Studies of GEANT4 performance for different ATLAS detector geometries and code compilation methods

Caterina Marcon[1,*], *Einar* Elén[1,**], *Jessica Rebecca* Madeira[1], *Benjamin* Morgan[2], *Oxana* Smirnova[1,***], and *David* Smith[3]

[1]Lund University, Department of Physics, Box 118, SE - 221 00 Lund, Sweden
[2]University of Warwick, Department of Physics, Gibbet Hill Road, Coventry CV4 7AL, UK
[3]CERN, 1211 Geneva CH

Abstract. Full detector simulation is known to consume a large proportion of computing resources available to the LHC experiments, and reducing time consumed by simulation will allow for more profound physics studies. There are many avenues to exploit, and in this work we investigate those that do not require changes in the GEANT4 simulation suite. In this study, several factors affecting the full GEANT4 simulation execution time are investigated. A broad range of configurations has been tested to ensure consistency of physical results. The effect of a single dynamic library GEANT4 build type has been investigated and the impact of different primary particles at different energies has been evaluated using GDML and `GeoModel` geometries. Some configurations have an impact on the physics results and are, therefore, excluded from further analysis. Usage of the single dynamic library is shown to increase execution time and does not represent a viable option for optimization. Lastly, the static build type is confirmed as the most effective method to reduce the simulation execution time.

1 Introduction

Particle physics has an ambitious experimental program for the coming decades: during the High-Luminosity Large Hadron Collider (HL-LHC) phase, scheduled to begin data taking in 2027, events will be collected at very high rates. The rate foreseen for the ATLAS experiment is 10 kHz, approximately ten times more than during previous runs [1, 2].

In addition to the experimental challenges of collecting, storing and analysing such a large volume of data, a comparable amount of Monte Carlo (MC) simulated data will be required in order to prevent simulation-dominated systematic uncertainties [3]. Currently, approximately half of the MC events in ATLAS are produced with *full simulations*, i.e. using the GEANT4 simulation toolkit [4]. The remaining MC events are instead produced with *fast simulations*, which adopt a parameterized approach.

At present, detector simulation accounts for almost 40% of the CPU hours consumed by the ATLAS experiment (see Fig. 1 in [3]). However, for many analyses, the scarce availability

[*]e-mail: caterina.marcon@hep.lu.se
[**]e-mail: einar.elen@nuclear.lu.se
[***]e-mail: oxana.smirnova@hep.lu.se

of MC events is still a limiting factor. The reduction of the time spent on simulations is, thus, a priority, and an active R&D program aimed at optimizing the GEANT4 CPU requirements is ongoing in ATLAS. As summarized in Fig. 1, the R&D program considers three different scenarios [3]:

- **Baseline**: this is the model for the LHC Run 3, starting in 2022. The events will be equally distributed between full GEANT4 and fast simulations. The latter, in particular, will be used for parameterized calorimeter response;
- **Conservative R&D**: the fraction of events produced with fast simulations is expected to increase significantly (up to 75%) throughout the HL-LHC phase (Run4 and Run5);
- **Aggressive R&D**: 90% of the events are assumed to be produced with fast simulations over the same period of time.

