First experience of vectorizing electromagnetic physics models for detector simulation

First experience of vectorizing electromagnetic physics models for detector simulation

G Amadio[1], J Apostolakis[2], M Bandieramonte[3], C Bianchini[1 8], G Bitzes[2], R Brun[2], P Canal[4], F Carminati[2], J de Fine Licht[5], L Duhem[6], D Elvira[4], A Gheata[2], S Y Jun[4], G Lima[4], M Novak[2], M Presbyterian7 O Shadura[2], R Seghal[7] and S Wenzel[2]

[1]Parallel Computing Center at São Paulo State University (UNESP), São Paulo, Brazil
[2]CERN, PH Department, CH-1211 Geneva, Switzerland
[3]Instituto Nazionale di Astrofisica, Osservatorio Astrofisico di Catania , Catania, Italy
[4]Fermilab, MS234, P.O. Box 500, Batavia, IL, 60510, USA
[5]University of Copenhagen, Copenhagen, Denmark
[6]Intel Corporation, Santa Clara, CA, USA
[7]Bhabha Atomic Research Centre (BARC), Mumbai, India
[8]Mackenzie Presbyterian University, São Paulo, Brazil

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. The GeantV vector prototype for detector simulations 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, parallelization needed to achieve optimal performance or memory access latency and speed. An additional challenge is to avoid the code duplication often inherent to supporting heterogeneous platforms. In this paper we present the first experience of vectorizing electromagnetic physics models developed for the GeantV project.

1. Introduction

Beginning in 2005, we ceased benefiting from automatic performance gains coming from increases in processor clock speed. Increases in the number of transistors on a chip now translate into an increase in the number of cores or vector units. In order to benefit from the full processing power of the latest chips, we must review our software architecture and increase the amount of work that can be done in parallel within a similar memory budget.

Within the High Energy Physics (HEP) software ecosystem, event simulation is one of the most time consuming parts of the work flow. However, the basis of HEP detector simulation is mostly independent from the details of individual experiments and thus easy to share among experiments. Consequently, any run-time performance improvement in physics event simulation can have significant impact on the amount and quality of HEP simulation overall.

Geant4 [1, 2, 3] is the most widely used simulation toolkit in contemporary HEP experiments, but does not efficiently utilize the vector capability of modern CPUs as it has been developed for sequential executions. The GeantV project [4] was launched in 2013 to research how to take advantage of modern processors and coprocessors in order to significantly increase run-time performance. The project studies performance gains from propagating multiple tracks from multiple events in parallel, improving code and data locality in the process. Using code specialized to take advantage of the hardware specifics, it aims to leverage both the vector pipelines in modern processors and the availability of coprocessors, including the Xeon Phi and general purpose GPUs. GeantV has three major components that need to be adapted for vectorization: scheduling, geometry and physics modeling. In this paper we focus on the techniques explored to enhance the existing electromagnetic (EM) physics models utilizing emerging parallel hardware architectures.

2. Vectorization of 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. Conventional HEP detector simulation processes all tracks of an event sequentially (i.e., one particle at a time) 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 in a vectorized manner 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 operations should be executed on multiple data. Therefore, vectorization of physics models requires proper algorithm decompositions in order to minimize conditional branch and to maximize instruction throughput. In this section, we briefly describe EM physics models and the alias method [5, 6] as an alternative sampling technique suitable for SIMD and SIMT architectures.

2.1. EM Physics Models

An essential component of particle interactions with matter is the final state analysis described by a physics model associated with the selected physics process for a given step; the physics process is sampled by the mean free path analysis beforehand. In most EM physics models, the atomic differential cross section of the underlying physics process plays a central role in updating kinematic states of the primary particle or producing secondary particles if necessary. For example, if the selected physics is the Compton process for a given step, the final state of the scattered photon (angle and energy) is sampled based on the two dimensional probability function of the Klein-Nishina differential cross section [7] for the given energy of the incoming photon. Then, a secondary particle, the recoil electron, is produced due to the inelastic scattering of the photon on atom.

In Geant4, combined composition and rejection methods [8, 9, 10] are often used to sample variables following probability distribution functions used in EM physics models. Inverse functions of their cumulative distributions are not generally analytically calculable. However, composition and rejection methods are not suitable for vectorization due to repeated conditional trials until a random draw is selected. One popular Monte Carlo technique to sample any discrete distribution that is effectively vectorizable is the alias method, described below.

