The GeantV project: preparing the future of simulation

2015 J. Phys.: Conf. Ser. 664 072006

(http://iopscience.iop.org/1742-6596/664/7/072006)

Journal of Physics: Conference Series **664** (2015) 072006 doi:10.1088/1742-6596/664/7/072006

The GeantV project: preparing the future of simulation

G Amadio[1], J Apostolakis[2], M Bandieramonte[3], A Bhattacharyya[4], C Bianchini[1,8], R Brun[2], Ph Canal[5], F Carminati[2,9], L Duhem[6], D Elvira[5], J de Fine Licht[7], A Gheata[2], R L Iope[1], G Lima[5], A Mohanty[4], T Nikitina[2], M Novak[2], W Pokorski[2], R Seghal[4], O Shadura[2], S Vallecorsa[2], S Wenzel[2]

[1] Sao Paulo State University, Sao Paulo, Brazil
[2] CERN, Geneva, Switzerland
[3] Istituto Nazionale di Astrofisica, Osservatorio Astrofisico di Catania (INAF-OACT), Catania, Italy
[4] Bhabha Atomic Research Centre (BARC), Mumbai, India
[5] Fermi National Accelerator Laboratory, Batavia, IL, USA
[6] Intel Corporation, Santa Clara, CA, USA
[7] University of Copenhagen, Copenhagen, Denmark
[8] Mackenzie Presbyterian University, Sao Paulo, Brazil

[9]E-mail: Federico.Carminati@cern.ch

Abstract. Detector simulation is consuming at least half of the HEP computing cycles, and even so, experiments have to take hard decisions on what to simulate, as their needs greatly surpass the availability of computing resources. New experiments still in the design phase such as FCC, CLIC and ILC as well as upgraded versions of the existing LHC detectors will push further the simulation requirements. Since the increase in computing resources is not likely to keep pace with our needs, it is therefore necessary to explore innovative ways of speeding up simulation in order to sustain the progress of High Energy Physics. The GeantV project aims at developing a high performance detector simulation system integrating *fast* and *full* simulation that can be ported on different computing architectures, including CPU accelerators. After more than two years of R&D the project has produced a prototype capable of transporting particles in complex geometries exploiting micro-parallelism, SIMD and multithreading. Portability is obtained via C++ template techniques that allow the development of machine-independent computational kernels. A set of tables derived from Geant4 for cross sections and final states provides a realistic shower development and, having been ported into a Geant4 physics list, can be used as a basis for a direct performance comparison.

1. Introduction

High Energy Physics needs ever-increasing computing power. Apart from securing a large amount of computing resources, the other obvious element to leverage to face a possible scarcity of computing resources is an aggressive optimisation of the code. Several factors conspired to have this aspect somewhat neglected until now. For at least two decades, from the mid eighties to about 2005, the CPU clock frequency increased steadily. This, together with other CPU architectural improvements more

[9] To whom any correspondence should be addressed.

than doubled the code performance every eighteen months at a constant price and with no programming cost. The amount of computing provided from 2001 onward by different Grid projects made these years a "happy time" for HEP computing, and there was no shortage of CPU cycles to worry about. Moreover most of the attention of the HEP developers was absorbed by the development of experiment specific software infrastructures of unprecedented size and complexity.
The fact that since 2006 the clock frequency stopped increasing and that extra computing capacity was provided increasingly in terms of parallelism raised little concern in the face of other priorities at the approach of data taking. As a result HEP code fell progressively behind in the utilisation of the potential (parallel) performance of the new CPUs. An alarming report by CERN openlab appeared in 2012 [1] announcing that HEP code could be using no more than 5% of the peak hardware speed. Some of the assumptions of the paper are debatable, as well as the 5% figure, however the message that HEP code was not taking advantage from the latest advances in CPU technology was essentially correct.
All the above clearly points to the fact that, barring another computing "happy time", it is high time to "bite the bullet" and invest the effort necessary to evolve our code to better exploit the full potential performance offered by new CPUs.

