Hindawi Publishing Corporation
Scientific Programming
Volume 2015, Article ID 980752, 10 pages
http://dx.doi.org/10.1155/2015/980752

Research Article

Performance Evaluation of Multithreaded Geant4 Simulations Using an Intel Xeon Phi Cluster

P. Schweitzer,[1,2,3] S. Cipière,[1,2,4] A. Dufaure,[1,4] H. Payno,[1,4] Y. Perrot,[1,4] D. R. C. Hill,[1,2,3] and L. Maigne[1,4]

[1] *Laboratoire de Physique Corpusculaire, Université Clermont Auvergne, Université Blaise Pascal, BP 10448, 63000 Clermont-Ferrand, France*
[2] *CNRS, UMR 6158, LIMOS, 63178 Clermont-Ferrand, France*
[3] *ISIMA, Institut Supérieur d'Informatique de Modélisation et de leurs Applications, BP 10125, 63178 Aubière, France*
[4] *CNRS, UMR 6533, LPC, 63178 Clermont-Ferrand, France*

Correspondence should be addressed to L. Maigne; maigne@clermont.in2p3.fr

Received 3 June 2015; Revised 29 September 2015; Accepted 8 October 2015

Academic Editor: Jan Weglarz

The objective of this study is to evaluate the performances of Intel Xeon Phi hardware accelerators for Geant4 simulations, especially for multithreaded applications. We present the complete methodology to guide users for the compilation of their Geant4 applications on Phi processors. Then, we propose series of benchmarks to compare the performance of Xeon CPUs and Phi processors for a Geant4 example dedicated to the simulation of electron dose point kernels, the TestEm12 example. First, we compare a distributed execution of a sequential version of the Geant4 example on both architectures before evaluating the multithreaded version of the Geant4 example. If Phi processors demonstrated their ability to accelerate computing time (till a factor 3.83) when distributing sequential Geant4 simulations, we do not reach the same level of speedup when considering the multithreaded version of the Geant4 example.

1. Introduction

Monte Carlo simulations have become an indispensable tool for radiation transport calculations in a great variety of applications. The Geant4 simulation toolkit [1] has come into widespread use in the field of high energy physics for simulating detectors in the Large Hadron Collider (LHC) experiments and also in the field of medical physics for diagnostic applications (e.g., analysis of the radiation burden to patients), for therapeutic applications (e.g., treatment planning in radionuclide therapy and treatment verification through emission scans), and for external beam therapy, especially in emerging areas such as proton and light-ion therapy [2–4]. The toolkit has demonstrated an attracting increasing interest because of its great versatility. It contains a comprehensive range of physics models for electromagnetic, hadronic, and optical interactions of a large set of particles over a wide energy range. It furthermore offers a diversity of tools for defining or importing the problem geometry, for modeling complex radiation sources and detection systems, including electromagnetic fields, electronic detector responses, and time-dependencies, and for exporting the required output data. The code is continuously being improved and extended with new functionalities. Since release 10.0, a multithreaded version of Geant4 provides the possibility to manage simulations on different threads at the event level. Through this new capacity and the market entry of novel Intel Xeon Phi hardware accelerators [5, 6], it was relevant to evaluate computing time performances of Geant4 10.0 with these new processors.

Some authors [7–11] already compared the performance of Intel Xeon Phi accelerators with GP-GPU (General Purpose Graphical Processing Unit) architectures; outcomes about these studies lead to convergent arguments on the necessity to initiate clever and optimized programming for Intel Phi in order to achieve performances very close to

GPUs. It has to be remarked that the portability of codes on GP-GPU is not always feasible; and in case it is, it demands greater code programming than Xeon Phi hardware accelerators. Effectively, regarding GP-GPU, highly parallel code sections have to be first identified in order to be fully rewritten using the CUDA language [12, 13]. For Xeon Phi hardware, simple rebuilding with correct memory usage is enough for using it. Bernaschi et al. [7] evaluated CPU Sandy Bridge, Kepler GPU, and Xeon Phi processors for a simulation in physics. In order to obtain good performances for their application, they highly tuned their application. For Xeon Phi hardware accelerators (5110P), they particularly worked on the memory management regarding the inputs and outputs used by their application and on the tuning of the C source code. When no changes are made on the application, they found that 1 GPGPU NVIDIA Kepler K20 is equivalent to 1 Phi and to 5 CPUs (Intel E5-2687W), while with a strong scaling (using MPI parallelization with asynchronous communication primitives to overlap data exchanges with the computations) 1 GPU is equivalent to 1.5 to 3.3 Phi. Lyakh [8] is more contrasted regarding this kind of comparisons by arguing on the difficulty to scale correctly an application to each architecture. Nevertheless, he obtained a steady x2-x3 speedup on GP-GPU architecture (Tesla K20X) compared to Phi (5110P). For Murano et al. [9], GPUs (NVIDIA Kepler K20) are constantly faster (between 12 and 27 times faster) than Xeon Phi hardware accelerators (5110P) without any tuning of the code (but using vectorization and adapted memory management). Other authors [14] experienced unsatisfactory performances when using Intel Xeon Phi accelerators for a medical imaging benchmark using a simple data structure and massive parallelism. In a recent study [15], we investigated performances of 7120P Intel Xeon Phi by distributing memory-bound Geant4 (version 9) simulation concerning the tracking of muons through a volcano in order to produce tomographic imaging. A maximum speedup of 2.56x was obtained when compared to calculations performed on a Sandy Bridge processor at 3.1 GHz.

