Electromagnetic Physics Models for Parallel Computing Architectures

Electromagnetic Physics Models for Parallel Computing Architectures

G Amadio[1], A Ananya[2], J Apostolakis[2], A Aurora[2], M Bandieramonte[2], A Bhattacharyya[3] C Bianchini[1 6], R Brun[2], P Canal[4], F Carminati[2], L Duhem[5], D Elvira[4], A Gheata[2], M Gheata[2 7], I Goulas[2], R Iope[1], S Y Jun[4], G Lima[4], A Mohanty[3], T Nikitina[2], M Novak[2], W Pokorski[2], A Ribon[2], R Seghal[3] O Shadura[2], S Vallecorsa[2], S Wenzel[2], and Y Zhang[2]

[1]Parallel Computing Center at São Paulo State University (UNESP), São Paulo, Brazil
[2]CERN, EP Department, Geneva, Switzerland
[3]Bhabha Atomic Research Centre (BARC), Mumbai, India
[4]Fermilab, MS234, P.O. Box 500, Batavia, IL, 60510, USA
[5]Intel Corporation, Santa Clara, CA, 95052, USA
[6]Mackenzie Presbyterian University, São Paulo, Brazil
[7]Institute of Space Sciences, Bucharest-Magurele, Romania

E-mail: `syjun@fnal.gov`

Abstract. The recent emergence of hardware architectures characterized by many-core or accelerated processors has opened new opportunities for concurrent programming models taking advantage of both SIMD and SIMT architectures. GeantV, a next generation detector simulation, has been designed to exploit both the vector capability of mainstream CPUs and multi-threading capabilities of coprocessors including NVidia GPUs and Intel Xeon Phi. The characteristics of these architectures are very different in terms of the vectorization depth and type of parallelization needed to achieve optimal performance. In this paper we describe implementation of electromagnetic physics models developed for parallel computing architectures as a part of the GeantV project. Results of preliminary performance evaluation and physics validation are presented as well.

1. Introduction

The latest evolution of parallel computing architectures and software technologies makes a variety of concurrent programming models applicable not only to high performance computing in advanced scientific computing research, but to the conventional computing intensive application such as the high energy physics (HEP) event simulation. Faced with ever-increasing demands of computing power required for present and future experimental (HEP) programs, there have been coherent efforts to promote efficient use of common resources and solutions to tackle next generation hardware [1] and to advance HEP community software [2]. Within HEP software ecosystem, event simulation is one of the most time consuming parts of the work flow, but a large portion of code used for HEP detector simulation is independent from the details of individual experiments and can be shared substantially. As one of domain specific research and development activities, the GeantV project [3] (GeantV) launched in 2013 studies performance

gains from propagating multiple tracks from multiple events in parallel, improving instruction throughput and data locality in the particle transportation process. Using code specialized to take advantage of the hardware specific features, it aims to leverage both the vector pipelines in modern processors and the massively parallel capability of coprocessors, including the Xeon Phi and general purpose GPUs. GeantV has three major components that need to be adapted for concurrent programing models: concurrent framework, vectorized geometry, and physics modeling. In this paper we focus on the techniques explored to enhance electromagnetic (EM) physics models used in HEP utilizing emerging parallel hardware architectures.

2. Parallelization of Electromagnetic Physics Models

HEP detector simulation models the passage of particles (tracks) through matter. The typical HEP event consists of a set of particles produced by a primary collision and subsequent secondary interactions or decays. Geant4 [4, 5], the most widely used simulation toolkit in contemporary HEP experiments, processes all tracks of an event sequentially even though multiple events can be processed simultaneously (event-level parallelism) using multi-processors or multi-threading. On the other hand, GeantV explores particle-level parallelism by grouping similar tracks and processing them concurrently with fine-grained parallelism to maximize locality of both data and instructions. To take full advantage of SIMD (Single Instruction Multiple Data) or SIMT (Single Instruction Multiple Threads) architectures, identical instructions without conditional branches should be executed on multiple data simultaneously. Therefore, vectorization of physics models requires algorithms adapted to utilize instruction level parallelism as much as possible.