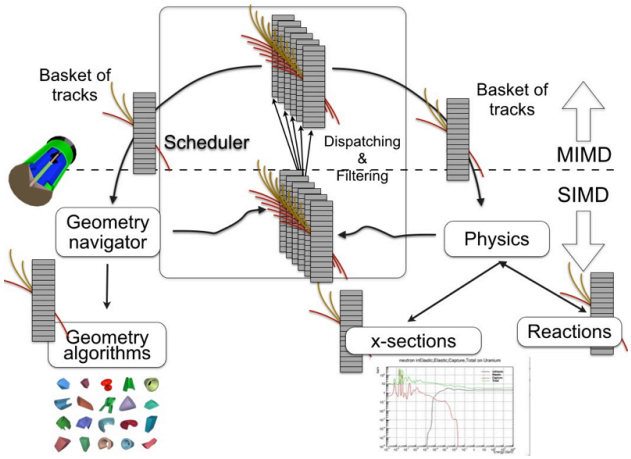


Figure 1. GeantV general architecture.

2. The GeantV project

HEP software is a typical example of a field in need of high performance computation, but with very little linear algebra, or high floating point content, the exploitation of emerging hardware architectures tends to be very challenging. On the other hand this offers an interesting window of opportunity, as HPC providers are of course interested in enlarging the family of target applications and we can offer a very large real-life example outside the usual realm of HPC with open-source code and expert users and developers. One clear proof of the interest of industry is the on-going collaboration between GeantV, Intel and CERN openlab in the form of two IPCC Intel projects [2, 3] thanks to which a large share of the results presented in this paper have been achieved.

2.1. A new simulation code

Optimisation of HEP code is both urgent and difficult to achieve. The total amount of active HEP code is probably approaching some 10^8 lines of code and its development and life-cycle management processes are very diverse and geographically distributed. Much of this code is experiment-specific as there is comparatively little reuse of code between different experiments in the application layers.
A notable exception to this scenario is simulation. A single simulation toolkit, namely Geant4 [4], accounts for more than 50% of the cycles used worldwide by HEP. Simulation applications based on

Geant4 are largely experiment independent and the simulation code is developed and maintained by an international collaboration with a clear governance structure. A substantial performance gain in Geant4 could immediately provide substantial benefits to the whole HEP community and relieve the increasing pressure on the computing resources.
Simulation seems an ideal place to start. A project was launched in 2013 to develop an all-particle transport simulation package with the broad objective of developing code several times faster than Geant4, while continuing improvements of the quality and accuracy of the physics models. The new project, named GeantV, should also offer an infrastructure to perform *fast* and *full* simulation with the possibility of mixing the two modes in different regions of the geometrical setup. One of the major objectives of the project is to achieve a high degree of utilisation of different advanced architectures, including (GP)GPUs and Intel® Xeon Phi™ Coprocessor.
The development of this new code should also allow us to understand what are the intrinsic limitations to achieve an even larger performance gain, up to one order of magnitude and beyond.