In this paper, we describe very precisely the methodology followed for the compilation of Geant4 software and dependencies in the objective to guide any user willing to take part in the expected Xeon Phi computing potentiality for time-consuming simulations. Then, we propose as benchmarks a set of computing tests performed for the Geant4 advanced example "TestEm12" in the objective to conclude the suitability of Xeon Phi architecture for such simulations.

2. Portability of Geant4 Software on Intel Xeon Phi Coprocessor

2.1. Xeon Phi Cluster Characteristics. The simulations have been performed on a machine having two Intel Xeon CPUs E5-2690v2 (http://ark.intel.com/) (10 cores, 12 MB Cache, 3.001 GHz, Hyper Threading, 2 threads per core) each capable of a theoretical 240.1 GFLOPS of double precision floating point instructions with 59.7 GB/sec memory bandwidth, 128 GB of memory, and four Xeon Phi hardware accelerators 5110P (60 cores, 1.052 GHz, 4 hardware threads per core) each

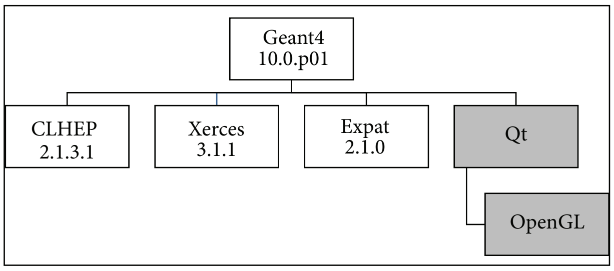

Figure 1: Dependencies concerning Geant4 10.0.p01 software. Libraries in grey are not used for our parallel benchmark.

capable of theoretical 1010.5 GFLOPS of double precision floating point instructions with 320 GB/sec memory bandwidth and 8 GB of memory. If we consider the announced performance in GFLOPS, the maximum speedup in favor of Xeon Phi should be 4.2x. The Turbo frequency (reaching 3.6 GHz) was kept inactivated for the reproducibility of the benchmark computing times. The version of the dedicated Linux distribution is 2.6.38.8+mpss3.4.1 for Xeon Phi specific architecture and 2.6.32-504.1.3.el6.x86_64 (CentOS 6.6) for Xeon CPU. In the whole paper, we simply refer to "Xeon" for Intel Xeon CPU and to "Phi" for Xeon Phi hardware accelerator.

2.2. Cross-Compilation of Geant4 Software and Dependencies. The present work was performed with version 10.0p01 of Geant4 [1] making use of CLHEP [16] libraries version 2.1.3.1, in order to handle pseudorandom numbers, numerics, points, vectors, planes, their transformations in 3D space, and Lorentz vectors and their transformations in simulations. Xerces 3.1.1 and Expat 2.1.0 libraries used to manage XML files are also necessary to run Geant4 correctly. The Qt library used for the OpenGL visualization has not been compiled since the visualization option is not recommended when using a parallel architecture for large computational tests (see Figure 1). Source codes were compiled with the Intel C++ Composer XE 2013 compiler set to O2 optimization level in order to avoid introducing uncontrolled bias when executing the codes compared to the O3 level like it is specified in the reference guide for Intel compilers (Quick-Reference Guide to Optimization with Intel Compilers version 12; retrieved from https://software.intel.com/sites/default/files/compiler_qrg12.pdf). The release of the CMake build system is version 2.8.

Each compilation process (native or cross-compilation) has to be run on Xeon (x86_64) processor using the Intel C Compiler (ICC) version 14.0.3 (compliant with GCC 4.8.1) in order to be later launched on Phi coprocessor architecture. For CMake compilation, the cross-compilation for Xeon Phi accelerators is activated when using the CMake flag "`-mmic`" for

(i) `DCMAKE_CXX_FLAGS`,
(ii) `DCMAKE_C_FLAGS`,`DCMAKE_EXE_LINKER_FLAGS`,
(iii) `DCMAKE_MODULE_LINKER_FLAGS`,
(iv) `DCMAKE_SHARED_LINKER_FLAGS` options.

```
#!/bin/sh
echo"Geant4"
PATH=/mic-install/geant4/geant4.10.0.p01install/bin/:$PATH
LDLIBRARYPATH=/mic-install/geant4/geant4.10.0.p01install/lib/:$LDLIBRARYPATH
G4LEDATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/data/G4EMLOW6.35
G4LEVELGAMMADATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/data/\
PhotonEvaporation3.0
G4NEUTRONXSDATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/data/\
G4NEUTRONXS1.4G4NEUTRONHPDATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/
data/G4NDL4.4
G4RADIOACTIVEDATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/data/\
RadioactiveDecay4.0
G4ABLADATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/data/G4ABLA3.0
G4PIIDATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/data/G4PII1.3
G4REALSURFACEDATA=/mic-install/geant4/geant4.10.0.p01install/share/Geant4-10.0/data/\
RealSurface1.0
G4INSTALL=/mic-install/geant4/geant4.10.0.p01/share/Geant4-10.0/geant4make
G4INCLUDE=/mic-install/geant4/geant4.10.0.p01/include/Geant4
G4LIBUSEGDML=1
G4LIB=/mic-install/geant4/geant4.10.0.p01/lib64/Geant4-10.0
G4LIBBUILDSHARED=1
G4WORKDIR=/home/user/geant4workdir
echo"CLHEP"
PATH=/mic-install/clhep/clhep-2.1.3.1install/bin/:$PATH
LDLIBRARYPATH=/mic-install/clhep/clhep-2.1.3.1install/lib/:$LDLIBRARYPATH
echo"XERCES"
PATH=/mic-install/xerces/xerces-c-3.1.1install/bin/:$PATH
LDLIBRARYPATH=/mic-install/xerces/xerces-c-3.1.1install/lib/:$LDLIBRARYPATH
echo"EXPAT"
PATH=/mic-install/expat/expat-2.1.0install/bin/:$PATH
LDLIBRARYPATH=/mic-install/expat/expat-2.1.0install/lib/:$LDLIBRARYPATH
echo"Intel"LDLIBRARYPATH=/mic-install/lib/mic/:$LDLIBRARYPATH
exportPATH=$PATH
exportLDLIBRARYPATH=$LDLIBRARYPATH
exportG4LEDATA=$G4LEDATA
exportG4LEVELGAMMADATA=$G4LEVELGAMMADATA
exportG4NEUTRONXSDATA=$G4NEUTRONXSDATA
exportG4NEUTRONHPDATA=$G4NEUTRONHPDATA
exportG4RADIOACTIVEDATA=$G4RADIOACTIVEDATA
exportG4ABLADATA=$G4ABLADATA
exportG4PIIDATA=$G4PIIDATA
exportG4REALSURFACEDATA=$G4REALSURFACEDATA
exportG4INSTALL=$G4INSTALL
exportG4INCLUDE=$G4INCLUDE
exportG4LIB=$G4LIB
```