Since most of the secondary particles produced by primary particle interactions with matter are electrons or photons in typical HEP events, physics models of electromagnetic (EM) interactions are the first candidates to be vectorized for GeantV. A list of EM physics processes and models for high energy electrons and photons that are considered for vectorization is summarized in Table 1 and current implementations of vector physics models are available from the GeantV repository [11].

Table 1. A list of electromagnetic physics processes and models of electron and photon processes that are considered for initial vectorization tests.

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^-
	Multiple Scattering	Goudsmit-Saunderson	–	e^-

2.2. Alias Sampling Method

The alias sampling method [5, 6] is similar to the acceptance-rejection (Von Neumann) 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. The alias table consists of the alias and the non-alias probability. Unlike composition and rejection methods, the alias method is effectively vectorized as each sampling procedure follows the same instructions without a branch or a conditional exit. It is also as accurate as the traditional table look-up method which is neither vectorizable nor efficient due to its use of a binary search. For GeantV vector physics models, we adopted the alias method for most of random samplings used in EM physics models, especially for secondary particle productions.

3. Implementation

GeantV has been designed from the start to enable the use of multiple modern hardware platforms. It is important for the long-term relevance of the project to preserve the option open to use existing platforms, as well as future hardware or software developments. In this section, we briefly describe the structure of code implementation for vector physics models.

3.1. Architecture Backends

GeantV uses backends, which are software layers between the generic, platform-independent simulation code and the hardware-specific details (e.g. SIMD vector registers or GPU threads) and their software-related constructs like SIMD intrinsics, MIC pragmas or CUDA `C++` extensions. Examples of currently available backends are the scalar, vector and cuda backends. A fourth backend, for the Intel MIC platform, is under intense development as part of two IPCC Intel projects [12, 13]. The vector backend uses the Vc library [14] to promote explicit SIMD vectorization by the client code in the kernels. Alternatively, both the cuda and scalar backends use standard types, since the GPU registers are scalar. Detailed descriptions of a backend may be found elsewhere [15, 16].

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. Each physics-relevant algorithm is coded into a separate kernel so we speak of algorithms and kernels interchangeably.

3.2. Generic kernels

Kernels are high-performance versions of performance-critical algorithms, developed using generic programming and based on the data structures defined 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.
- Virtual function calls inside the kernels are avoided. Kernels themselves are coded in terms of `C++` templates, with a specific backend as the template parameter. Platform-specific, high-performance kernels are built at compilation time, based on the generic kernels and the backends, selected by a user request or local hardware configuration.
- Branching of execution flow is strictly minimized. Kernels can only use conditional constructs sparingly, preferentially using constant-expression conditions known at compilation time. Modern compilers will completely remove unused sections of the code from resulting executable binaries.

Input data for the algorithms come as kernel arguments, triggering the compile-time instantiation of the binary objects appropriate for the hardware used.

4. Data Organization

The basic flow of data in vector physics models is a group of particles (a basket of tracks received from the GeantV scheduler or high level interfaces of physics processes) with similar properties processed by the map pattern. Map, a parallel control pattern, applies a function to every element of collected data in parallel or in a vectorized way. Each track contains a set of elements (data members) describing the state of the particle during the course of tracking. Position, momentum or energy of the track are frequently queried and updated throughout the physics process and should be laid out contiguously in memory for efficient vector or parallel operations. Therefore, the organization of track data is one of important considerations to achieve efficient memory accesses for both SIMD and SIMT.

In general, the Structure of Arrays (SOA) is more efficient than the Array of Structures (AoS) both for SIMD and SIMT [17]. Currently, interfaces to physics processes specialized only for the vector backend take track data in the SoA format. With SoA tracks, each vector kernel processes a set of vector instructions on coalesced chunks of SoA track elements for the number of iterations equivalent to the number of tracks within the basket divided by the corresponding vector size of SIMD instruction sets (2 for SSE, 4 for AVX, 8 for MIC for the double precision).

Another consideration for *effective* vectorization is related to the table lookup used in the *vectorized* sampling procedure with alias tables. Sampling using the alias technique involves scattered memory accesses to get the final state variable in parallel. For example, sampling outgoing photon energies in the Compton process using the alias table randomly selects target bins with values that usually are not contiguous in memory. For the vector backend, gather operations are used to rearrange queried data (scattered in memory) into a contiguous memory segment so that subsequent instructions can be executed through vector pipelines. The scatter operation is also required to store the vector of results back into the original track data. Since a gather operation itself is an additional sequential operation, it introduces an overhead in the performance of the sampling kernel.