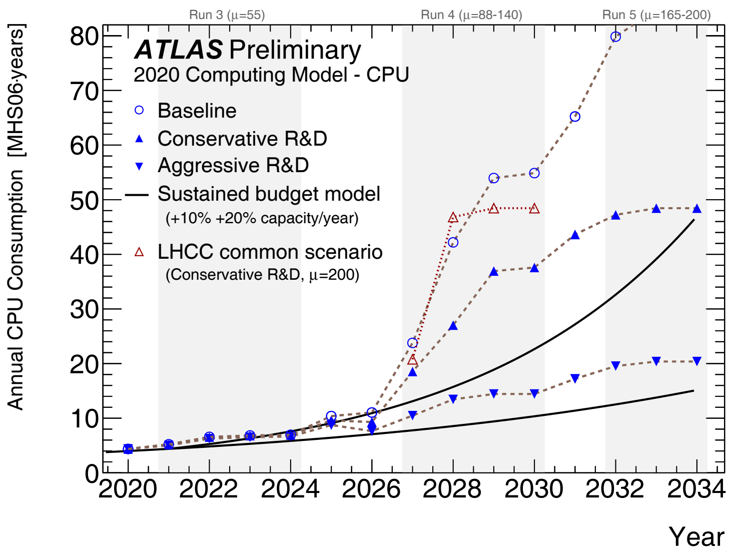

Figure 1: Projected CPU requirements for ATLAS between 2020 and 2034 based on 2020 assessment. The solid black lines represent a projection of the computing availability assuming a yearly budget increase of +10% and +20%. The empty circles show the projection of what the computing needs will be if the experiment would keep the same computing model as in Run 2. The filled triangles refer to the *conservative* and *aggressive* R&D scenarios. The empty triangles indicate the *conservative* R&D scenario under the assumption of the LHC reaching an average of 200 proton-proton interactions per bunch crossing in Run4 and beyond (2028-2030) [3].

The full simulation requires around five times more computing resources than the fast simulation, that will be the preferred choice for Run 4 and beyond. Nevertheless, the use of the full simulation will remain unavoidable for certain detectors and will be required to tune the fast simulation [3]. It is, therefore, extremely important to continue the GEANT4 optimization in order to ensure unbiased physics results while minimizing the computational footprint.

The aim of this study is to investigate different methods to reduce the full simulation execution time without sacrificing the quality of the simulated data and without altering the existing source code [5]. A broad range of build-time configurations has been tested in order to perform a consistency check to ensure the independence of physics results from compiler-specific options. Moreover, the impact of different build types and of different primary particles on the simulation execution time has been investigated.

2 Methods

All the calculations presented in this paper are based on a standalone GEANT4 simulation [6]. This study is articulated in three main parts:

1. Validation to ensure that physics results (energy deposition is used as a metric) are not affected by compiler-specific options. This was carried out on a broad range of compilers, GCC 4.9.4, 6.2.0 and 8.3.0, Clang and ICC, and build-time configurations, including Link-Time Optimization (LTO), `Ofast` and native architecture instructions [7]. Calculations were carried out on a CERN standalone machine and the Aurora cluster at Lund University (Table 1) with a single-thread GEANT4 10.5.0 installation. Negative pions at 50 GeV were used as primary particles. For these tests, a GDML geometry comprising the full inner detector, the LAr hadronic and tile calorimeters, the EM barrel and the muon spectrometer has been used. This geometry does not contain a definition for the electromagnetic calorimeter endcap (EMEC).

2. In order to evaluate the impact of using a single dynamic library on the simulation execution time, multiple runs of the standalone simulations have been performed, each with 2500 50 GeV negative pions as primary particles. The code was built against GEANT4 10.5.0 on a CERN standalone machine (see Table 1) with GCC 6.2.0 and 8.2.0 compiler versions, four optimization levels and was executed with 4 threads. To build the single dynamic GEANT4 library for these tests, the CMake structure has been modified. The new flag `BUILD_SINGLE_LIB` was added; it is an optional flag and it must be enabled in addition to the standard `BUILD_SHARED_LIBS` and `BUILD_STATIC_LIBS` flags. This allows the choice of which build type should be used for the single library [8]. For these tests, the same GDML geometry file was used.

3. To estimate the impact of different particles on the simulation execution time, a first preliminary study has been carried out using the same GDML geometry; protons, positive/negative pions and geantinos, the massless virtual particles available in GEANT4, were chosen as primary particles. For each of them, two energies were considered: 10 and 20 GeV; for each run 5000 primaries were generated. All simulations were performed on the Aurora cluster at Lund University and full nodes were reserved with the `exclusive` option (see Table 1). The code was built against GEANT4 10.6.2 and GCC 8.2.0.

   In addition, in order to include the effect of the EMEC on the simulation execution time, a second, more complete geometry definition was adopted. Support for the `GeoModel` representation of the ATLAS geometry has been added to the standalone simulation [9, 10]. The impact of different primary particles, namely charged pions and protons, at different energies (10, 20 and 50 GeV) has been evaluated. The simulations were run with 5000 primary particles on a CERN standalone machine (Table 1) and built with GCC 8.2.0 against GEANT4 10.6.2. For both geometry configurations, the standard static and a multi-library dynamic GEANT4 build types have been tested. The reference physics list used is FTFP_BERT, the current Geant4 default physics list [11].