Script 1: Environment variables defined in env.sh file before launching compilation process.

For libraries built using a configure script, the cross-compilation for Xeon Phi accelerators is activated when using the flag "`-mmic`" and when specifying that the x86_64 host machine is unknown in order to be sure that the compilation process does not refer to the current Linux version installed on the machine hosting the compiler.

2.2.1. Definition of Environment Variables. In order to compile Geant4 toolkit and dependencies, it is necessary to set specific environment variables. It is mandatory to append the corresponding libraries to the LD_LIBRARY_PATH variable and the executable binaries to the PATH variable. For our compilation, we created a `mic-install` repository where all the libraries were downloaded, built, and installed.

All variables starting their name with G4 are used for setting the Geant4 data libraries paths (photon evaporation data, radioactive decay data, particle cross sections for different energy ranges, cross section data for impact ionization, nuclear shell effects, optical surface reflectance, and nuclide state properties). It has to be noticed that as the bash and tcsh Unix shells are not supported on Phi coprocessor, environment variables have to be set using a basic sh script file; this file is then sourced using the command line "`source env.sh`" (Script 1).

```
cd xerces-c-3.1.1 build
CC=/opt/intel/bin/icc CFLAGS="-ansi -fp-model precise -mmic -static-intel" CXXFLAGS= \
"-ansi -fp-model precise -mmic \
-static-intel" CXX=/opt/intel/bin/icpc../xerces-c-3.1.1 src/configure \
--prefix=/mic-install/xerces/xerces-c-3.1.1 install/ --host=x86 64-unknown-linux-gnu
make -j40
make install
```

Script 2: Cross-compilation instructions to build xerces library using a configure script.

```
cd expat-2.1.0 build
CC=/opt/intel/bin/icc CFLAGS="-ansi -fp-model precise \
-mmic -static-intel"../expat-2.1.0 src/configure \
--prefix=/mic-install/expat/expat-2.1.0 install/ \
--host=x86 64-unknown-linux-gnu
make -j40
make install
```

Script 3: Cross-compilation instructions to build expat library using a configure script.

```
cd clhep-2.1.3.1 build
cmake \
-DCMAKE CXX COMPILER=/opt/intel/bin/icpc \
-DCMAKE CXX FLAGS="-ansi -fp-model precise -mmic -static-intel -gxx-name=g++-4.4" \
-DCMAKE C COMPILER=/opt/intel/bin/icc \
-DCMAKE C FLAGS="-ansi -fp-model precise -mmic -static-intel -gcc-name=gcc-4.4" \
-DCMAKE EXE LINKER FLAGS="-ansi -fp-model precise -mmic -static-intel" \
-DCMAKE INSTALL PREFIX=/mic-install/clhep/clhep-2.1.2.3 install \
-DCMAKE MODULE LINKER FLAGS="-ansi -fp-model precise -mmic -static-intel" \
-DCMAKE SHARED LINKER FLAGS="-ansi -fp-model precise -mmic -static-intel" \
../clhep-2.1.2.3 src/CLHEP
make -j40
make install
```

Script 4: CMake instructions for CLHEP library compilation.

2.2.2. Cross-Compilation of Geant4 Toolkit and Dependencies. The methodology followed for cross-compiling Geant4 toolkit and dependencies is inspired from a preliminary work [17] tested for a previous Geant4 version (9.6 p02) which was not developed for multithreaded execution. The methodology has been improved and lightened especially regarding the ROOT compilation, which is included in Geant4 since 10.0 release.

Xerces and Expat libraries have been compiled using the configuration instructions specified in Scripts 2 and 3.

Script 4 details the CMake instructions for the compilation of the CLHEP library.

Finally, Script 5 describes the CMake instructions for compiling Geant4 on Xeon Phi coprocessors.

It has to be remarked that we used the "`-fp-model precise`" flag making ICC compiler fulfill the IEEE 754 standard for floating point number representation and computation. This flag also prevents compiler optimizations that could introduce numerical errors according to the current floating point standard.

2.2.3. Compilation Test. The Geant4 extended example TestEm12 migrated to enable multithread computing (accessible at $G4INSTALL/examples/extended). To compile TestEm12 for Xeon Phi coprocessors, the listed CMake instructions have to be used (see Script 6).