An essential component of particle interactions with matter is the generation of the final state described by a physics model associated with the selected physics process for a given particle. As electrons and photons are primary components of the particles produced in typical collider detectors, simulation of EM physics processes is one of leading consumers of computing resources during HEP event simulation. In most EM physics models, the atomic differential cross section of the underlying physics process plays a central role in modifying the kinematic state of the primary particle or producing secondary particles. In Geant4, combined composition and rejection methods [6, 7, 8] are often used to sample variables following probability distribution functions used in EM physics models, as inverse functions of their cumulative distributions are not analytically calculable in general. However, composition and rejection methods are not suitable for SIMD computing architectures due to repetitive conditional trials until a random draw is selected. Alternative sampling techniques that can be effectively parallelizable for SIMD and SIMT architectures are the alias method [9, 10] and the shuffling technique similar to the pack pattern used in parallel programing models [11].

The alias sampling method is similar to the acceptance-rejection method, but it uses an alias outcome for the rejected case which is thrown away in the traditional acceptance-rejection method. It recasts the original probability density function with N equally probable events, each with likelihood $c = 1/N$, but keeps information of the original distribution in the alias table that consists of the alias index and the non-alias probability.

Even though the alias method with a finite bin size reproduces an input probability distribution function within a desired precision, it is subject to have biased outcomes if p.d.f. is significantly nonlinear within a bin. One alternative sampling method that still incorporates the combined composition and rejection technique is the shuffling technique. It eliminates accepted trials from a collection and reorganize failed elements in contiguous memory for the next vectorization loop and repeat the process until all retained elements are accepted. The shuffling process will reproduce the original p.d.f. at the cost of the overhead of gathering data in each interaction and can be used when the alias method has a bias.

To simulate kinematical distributions of secondary particles produced by EM physics models, both the alias method and the shuffling method are tested and their results are compared in

different energy regions. A list of EM physics processes and models for high energy electrons and photons that are currently implemented is summarized in Table 1.

Table 1. A list of electromagnetic physics processes and models of electron and photon processes that are implemented and tested for SIMD and SIMT architectures.

Primary	Process	Model	Secondaries	Survivor
γ	Compton Scattering	Klein-Nishina	e^-	γ
	Pair-Production	Bethe-Heitler	e^-e^+	–
	Photo-Electric Effect	Sauter-Gavrila	e^-	–
e^-	Ionization	Moller-Bhabha	e^-	e^-
	Bremsstrahlung	Seltzer-Berger	γ	e^-

3. Implementation

Besides performance and scalability, another important objective of GeantV design is portability. To enable the use of multiple modern hardware architectures, GeantV uses backends [14, 15], which are software layers between the platform-independent generic simulation code and the hardware-specific details and their software-related constructs like SIMD intrinsics, MIC pragmas or CUDA `C++` extensions. The main purpose of the backends is to isolate all the complexity of low-level, high performance details behind simplified abstractions which are then available for use by carefully designed, generic kernels. The Vc library [12] is used for the choice of the vector backend for this paper, but UME::SIMD [13] is also being integrated to promote explicit SIMD vectorization.

There are three categories of methods implemented in vector physics package: initialization and management, interfaces to physics processes, and kernels. Kernels are high-performance versions of performance-critical algorithms and based on the data structures offered by the backends. In order to take full advantage of the performance capabilities of the underlying hardware, some important choices were made:

- Inlined functions are used extensively, to avoid the overhead due to function calls.
- Static polymorphism is used instead of virtual inheritance. Virtual function calls inside the kernels are explicitly avoided since kernels themselves are coded in terms of `C++` templates, with a specific backend as the template parameter.
- Branching of execution flow is strictly minimized and mask operations are used whenever necessary.
- The structure of arrays (SOA) is used for track data
- Gather and scatter operations are used to rearrange queried data into a contiguous memory segment and to store the vector of results back into the original track data, respectively.

4. Physics Validation

One of essential requirements of vectorized EM physics models is to verify the accuracy and precision of simulated physics results. Since implementations of EM physics models are designed to be architecture-independent, they can be executed in the same way for different backends, allowing direct validation of simulation results within statistical uncertainties due to the use of different random number generators - the vector and CUDA backend use random numbers generated by `Vc` and `CURAND` [16], respectively. To test the correctness of the algorithms'

implementation, we extended the validation by executing the same operations using the original Geant4 sampling method. Figure 1 shows kinematic distributions (energy and angle) of scattered photons of the Klein-Nishina model obtained by Geant4 algorithm and the vector backend using the alias sampling method. Based on Pearson's chi-squared test [17], kinematical distributions of simulated events using different sampling techniques are statistically validated for each model at selected monoenergetic inputs.