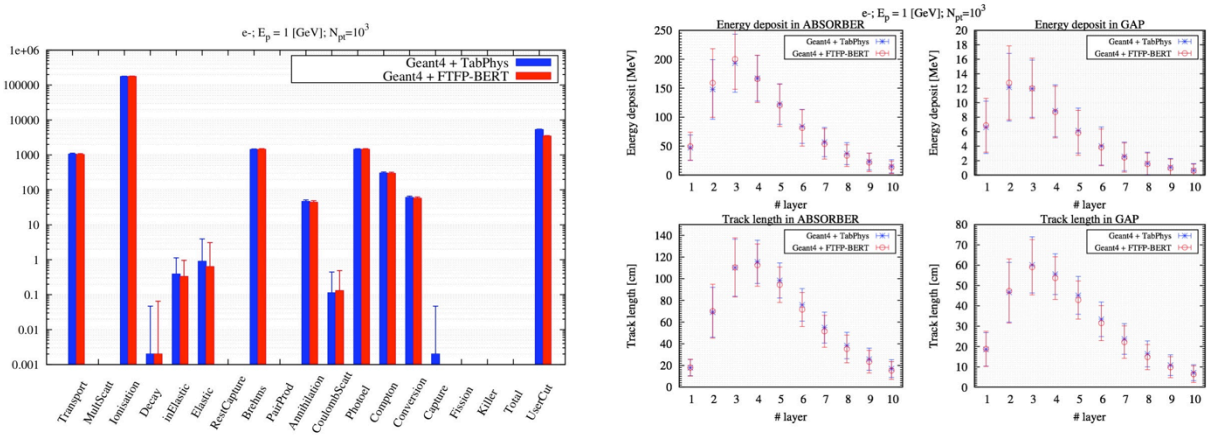


Figure 2. Comparison between tabulated and full physics for electrons.

2.2. The GeantV architecture
The basic principle of GeantV, described in previous papers [5, 6] is that, to exploit SIMD operations, particles are collected in contiguous and aligned arrays (baskets) via filters. The filters make sure that the same algorithm will treat all the particles collected to maximise SIMD effectiveness. Unfortunately this is usually only true for a single "transport step", after which particles move on to different fates, so they have to be re-filtered to be processed again in SIMD mode. The general architecture of the system is shown in Figure 1.
Different "worker" threads handle the transport of different baskets while a separate thread orchestrates the distribution of the baskets to the workers. This basic element (scheduler) is intended for a multi-core socket with concurrent threads sharing read-only data (e.g. cross sections and geometry definitions). Different sockets will be coordinated via an MPI-like coarse-granularity parallelism. The real throughput gain is expected to come from the processing of several tracks at a time in SIMD fashion, while improving at the same time cache coherency. To maximise vector content[10] for different locality criteria, several events will be treated in parallel, which means that tracks from different events will be intermixed in the same basket. This multithreading model will be soon tested in native mode in native mode on current Xeon Phi (codenamed Knights Corner), and extended to include accelerators, considered as *fat* workers to which several baskets will be transferred

[10] Throughout this paper terms like "vector" and "vectorisation" refer to SIMD vectors, as opposed to linear-algebra (3D) vectors, or even STL-related vectors.

for transport. Once all tracks have completed at least a transport step, the baskets will be sent back to the hosts for re-filtering. This scenario assumes asynchronous data transfers and will also work in *hybrid* architectures where a host CPU has different kinds of accelerators attached.
The success of this approach depends on several conditions. First of all there should be an effective performance gain in the use of SIMD instructions. Second, the overhead due to particle *reshuffling* should be small enough not to offset the gain from using SIMD instructions. Third, the resulting code should be reasonably maintainable; in particular it should be portable to different architectures and still achieve good performance.

3. Current status and results obtained

3.1. The physics

In order to test our code we needed a realistic simulation of the physics interactions. We used Geant4 to create tables of cross sections and final states for all particles and all materials. This system provides reasonably accurate cross sections, depending on the number of bins and the energy range. The final state of an interaction is sampled randomly amongst the stored final states, which in our current implementation is five states for each energy, particle and target. While it is impossible to expect from this system an accurate description of the interactions, it nevertheless provides a very faithful reproduction of the shower in terms of global parameters such as energy deposit, number of steps and multiplicity. To validate this approach we have transformed this tabulated physics system into a Geant4 physics list and we have compared the results obtained for a simple calorimeter (in fact a modified version of the Geant4 example N03) with the original Geant4 physics lists from which the tabulated values have been taken from, FTFP_BERT. The results can be seen in Figure 2.