This example, already validated by authors against other Monte Carlo codes [18], enables scoring the energy deposited per source particle in thin, concentric, spherical shells around an isotropic, monoenergetic, electron point source of 4 MeV (Mega Electron-Volt) centered in spherical geometry. The material of the sphere was chosen to be liquid water (density $1\,\mathrm{g{\cdot}cm^{-3}}$); the radius of the sphere was set to 3 cm and the number of shells was fixed to 150. The standard electromagnetic (EM) physics list option 3, describing electron and photon interactions between ~1 keV and 100 TeV, was used

```
cd geant4.10.0.p01_build
cmake \
-DGEANT4_USE_GDML=ON \
-DXERCESC_ROOT_DIR=/mic-install/xerces/xerces-c-3.1.1_install \
-DXERCESC_LIBRARY=/mic-install/xerces/xerces-c-3.1.1_install/lib/libxerces-c.so \
-DXERCESC_INCLUDE_DIR=/mic-install/xerces/xerces-c-3.1.1_install/include \
-DCLHEP_ROOT_DIR=/mic-install/clhep/clhep-2.1.2.3_install \
-DGEANT4_USE_QT=OFF \
-DEXPAT_INCLUDE_DIR=/mic-install/expat/expat-2.1.0_install/include \
-DEXPAT_LIBRARY=/mic-install/expat/expat-2.1.0_install/lib/libexpat.so \
-DGEANT4_INSTALL_DATA=ON \
-DGEANT4_BUILD_EXAMPLES=ON \
-DGEANT4_BUILD_MULTITHREADED=ON \
-DCMAKE_INSTALL_PREFIX="/mic-install/geant4/geant4.10.00.p01_install" \
-DCMAKE_C_COMPILER="/opt/intel/bin/icc" \
-DCMAKE_CXX_COMPILER="/opt/intel/bin/icpc" \
-DCMAKE_C_FLAGS="-ansi -fp-model precise -mmic" \
-DCMAKE_CXX_FLAGS="-ansi -fp-model precise -mmic" \
-DCMAKE_SHARED_LINKER_FLAGS="-ansi -fp-model precise -mmic -Wl, \
-rpath,$LD_LIBRARY_PATH -Wl,-rpath-link,$LD_LIBRARY_PATH"\
-DCMAKE_EXE_LINKER_FLAGS="-ansi -fp-model precise -mmic -Wl, \
-rpath,$LD_LIBRARY_PATH -Wl,-rpath-link,$LD_LIBRARY_PATH"\
-DWITH_ANALYSIS_USE=ON \
-DCMAKE_VERBOSE_MAKEFILE=OFF -Wno-dev \
../geant4.10.0.p01_src
make -j40
make install
```

Script 5: CMake instructions for Geant4 software compilation.

```
cd TestEm12MT_build
cmake \
-DCMAKE_CXX_COMPILER=/opt/intel/bin/icpc \
-DCMAKE_CXX_FLAGS="-ansi -fp-model precise -mmic -static-intel -gxx-name=g++-4.4" \
-DCMAKE_C_COMPILER=/opt/intel/bin/icc \
-DCMAKE_C_FLAGS="-ansi -fp-model precise -mmic -static-intel -gcc-name=gcc-4.4" \
-DCMAKE_EXE_LINKER_FLAGS="-ansi -fp-model precise -mmic -static-intel"../TestEm12MT
make -j40
```

Script 6: CMake instructions for compiling Geant4 TestEm12 example.

for all simulations, taking into account electron impact ionization, multiple scattering, and Bremsstrahlung generation. In Script 7 is presented a TestEm12 macro file example. When using the multithreaded mode, the command line "`/run/numberOfThreads 10`" has to be added with the corresponding number of threads (in this example: 10 threads). The macro file is launched on coprocessors with the executable generated after the compilation using "`./exe TestEm12.mac`".

In order to verify the correct cross-compilation of Geant4 and dependencies, we tested the reproducibility of results for TestEm12 electromagnetic example on 1 Xeon and its multithreaded version TestEm12MT running on 40 Xeon threads and 240 Phi threads using or not the optimized compilation flag "`-fp-model precise`". Figure 2 represents the energy deposited per source particle for the four test cases. Each test case has been repeated 5 times. As it can be noticed, if the compilation flag "`-fp-model precise`" is omitted during the compilation procedure of Geant4 and CLHEP, then the energy deposition profile is shifted to significantly smaller radial distances which leads to a bad agreement with other configurations. If we quantify the difference between energy (E_A) at distance r calculated with Xeon CPU and the energy (E_B) at the same distance calculated with Phi without applying any optimized compilation process, as a percentage of the maximum value $\max(E_A; E_B)$ of the two calculated energy distributions, then differences between $r/r_0 = 0$ and $r/r_0 = 2.0$ up to 40% are found when a threshold less than 3% is usually accepted. No other optimized compilation flags were tested as we achieved a very good agreement with "`-fp-model precise`" flag.

```
/control/verbose 0
/run/verbose 0#
/testem/det/setMat G4_WATER
/testem/det/setRadius  3 cm#
/testem/phys/addPhysics  emstandard_opt3    # em physics
/run/numberOfThreads 10
/run/initialize
/gun/particle e-
/gun/energy 4 MeV
/analysis/setFileName test
/analysis/h1/set 1 150 0.  3.  cm       #edep profile
/analysis/h1/set 3 100 0.   3.  cm      #true track length
/analysis/h1/set 8 120 0.   1.2  none   #normalized edep profile
/testem/applyAutomaticStepMax true
/run/beamOn 1000000
```

Script 7: TestEm12 macro file example (TestEm12.mac).

