 MeV) + alternative method

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