5. Validation and Performance

As vectorized physics models adopt the alias sampling method and generic implementations for different architectures, it is critical to understand the computing performance of physics kernels and to validate results. Since physics kernels are designed to be architecture-independent, they can be executed in the exactly same way for different backends, allowing direct validation of simulation results within statistical uncertainties. Statistical fluctuations are due to the use of different random number generators: the vector and cuda backend use random numbers from `Vc` and `CURAND` (a CUDA library that provides efficient algorithms to generate pseudo–random numbers for a GPU) [18], respectively. To test the implementation of the alias algorithm, we extended the validation to execute the same operations using the original Geant4 library. Simulated quantities such as the final energy and angle of the primary track as well as kinematic distributions of secondary particles have been compared and validated with respect to results obtained by Geant4. Figure 1 shows some kinematic distributions of the Compton scattering as a validation example.

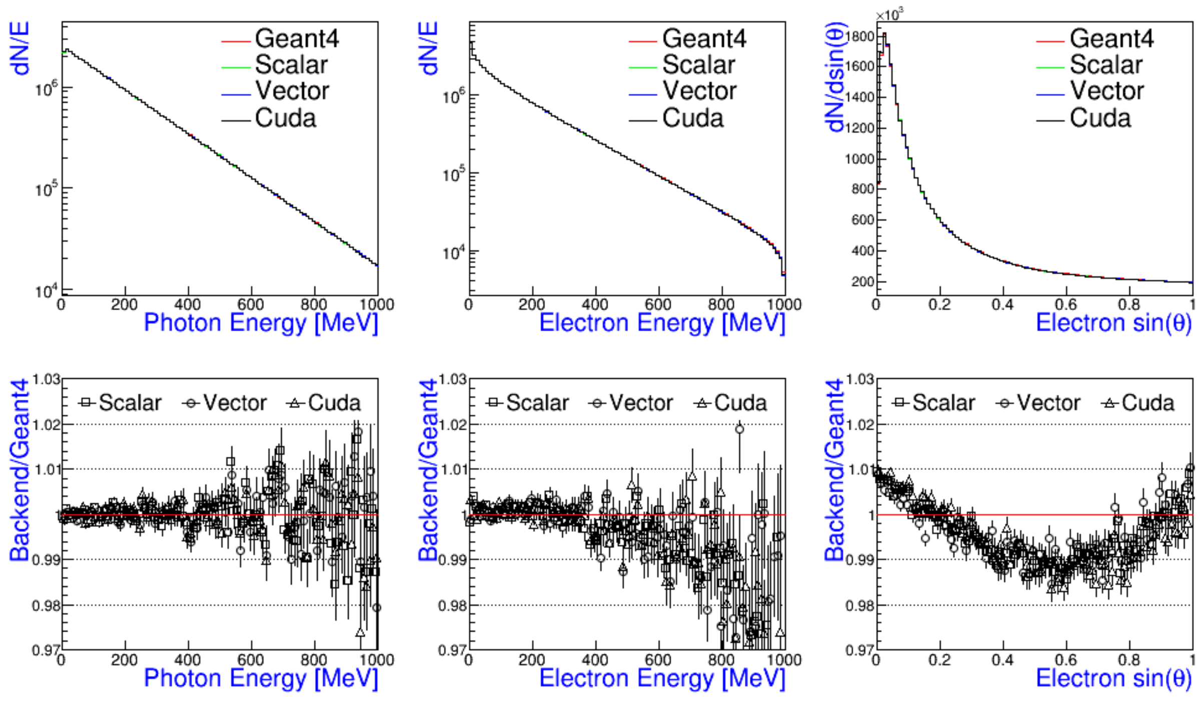

Figure 1. Validation of simulated events: 1) (top) distributions of input photons and secondary electrons of the Klein-Nishina model (Geant4, Scalar, Vector, Cuda), 2) (bottom) ratios of distributions obtained by different backends to the Geant4 distribution.