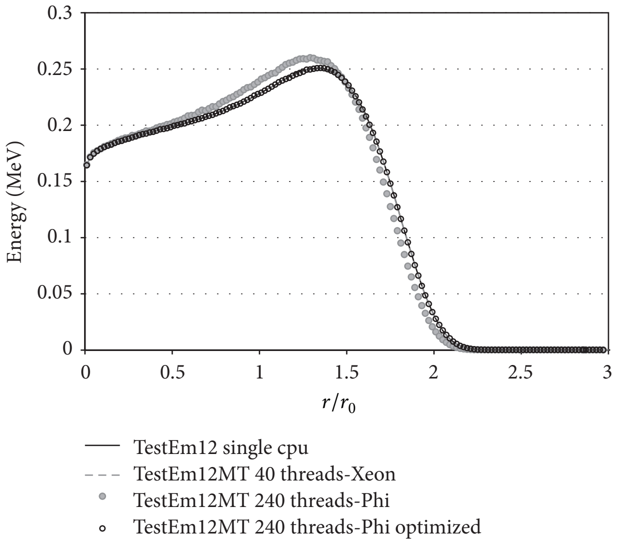

Figure 2: Comparison of energy depositions for monoenergetic electrons with initial energy of 4 MeV using TestEm12 Geant4 example calculated using a single Xeon CPU (black line), the multithreaded TestEm12 Geant4 example calculated using 40 Xeon threads (grey dashed line), using 240 Phi threads (grey circles), and using 240 threads on a Phi accelerator with an optimized compilation process (black circles). A perfect agreement is noticed between the example of reference on a single CPU and the optimized compilation process on Phi accelerator.

3. Performance Evaluations

3.1. Benchmark Descriptions. Geant4 simulations, through TestEm12 extended example, have been tested on Xeon Phi accelerators using distributed (TestEm12) and multithreaded (TestEm12MT) modes. Prior to any computational tests, we profiled TestEm12 example using the Intel VTune toolkit in order to quantify memory bandwidth consumption. We could conclude that the simulation is highly compute-bound.

In this study, we consider that the "distributed" mode means launching several independent simulation instances at the same time without involving any communication between instances. Concerning the "multithreaded" mode, simulations are launched using the pthread library. In both cases, simulations are balanced regarding the number of particles equally spread among worker nodes.

For the distributed mode, the total number of particles is split between the multiple instances of runs like described by authors in [19, 20]. The postprocessing time due to the merging of results has been evaluated and represents a negligible percentage (less than 0.5%) of the total execution time measured.

Table 1: Description of number of partitions (or threads) used for benchmarking on Xeon and Phi coprocessors.

Number of partitions	Xeon			Phi						
TestEm12 example	1	10	20		1		60		240	960
TestEm12MT example	10	20	40	10	20	40	60	120	240	960

Table 1 is listing the number of partitions used for speedup calculations. For each benchmark, simulations were repeated five times; mean time values have been used for all the results. Standard deviations for all recovered time values do not overlap 1%. The standard EM option 3 used 4 MeV electron point sources. All times are wall clock times, measured by the internal clock of the host machine. Distributed simulations used the Mersenne Twister pseudorandom number generator (PRNG). The pseudorandom numbers were generated using a sequence splitting method ensuring a sufficiently large sequence of 3.10^{10} nonoverlapping random numbers in order to not reproduce any particle event. When using the multithreaded version of Geant4, nothing was modified to the process of partitioning; the Mersenne Twister PRNG was used.

The Xeon Phi hardware was used with a "native mode," meaning whole simulations were executed on the Xeon Phi or directly started on the Xeon Phi using SSH.

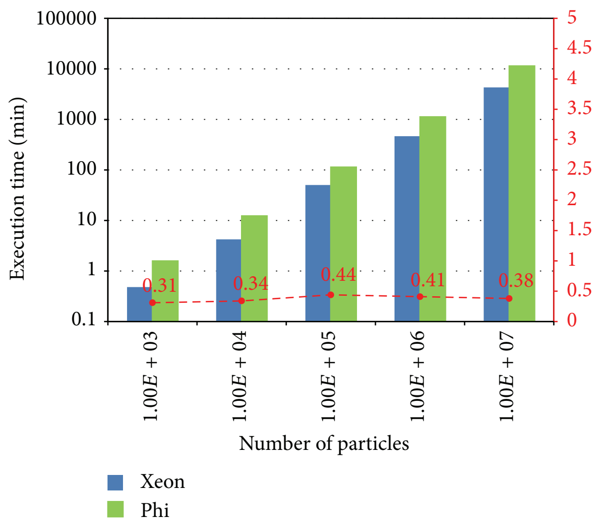

Figure 3: Computing time in minutes of TestEm12 Geant4 example running on 1 Xeon thread (blue) and 1 Phi thread (green) coprocessors for generated source particles going from 10^3 to 10^7. Speedup (red) is indicated for each number of particles.

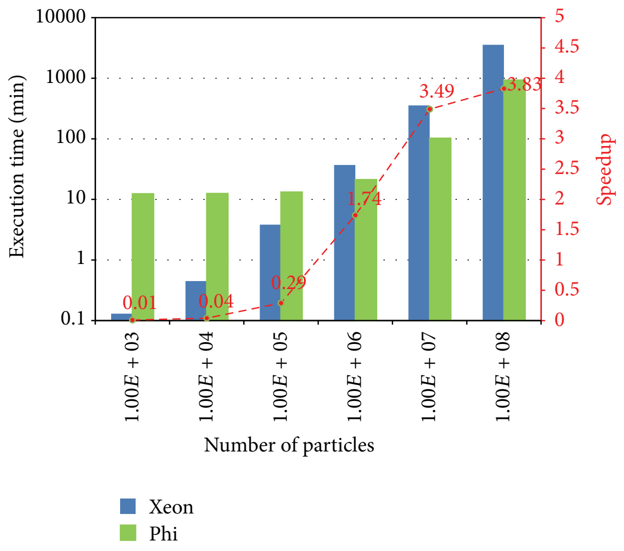

Figure 5: Computing time in minutes of TestEm12 Geant4 example running on 20 Xeon (blue) threads and 240 Phi (green) threads for generated source particles going from 10^3 to 10^8. Speedup (red) is indicated for each number of particles.