Table 1: Computing resources

	CERN standalone machine	**Compute node on Lund University cluster**
CPU	2× Intel Xeon E5-2630 v3 2.40GHz	2× Intel Xeon E5-2650 v3 2.30GHz
Architecture	64 bit Haswell x86_64	64 bit Haswell x86_64
N. of cores	16	20
Threads per core	2	1
Cache	20 MB (L1: 64 KB, L2: 256 KB, L3: 20 MB)	25 MB (L1: 64 KB, L2: 256 KB, L3: 25 MB)
RAM	64 GB	128 GB
Filesystem	XFS	IBM General Parallel File System (GPFS)
Operating system	CentOS 7	CentOS 7

3 Results and discussion

3.1 Physics validation

The analysis of the average energy deposition per event in all parts of the detector including active and non-active material, carried out with 5 different compilers, revealed that the results are not always compiler-independent, and the observed differences can be ascribed to the following causes (Fig. 2) [12]:

- use of unsafe math optimizations (`-Ofast`, ICC compiler or native architecture instructions);
- use of compilers from the Clang family or older versions of GCC (such as 4.9.4), that produce different energy depositions and different random numbers sequences, despite the use of a fixed random seed. The CLHEP implementation of the Mersenne Twister algorithm is used in the benchmark simulation [13]. Further studies are ongoing to assess the reproducibility of random sequences.

The observed deviations in energy deposition suggest the need to exclude the aforementioned cases and to limit the rest of the studies presented to two GCC versions, namely 6.2.0 and 8.2.0, and to four optimization flags, `-Os`, `-O1`, `-O2`, `-O3`.

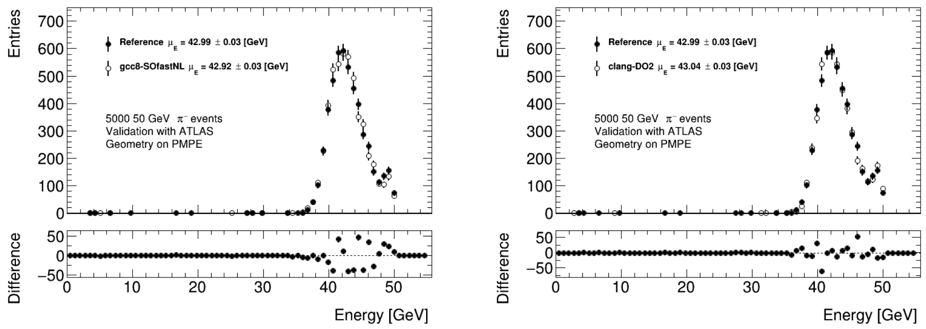

Figure 2: Energy depositions per event in the full detector in two of the configurations resulting in differences in the average energy deposition. GCC 4.9.4, Dynamic linking, `-O2` is used as a reference. The average energy per event is $\mu_E = 42.92 \pm 0.03$ GeV (left, GCC 8.2.0, static linking,`-Ofast`, native architecture instructions, and LTO) and $\mu_E = 43.04 \pm 0.03$ GeV (right, Clang, dynamic linking, `-O2`). The GDML geometry (see Sec. 2) was used.

3.2 Studies with the single dynamic library

In Fig. 3, a comparison of the execution time between three different build types is shown: static, dynamic (default multi-library configuration) and single dynamic library. For all cases, differences in performance are expressed as a relative percentage with respect to the reference case: multi-library, GCC 8.2.0 with -O2 optimization. For each of the studied configurations the benchmark simulation was run 5 times. Average values are presented and in all cases, standard deviations are of the order of 2%.

For both compiler versions, the single-library approach exhibits an increase of ∼ 10% in execution time. This effect seems to be counter-intuitive, but could be explained by considering how shared libraries call and load objects in memory and how the interaction between GEANT4 core libraries and user application is structured. Each call to a function in a dynamic library takes advantage of a trampoline which reads the memory address of the called method from a lookup table and passes it to the calling function. This results in an increased number of calls and jumps, which eventually slows down the simulation execution [14].