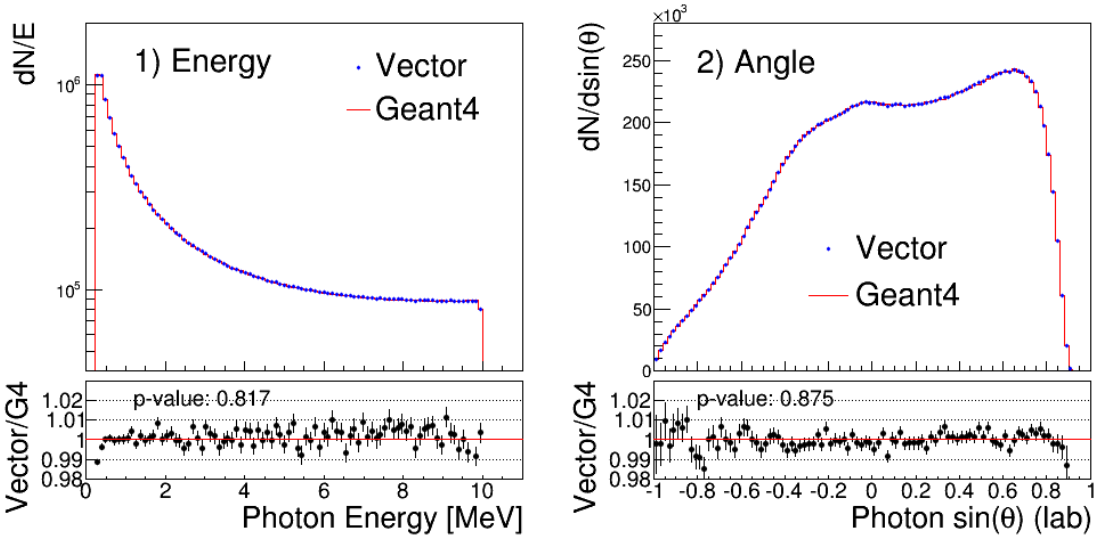

Figure 1. Examples of physics validation with kinematical distributions of simulated events: 1) Energy and 2) Angle of scatter photons obtained by the Klein-Nishina model (Geant4 vs. Vector) and their ratios and p-values of Pearson's chi-squared test with the monoenergetic input photon energy at 10 MeV.

5. Performance

To have efficient parallel code, performance analysis is a critical part of the development cycle. As the primary measure of the performance, we define the relative speedup as the ratio between the time taken by a model with a specific backend (Scalar, Vector, CUDA and etc) and by the Geant4 algorithm to execute the same task. For the purpose of performance measurements, particles are generated with a set of exponentially falling spectra within valid energy ranges for each model. Even though the relative speedup is not an absolute measure of the speedup, because the efficiency of sampling varies as a function of the energy, it can be used as a general guideline for performance comparisons, to identify potential problems, and to tune models optimized for a specific architecture. Figure 2 shows preliminary performance results of vectorized code tested on Intel® Xeon Phi (5110P) and CUDA code on NVidia GPU (Kepler K20M) for simulating interactions and sampling secondary particles using the alias method - the host used for performance evaluate is Intel® Xeon E5-2620 for both Xeon Phi and GPU.

6. Summary

As the GeantV project assembles the various pieces of the infrastructure: framework, geometry and physics, we demonstrated feasibility of implementing high-performance electromagnetic physics models for SIMD/SIMT architectures with common source code. Preliminary performance evaluation shows that parallelized physics code improves simulation speed using