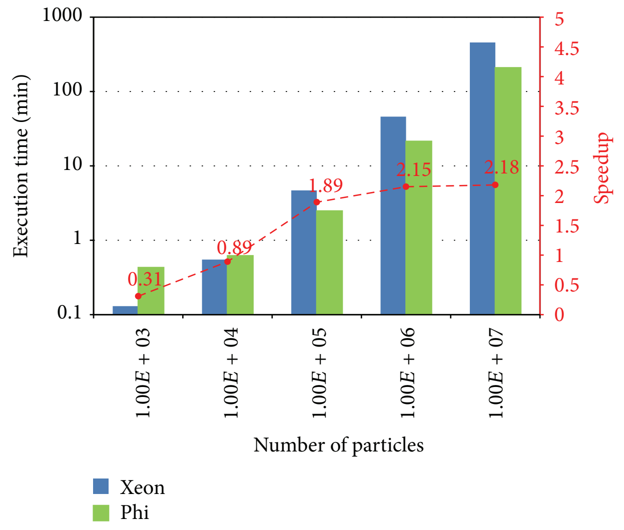

Figure 4: Computing time in minutes of TestEm12 Geant4 example distributed on 10 Xeon (blue) cores and 60 Phi (green) cores for generated source particles going from 10^3 to 10^7. Speedup (red) is indicated for each number of particles.

3.2. Xeon versus Phi for Distributed Geant4 Simulations (TestEm12). Speedup was evaluated for distributed TestEm12 simulations for generated source particles going from 10^3 to 10^8. Figure 3 presents execution times in minutes obtained for one Xeon thread and one Phi thread as the speedup reached. It can be observed that the speedup keeps constant (~0.38) whatever the number of particles selected. This result was expected since the ratio of intrinsic frequencies between Xeon and Phi is about 3 (3.001 GHz for Xeon compared to 1.052 GHz for Phi).

In order to verify the Intel claimed performances for Xeon and Phi [5], we represented on Figure 4 execution times in minutes obtained for 10 Xeon versus 60 Phi hardware cores. It can be observed that when using a higher number of particles (10^7), we reach a speedup of 2.18. For this benchmark, it appears that Phi coprocessors are more suitable than Xeon coprocessors for number of particles higher than 10^5.

When using the total amount of threads available on 1 Xeon and 1 Phi, respectively, 20 and 240 threads (see Figure 5), we reach a speedup of 3.83 for 10^8 particles, which is close to the expected Intel maximum speedup of 4.2, when comparing the announced GFLOPS performances. Since we have selected limited optimization and a floating point precision flag, this result is very satisfying.

3.3. Distributed (TestEm12) versus Multithreaded (TestEm12MT) Geant4 Simulations on Xeon Processors. Speedup was evaluated for three different numbers of threads: 10, 20, and 40 corresponding, respectively, to the number of hardware cores for one Xeon CPU, the number of hardware cores for 2 Xeon CPUs, and the number of threads for 2 Xeon CPUs. The goal was to evaluate the potential impact of using a multithreaded version compared to a distributed one on Xeon processors. This case study presented on Figure 6 demonstrates that TestEm12MT, the multithreaded version of TestEm12 Geant4 example, is slower than TestEm12 (speedup between 1.08 and 1.33) whatever the number of threads used.

3.4. Impact of the Number of Phi Threads for Multithreaded (TestEm12MT) Geant4 Simulations. In the objective to evaluate if a high number of threads reduces significantly the execution time of TestEm12MT on Phi whatever the number of particles generated, we plotted on Figure 7 the different computing times obtained for generated source particles going from 10^5 to 10^8 and a number of Phi threads going from 10 to 240.

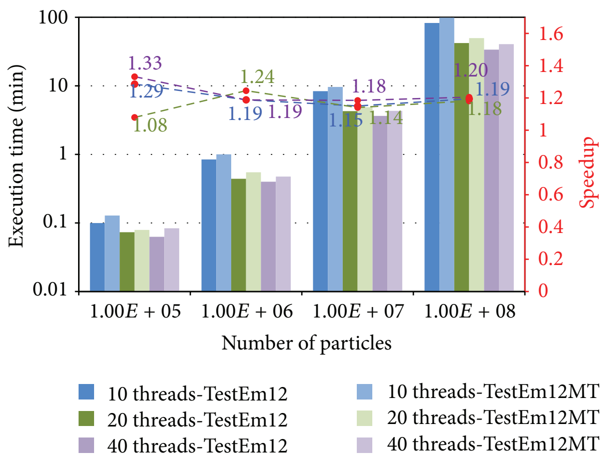

FIGURE 6: Comparison of computing time in minutes of TestEm12 and TestEm12MT Geant4 examples running on 10 (blue), 20 (green), and 40 (purple) Xeon threads for generated source particles going from 10^5 to 10^8. Speedup (red) is indicated for each number of particles and the three test cases.