As a measure of performance, we define the relative speedup as the ratio between the time taken by the vector code and by the scalar code to execute the same task. Elapsed times on both the scalar and vector code were measured for simulating 100 trials with 4992 photons undergoing the Klein-Nishina process using the alias method. Secondary particles were sampled and stored on a temporary stack. Preliminary performance results are shown in Figure 2.

6. Conclusion

As the GeantV project assembles the various pieces of the infrastructure: scheduler, geometry and physics, we see tantalizing hints that the goal of significantly improving the performance of physics simulation applications is achievable. For example, as shown in this paper, vectorizing physics code improves simulation speed by a factor 2 to 5 depending on the hardware architecture. Further investigation is necessary to verify this result and implement the same type of improvements to other EM physics models. We must see if they can be similarly improved and if these improvements carry through when applied within a full GeantV simulation.

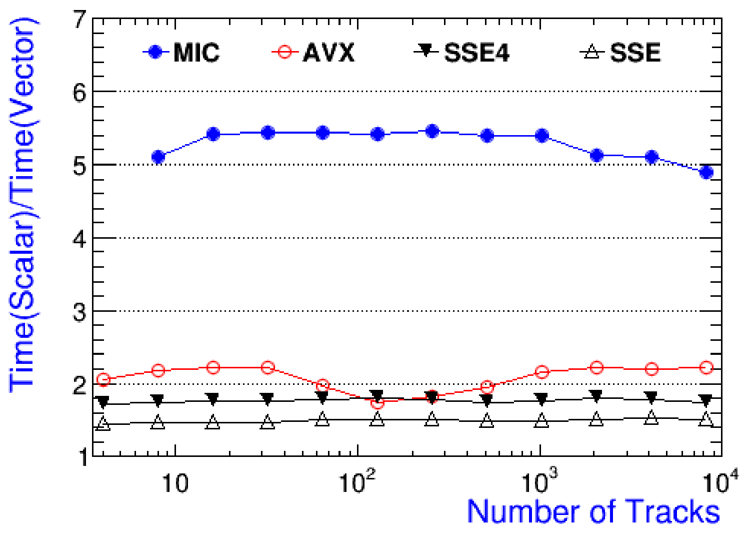

Figure 2. Relative speedup as the ratio between the CPU time taken by the vector code and by the scalar code to execute the same task. The average CPU time was measured for 100 trials for 4992 photons that undergo the Compton process using the Klein-Nishina model.

References

[1] Allison J *et al* 2006 Geant4 Developments and Applications *IEEE Trans. on Nucl. Sci.* **53** No. 1 270-278
[2] Agostinelli S *et al* 2003 Geant4 - A Simulation Toolkit *Nucl. Instrum. Methods Phys. Res.* A **506** 250-303
[3] 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
[4] Apostolakis J *et al* 2014 A concurrent vector-based steering framework for particle transport *J. Phys.: Conf. Series* **523** 012004
[5] Walker A J 1977 *ACM Trans. Math. Software. 3*, **3**, 253-256
[6] Brown F J, Martin W R, and Calahan D A 1981 *Trans. Am. Nucl. Soc.* **38** 354-355
[7] Klein O and Nishina Y Z 1929 *Physik* **52** 853
[8] Butcher J C and Messel H 1960 *Nucl. Phys.* **20** 15
[9] Messel H and Crawford D 1970 Electron-Photon shower distribution, Pergamon Press
[10] Ford R and Nelson W 1985 SLAC-265, UC-32
[11] http://git.cern.ch/pub/geant
[12] Intel® Parallel Computing Center at CERN `https://software.intel.com/en-us/articles/ipcc-at-cern-european-organisation-for-nuclear-research`
[13] Intel® Parallel Computing Center at São Paulo State University (Universidade Estadual Paulista, UNESP) `https://software.intel.com/en-us/articles/intel-parallel-computing-center-sao-paulo-state-university`
[14] Kretz M and Lindenstruth V 2011 Vc: A C++ library for explicit vectorization *Software: Practice and Experience.* Online at `http://dx.doi.org/10.1002/spe.1149`
[15] 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)*
[16] de Fine Licht J 2014 First experience with portable high-performance geometry code on GPU *GPU Computing in High Energy Physics 2014*
[17] Canal P *et al* 2013 High energy electromagnetic particle transportation on the GPU *Computing in High Energy Physics 2014*
[18] NVIDIA CUDA Toolkit 5.0 CURAND Guide `http://docs.nvidia.com/cuda/pdf/CURAND_Library.pdf`