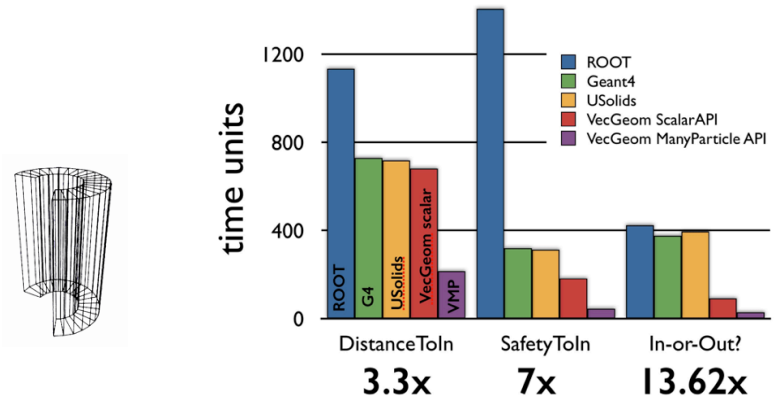


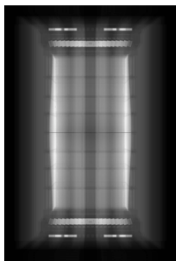
Figure 3. Speed up obtained with VecGeom library with respect to the USolids library for segmented tube shape. gcc 4.7; -O3 -funroll-loops -mavx; no FMA; Geant4.10; Root 5.34.18; benchmark executed with 1024 particles.

3.2. The geometry

Geometry navigation accounts for a significant fraction of the time spent in simulation, it is therefore essential to optimise it to run efficiently on SIMD machines. We therefore started the development of an optimised library for primitive solids and their compositions, and of the development of optimised algorithms for track navigation with API for both single-track and many-track queries. Starting from the USolids library [7], which was developed in the framework of the AIDA project, we have developed a high performance library (VecGeom) of geometrical routines with vector signature [8, 9]. One example of the gain obtained with this new code can be seen in Figure 3. Indeed substantial gains have been obtained already starting from vectors of ten particles. Moreover the new code has also a very good scalar performance in the case of transporting a single particle at a time.

The resulting library is integrated into the USolids code base keeping the existing interface for single particle methods and it will be soon offered as an alternative to the standard Geant4 geometrical solid library. At the moment of writing all solids used in the geometry model of the CMS 2015 detector are implemented and work is continuing on the other solids. Generic vectorised kernels (Section 3.4) are implemented for Box, Tube, Polyhedron, Torus, Trapezoid and Boolean Solids, while a non-vector implementation exists for Cone and Polycone. Backends have been developed so that VecGeom also works on CUDA devices, and on Intel Xeon Phi via the collaboration between CERN PH, Fermilab, CERN openlab and Intel IPCC projects.
VecGeom can trace (navigate) particles in complex geometry hierarchy, both in single-track as well as in multi-track mode, so that we are can perform a first global evaluation of the VecGeom improvements in navigation and shape algorithms. As an example, we have taken the CMS calorimeter and traced so-called geantinos, particles with no physics interaction, through the detector.
Transporting single tracks pixel-by-pixel, images like the one shown in Table 1 are produced, where the pixel intensity correlates with the number of boundaries crossings (much like an x-ray image).

Table 1. Time in s for an *x-ray* scan benchmark with geantinos of the CMS calorimeter along x.



Voxelization	ROOT	Geant4	VecGeom
OFF	31.5	47.5	12.2
ON	11.7	19.2	-

This method helps validating our algorithms compared to ROOT [10] and Geant4 and also allows for a realistic performance comparison. As can be seen in Table 1, when ROOT, Geant4 and VecGeom all perform simple (non voxelised) navigation, we find that VecGeom is almost three times faster than ROOT and equals ROOT and Geant4 when they include optimised voxelisation structures. We expect that voxelisation, when available, will further improve the VecGeom performance.
The same benchmark can also be used to estimate an upper bound for the SIMD gain from tracing many particles at a time. If we trace groups of identical geantinos with our vector-navigator pixel-per-pixel, on an Intel iCore7 with SSE, we improve the tracing time per geantino by a factor of 2.1.
The test suites for Geant4 geometry, USolids and ROOT TGeo have been adapted and extended and they are regularly run on VecGeom. A large database of test cases has been created and careful cross comparisons have been performed with ROOT TGeo and Geant4 geometries, which have allowed discovery of issues also in Geant4 and ROOT that can now be addressed. We have also developed generic benchmarking tools that enable quick performance assessment.