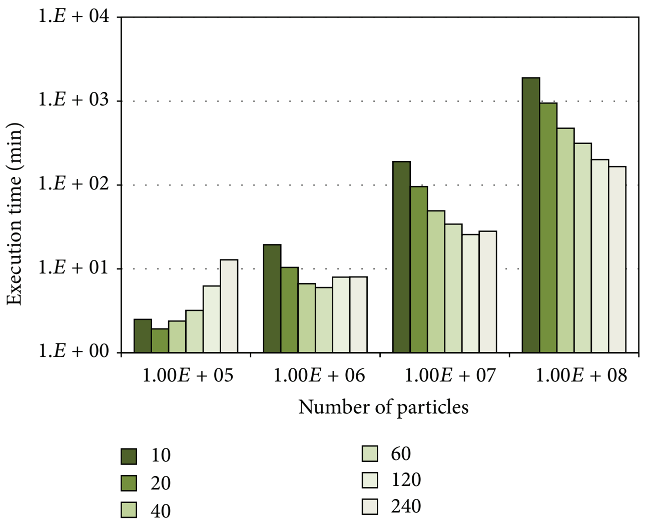

FIGURE 7: Comparison of computing times in minutes of TestEm12MT Geant4 example running on 10, 20, 40, 60, 120, and 240 Phi threads for generated source particles going from 10^5 to 10^8.

We can remark that the higher the number of generated source particles is, the higher the number of threads must be to reduce the execution times. For 10^8 particles, we obtain an almost linear diminution of computing time with the number of threads (till 60 threads), as it is also shown in Table 2 representing the speedups obtained for 10, 20, 40, 60, 120, and 240 Phi threads for 10^8 particles compared to one Phi thread. For source particles inferior to 10^5, the initialization process to fix physics datasets and geometry is of the same order of duration as the emission and tracking of particles which explains that in this case we obtain the best computing time for a lower number of threads (20 threads).

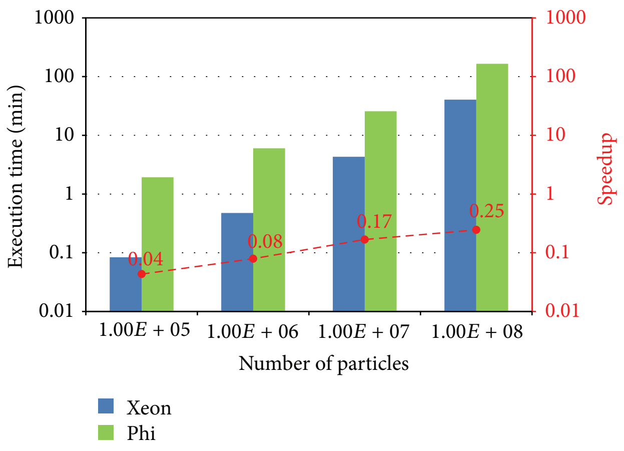

FIGURE 8: Computing time in minutes of TestEm12MT Geant4 example running on 40 Xeon threads (blue) and the best computing time obtained with Phi threads (green) for generated source particles going from 10^5 to 10^8. Speedup (red) is indicated for each number of particles.

TABLE 2: Speedups obtain for 10, 20, 40, 60, 120, and 240 Phi threads for 10^8 particles compared to 1 Phi thread.

Number of threads	10	20	40	60	120	240
Speedup	10.00	19.96	39.83	60.11	94.02	114.20

TABLE 3: Computing time obtained for 10^8 and 10^9 particles on 40 Xeon threads and 960 Phi threads.

Number of particles	Computing time (mins)	
	40 Xeon threads	960 Phi threads
10^8	40.6	49.9
10^9	407.5	390.4

3.5. Xeon versus Phi for Multithreaded (TestEm12MT) Geant4 Simulations. The speedup was evaluated for TestEm12MT simulations using the standard EM physics list for generated source particles going from 10^5 to 10^8 running on 40 Xeon threads and the best computing time obtained for Phi threads. It can be observed that whatever the number of particles generated, Phi provides longer execution times (see Figure 8). The speedup is nevertheless increasing in function of the number of particles generated to reach 0.25 for 10^8 particles.

In Table 3, we listed the computing time obtained on 40 Xeon threads and 960 Phi threads for 10^8 and 10^9 particles.

When using 960 Phi threads, the computing time reaches 49.9 minutes for 10^8 particles, which is 23% higher than when using 40 Xeon threads (computing time corresponding to 40.6 minutes). But when reaching 10^9 particles the computing time is finally reduced on 960 Phi threads compared to Xeon; for this last configuration, we obtain a speedup of 1.04.

4. Conclusion

The objective of this paper was to first detail a clear and understandable methodology to compile and execute any Geant4 application on Xeon Phi accelerators. Special attention should be paid for using the optimization compilation flag "`-fp-model precise`" in order to obtain identical results compared to an execution on CPUs.

Then, the ambition of authors was to evaluate the performance of Xeon Phi accelerators for such applications especially due to the availability of the multithreaded version of the Geant4 toolkit. We have to remind the reader that, in a first instance, no tuning of the source code has been initiated in this study. Regarding the different outcomes obtained, we may conclude that

(i) when distributing sequential Geant4 simulations (40 Xeon threads compared to 240 Phi threads), Phi (5110P at 1 GHZ) are faster than Xeon CPUs (E5-2690v2 at 3 GHZ), almost reaching the maximum speedup (3.83x versus 4.2x) though limited optimization was considered to save the precision of the final results;

(ii) when considering multithreaded Geant4 simulations on Xeon CPUs, we can remark that this version is unfortunately slightly slower than the classical distribution of the sequential Geant4 simulations whatever the number of threads used;

(iii) even if we observe a loss of performance for the multithreaded version of Geant4 on Phi compared to Xeon CPUs, it has to be noticed that, using a high number of particles in simulations (corresponding to more than 6 hours of computing on 40 Xeon CPUs for 10^9 particles), we finally reach a very tiny speedup of 1.04 using 960 Phi threads.