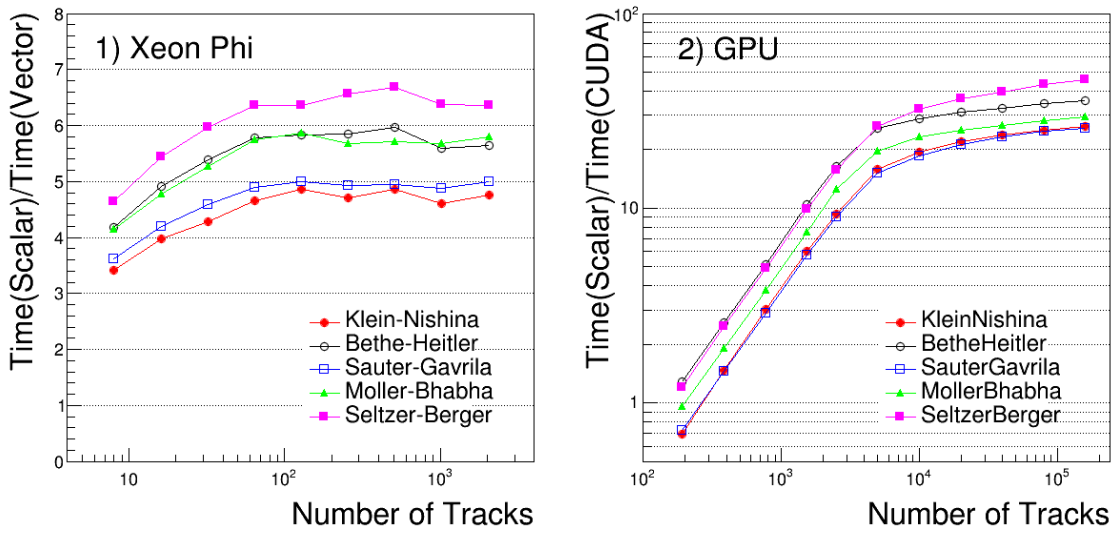

Figure 2. Relative speedup for simulating particles that undergo EM processes using alias sampling method: 1) the ratio between the CPU time taken by the vector code and by the scalar code on Intel Xeon Phi (5110P, one-thread), 2) the ratio between the time taken by the CUDA code on NVidia GPU (K20, 2496-cores) and by the scalar code on Intel Xeon (E5-2620, one-core). The energy range of input particles is [2,20] MeV where all models are valid.

the vector pipeline of SIMD and massively-parallel threads of SIMT. For example, as shown in this paper, vectorizing physics code improves simulation speed by a factor around 4-6 on Xeon Phi and around 20-50 on GPU depending on the size of tracks for different models. As integration of EM vector physics models into the GeantV framework is being carried out, computing performance will be re-evaluated to understand the impact of parallelization and to have insights for optimization.

References

[1] Habib S *et al* 2015 High energy physics forum for computing excellence *arXiv:1510.08545*
[2] The HEP Software Foundation `http://hepsoftwarefoundation.org`
[3] Apostolakis J *et al* 2014 A concurrent vector-based steering framework for particle transport *J. Phys.: Conf. Series* **523** 012004
[4] Agostinelli S *et al* 2003 Geant4 - A Simulation Toolkit *Nucl. Instrum. Methods Phys. Res.* A **506** 250-303
[5] Ahn S *et al* 2014 Geant4-MT: bringing multi-threading into Geant4 production *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA + MC 2013)* 04213
[6] Butcher J C and Messel H 1960 *Nucl. Phys.* **20** 15
[7] Messel H and Crawford D 1970 Electron-Photon shower distribution, Pergamon Press
[8] Ford R and Nelson W 1985 SLAC-265, UC-32
[9] Walker A J 1977 *ACM Trans. Math. Software.* **3** 253-256
[10] Brown F J, Martin W R, and Calahan D A 1981 *Trans. Am. Nucl. Soc.* **38** 354-355
[11] McCool M, Robinson A D, Reinders J, Structured parallel programing, ISBN-9780124159938, p. 187
[12] Kretz M and Lindenstruth V 2012 *Software: Practice and Experience* **42** 1409-1430
[13] UME::SIMD `https://bitbucket.org/edanor/umesimd`
[14] Wenzel S 2014 Towards a high performance geometry library for particle-detector simulation *16th International workshop on Advanced Computing and Analysis Techniques in physics research (ACAT)*
[15] de Fine Licht J 2014 First experience with portable high-performance geometry code on GPU *GPU Computing in High Energy Physics 2014*
[16] NVIDIA CUDA Toolkit 5.0 CURAND Guide `http://docs.nvidia.com/cuda/pdf/CURAND_Library.pdf`
[17] Pearson K 1900 *Philosophical Magazine Series 5* **50** 157175