3.3. Scheduler performance
To test our architecture we have decided not to rely on toy models, but to use an LHC-size detector. Since the transport problem is intrinsically non-linear, we felt that it was not possible to just extrapolate a test made on a simple geometrical setup. We have therefore decided to use the geometry of the CMS experiment, exported in GDML format from Geant4.
To verify whether the basket management does not offset the possible SIMD gains in physics and geometry, we have implemented a single threaded version of our basket scheduler and we have exercised it on the full CMS geometry with TGeo (the geometrical package of ROOT), since the VecGeom navigation is not yet fully validated on the complete CMS detector geometry. We have compared this with a simple Geant4 application using the same CMS geometry and scoring routines, and with the same tabulated physics list implemented as a user process.

For a run using as input 100 Pythia pp minimum bias events at 14 TeV, GeantV with TGeo is 2.3 times faster per event than Geant4 in single thread mode, both being run with tabulated physics. The two geometry systems are roughly the equivalent in speed and the same can be said for the physics part. The magnetic field propagator of Geant4, which accounts for less than 10 % of the CPU time, is however more accurate than the GeantV helix propagator, which uses negligible CPU time. This result should not be taken as an absolute speed comparison between Geant4 and GeantV, since GeantV transport is somewhat simplified and likely to become more complex and hence slower as it evolves. This however shows that basket management, which itself is not completely optimised, does not introduce a large overhead. A preliminary profiling of the two applications indicates that GeantV has an improved cycle per instruction ratio (CPI, 0.9 for GeantV compared to 1.1 for Geant4) and much less (factor of ~5) instruction cache misses. To be noted that this benchmark is not yet using the main feature GeantV was designed for, namely vectorisation, which is expected to provide important improvements once the VecGeom geometry is validated for CMS.
To verify that the transport is comparable, we have introduced the same simple scoring both in Geant4 and in GeantV. We have scored particle flux and energy deposit density in CMS ECAL cells for electrons, protons, charged pions, kaons and for gammas. The results of the two codes are compatible modulo some small biases that can be due to different geometry modellers, but still need to be completely understood. See for instance Figure 4 for one such benchmark scoring plots.

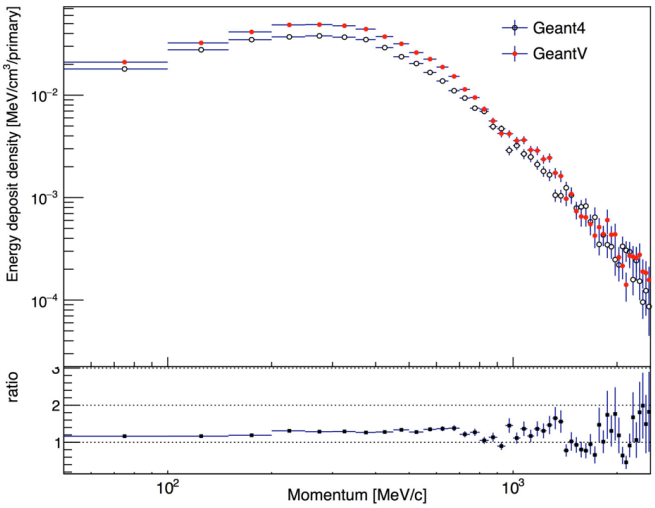


Figure 4. Comparison of proton energy deposit density per primary in the ECAL CMS volume between GeantV with TGeo and Geant4, both with tabulated physics.