For the moment, we can state that the multithreaded version of Geant4 is not yet optimized to compete with a distributed submission of simulations on a farm of CPU clusters, on a cluster of Phi hardware accelerators, or on a grid infrastructure. It would certainly necessitate tuning drastically the source code and suppressing any verbose display, in order to make such applications fully compliant with Xeon Phi architectures. One would expect that the next generation of the Geant toolkit (Geant5) would answer such problematic.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was carried out as part of the Laboratory of Excellence ClerVolc project. The authors wish to thank the Geant4 collaboration for its technical support.

References

[1] S. Agostinelli, J. Allison, K. Amako et al., "Geant4—a simulation toolkit," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 506, no. 3, pp. 250–303, 2003.

[2] S. Jan, G. Santin, D. Strul et al., "GATE—Geant4 application for tomographic emission: a simulation toolkit for PET and SPECT," *Physics in Medicine and Biology*, vol. 49, no. 19, pp. 4543–4561, 2004, http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3267383/?tool=pmcentrez&report=abstract.

[3] S. Jan, D. Benoit, E. Becheva et al., "GATE V6: a major enhancement of the GATE simulation platform enabling modelling of CT and radiotherapy," *Physics in Medicine and Biology*, vol. 56, no. 4, pp. 881–901, 2011.

[4] D. Sarrut, M. Bardiès, N. Boussion et al., "A review of the use and potential of the GATE Monte Carlo simulation code for radiation therapy and dosimetry applications," *Medical Physics*, vol. 41, no. 6, Article ID 064301, 2014.

[5] J. Jeffers and J. Reinders, "Offload," in *Intel Xeon Phi Coprocessor High Performance Programming*, chapter 7, pp. 189–241, Morgan Kaufmann Publishers, Elsevier, 2013.

[6] A. Vajda, *Programming Many-Core Chips*, Springer, New York, NY, USA, 1st edition, 2011.

[7] M. Bernaschi, M. Bisson, and F. Salvadore, "Multi-Kepler GPU vs. multi-Intel MIC for spin systems simulations," *Computer Physics Communications*, vol. 185, no. 10, pp. 2495–2503, 2014.

[8] D. I. Lyakh, "An efficient tensor transpose algorithm for multicore CPU, Intel Xeon Phi, and NVidia Tesla GPU," *Computer Physics Communications*, vol. 189, pp. 84–91, 2015.

[9] K. Murano, T. Shimobaba, A. Sugiyama et al., "Fast computation of computer-generated hologram using Xeon Phi coprocessor," *Computer Physics Communications*, vol. 185, no. 10, pp. 2742–2757, 2014.

[10] G. Crimi, F. Mantovani, M. Pivanti, S. F. Schifano, and R. Tripiccione, "Early experience on porting and running a Lattice Boltzmann code on the Xeon-Phi co-processor," *Procedia Computer Science*, vol. 18, pp. 551–560, 2013.

[11] J. Dokulil, E. Bajrovic, S. Benkner, S. Pllanaa, M. Sandrieser, and B. Bachmayer, "High-level support for hybrid parallel execution of C++ applications targeting Intel Xeon Phi coprocessors," *Procedia Computer Science*, vol. 18, pp. 2508–2511, 2013.

[12] H. Perez-Ponce, Z. El Bitar, Y. Boursier et al., "Implementing Geant4 on GPU for medical applications," in *Proceedings of the IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC '11)*, pp. 2703–2707, IEEE, Valencia, España, October 2011.

[13] L. Jahnke, J. Fleckenstein, F. Wenz, and J. Hesser, "GMC: a GPU implementation of a Monte Carlo dose calculation based on Geant4," *Physics in Medicine and Biology*, vol. 57, no. 5, pp. 1217–1229, 2012.

[14] J. Fang, H. Sips, C. Xu, Y. Che, L. Zhang, and A. L. Varbanescu, "Test-driving intel xeon phi," in *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering (ICPE '14)*, pp. 137–148, ACM, Dublin, Ireland, March 2014.

[15] P. Schweitzer, C. Mazel, C. Carloganu, and D. R. Hill, "Performance analysis with a memory-bound Monte Carlo simulation on Xeon Phi," in *Proceedings of the IEEE International Conference on High Performance Computing & Simulation (HPCS '15)*, pp. 444–452, Amsterdam, The Netherlands, July 2015.

[16] L. Lönnblad, "CLHEP—a project for designing a C++ class library for high energy physics," *Computer Physics Communications*, vol. 84, no. 1–3, pp. 307–316, 1994.

[17] P. Schweitzer, C. Mazel, C. Cârloganu, and D. Hill, "A method for porting HEP software Geant4 and ROOT to Intel Xeon Phi hardware accelerator," in *Proceedings of the European Simulation and Modelling Conference*, pp. 73–79, ACM, Porto, Portugal, October 2014.

[18] L. Maigne, Y. Perrot, D. R. Schaart, D. Donnarieix, and V. Breton, "Comparison of GATE/GEANT4 with EGSnrc and MCNP for electron dose calculations at energies between 15 keV and 20 MeV," *Physics in Medicine and Biology*, vol. 56, no. 3, pp. 811–827, 2011.

[19] L. Maigne, D. Hill, P. Calvat et al., "Parallelization of Monte Carlo simulations and submission to a grid environment," *Parallel Processing Letters*, vol. 14, no. 2, pp. 177–196, 2004.

[20] D. R. C. Hill, C. Mazel, J. Passerat-Palmbach, and M. K. Traore, "Distribution of random streams for simulation practitioners," *Concurrency Computation Practice and Experience*, vol. 25, no. 10, pp. 1427–1442, 2013.