3.4. Code portability and optimisation
One of the main challenges for our project is to obtain good code performance on CPU and accelerators (Xeon Phi, (GP)GPU) without duplicating code and writing as many architecture-specific algorithms as we have platforms, as this would clearly be unmaintainable in the long-term. The solution we have adopted is to develop "templated" generic computational kernels using abstract types. These kernels are used to implement single-track, many-track and (GP)GPU methods. We perform the substitution or specialisation via *backends* that are (trait) structs encapsulating standard types and properties for "scalar, vector or CUDA" programming, simplifying information injection into template functions. In our current implementation we use the Vc library [11] to implement machine-specific optimised vector operations with reliable and efficient vectorisation. The extensive usage of templates in the geometry code allows us to also eliminate branches at runtime by specialising the code based on the volume features. For instance a tube with zero internal radius is

instantiated as a cylinder, avoiding all checks on the inner radius at runtime. A similar technique is used for the transformations used to position the volume.

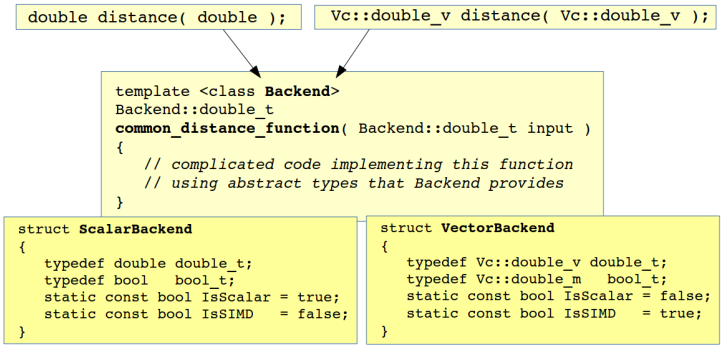


Figure 5. Schema of the substitution of specialised types for vectorisation.

The schema of the template backend can be seen in Figure 5. An example of the power of this technique can be seen in Figure 6 where we show a remarkable relative speedup for geometrical code run on the Xeon Phi. Similar results are obtained with the CUDA backend on (GP)GPUs. These results are encouraging since they seem to indicate that it is possible to achieve portable High Performance Computing code that is generic and still shows high performance on different platforms.

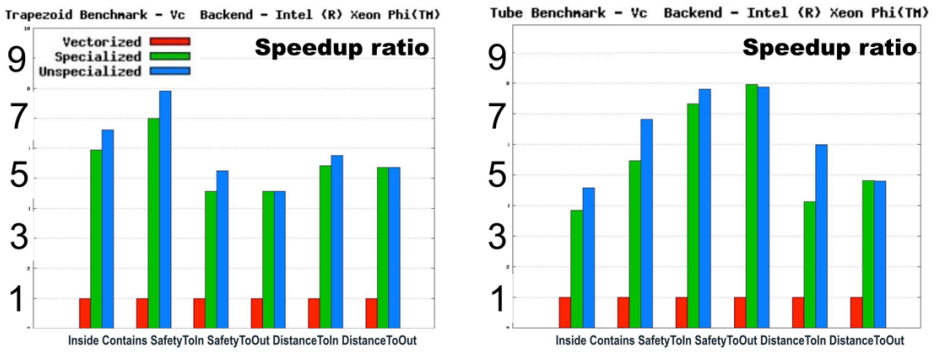


Figure 6. Time to execute specific geometry calculations normalised to the time taken by the vectorised version.

3.5. Optimisation of the physics code

In parallel to the assessment of the limitations and capabilities of the tabulated physics code, we are also analysing the optimisation opportunities for the physics code. We have started with the electromagnetic processes and in particular with the optimisation of the Compton scattering, using the same template techniques used for the optimisation of the geometry. The first results are very encouraging and the performance obtained can be seen in Figure 7. This is a preliminary figure and we still have to conduct a detailed performance study.

4. Concluding Remarks

In the framework of the GeantV project we have developed a new particle transport framework with three main components: a multithreaded scheduler to handle particle arrays (baskets); a vectorised geometry library and navigator and a vectorised version of the Compton scattering method. Our results indicate that basket handling introduces a minimal overhead and SIMD optimisation of geometry and physics introduces a gain in performance that range from half an order of magnitude for Xeon CPUs to

one order of magnitude for Xeon Phi for a single thread. We believe we are on track with our objective to gain a factor between two and five over existing simulations codes.
The next step will be to use all these components to transport a complete event in a few setups of the complexity of a full LHC detector. For this we will have to finish the vectorisation of the remaining geometry shapes (which will be integrated into USolids) and implement a fully vectorised geometry navigator using spatial voxelisation. We will also implement vector transport in a magnetic field and continue the work of vectorising and optimising the electromagnetic physics processes.

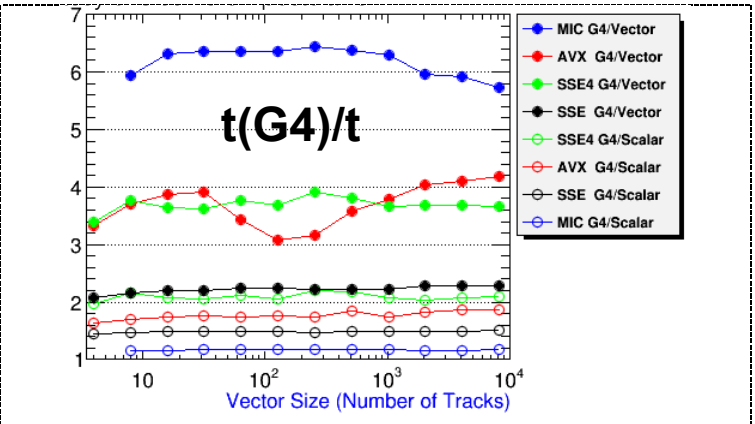


Figure 7. Speedup obtained for SIMD-optimised Compton scattering with respect to Geant4.

Work towards a credible demonstrator for a high-performance particle transport code is progressing, with encouraging results in different areas. The scheduler has been demonstrated transporting a full shower in a large HEP detector. Large portions of the code developed are interfaced with Geant4, and in the case of vectorised physics and VecGeom - USolid library these are more efficient even in scalar execution mode. Once fully validated, these libraries will be available as drop-in alternatives for their current Geant4 counterparts.
The techniques described in this paper have the potential to benefit not only simulation, but also many other areas, within and outside of High Energy Physics.

References

1 Jarp S, Lazzaro A and Nowak A 2012 *J. Phys.: Conf. Series* **396** 052058
2 https://software.intel.com/en-us/articles/ipcc-at-cern-european-organisation-for-nuclear-research
3 https://software.intel.com/en-us/articles/intel-parallel-computing-center-sao-paulo-state-university
4 Allison J et al. 2006 *IEEE Transactions On Nuclear Science* **53** 1, http://geant4.cern.ch
5 Apostolakis J, Brun R, Carminati F, Gheata A and Wenzel S 2014 *J. Phys.: Conf. Series* **513** 052006
6 Apostolakis J, Brun R, Carminati F and Gheata A 2012 *J. Phys.: Conf. Series* **396** 022014
7 Gayer M et al 2012 *IOP Conf Proc.* **396** 052035
8 Apostolakis J, Brun R, Carminati F, Gheata A and Wenzel S 2014 *J. Phys.: Conf. Series* **513** 052038
9 Apostolakis J et al. 2014, Towards a high performance geometry library for particle-detector simulation. Proceedings of the 16th International workshop on Advanced Computing and Analysis Techniques in physics research (ACAT); Prague 1.9.-5.9.2014, in publication
10 Brun R and Rademakers F 1997 *Nucl. Instr. Meth. Phys. Res.* A **389**, p81-86, http://root.cern.ch
11 M. Kretz and V. Lindenstruth 2011 „Vc: A C++ library for explicit vectorisation“, *Software: Practice and Experience*