As found in previous studies [5], different optimization flags do not have a significant impact on the simulation times.

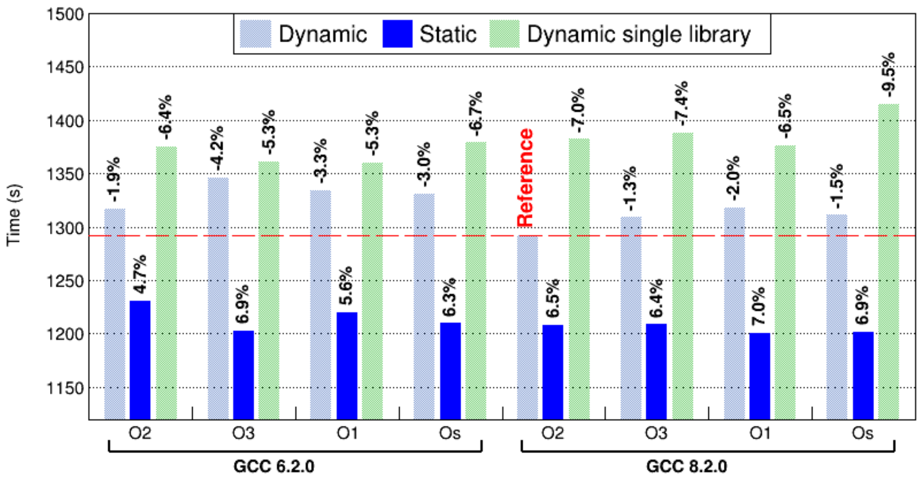

Figure 3: Comparison of the execution times between three different build types: static, dynamic (default multi-library configuration) and dynamic single library.

3.3 Impact of different particles

In order to investigate the impact that static and dynamic builds have on interactions of different complexity, several primary particles (protons and charged pions) of different energies (10, 20 and 50 GeV) have been considered. The average results are summarized in Table 2 and Table 3; the former were obtained from the GDML geometry (without EMEC), whereas the latter were produced with the complete ATLAS geometry.

For all the primary particles analyzed, a decrease in the simulation execution time is observed for the static build, when compared to the dynamic case. This improvement is increasingly pronounced as the complexity of the interactions grows. Considering the geantinos, a

5% decrease in time was observed (Table 2). The speed-up rises to 6% in the case of 50 GeV protons tested with the full ATLAS geometry (Table 3) and exceeds 10% in case of 20 GeV protons tested with GDML geometry (Table 2).

The static build shows a tendency to be less sensitive to the type of primary particles used. For example, in simulations run at 20 GeV with dynamically linked libraries, the proton exhibited an average 4.5% increase in the simulation time with respect to the pions. This percentage decreases to about 3.6% in the static case.

For energies of 20 and 50 GeV, computations with protons show a longer execution time. Following the cumulative distribution function [15], the proton undergoes more ionization processes in the medium it traverses. Additionally, based on the particles stopping power plots [16], the energy loss of the pion is larger than the proton's at these energies. Thus, the extra ionization processes simulated for the proton, not only due to its higher probability of interaction, but also longer distances travelled before absorption, are the primary cause for the increase in execution time.

For both build types and both geometries, differences in the results for positive and negative pions are consistent with the slightly larger interaction cross section of the negative particle at the considered energy [16].

Pure propagation, tested using geantinos, has a negligible impact on the running time which in all cases is ~3 s per run.

Table 2: Execution times per run for p, $\pi^\pm$ and geantinos at 10, 20 and 50 GeV, tested with static and dynamic GEANT4 builds. The GDML geometry (without EMEC) is used with 5000 primary particles [17].

Particle type	**Simulation time (s)**	**Decrease w.r.t. protons (%)**	**Increase w.r.t. static case (%)**
	Dynamic library (10 GeV, 20 runs)		
p	601 ± 9	—	9.9
π^-	594 ± 10	1.1	10.4
π^+	577 ± 5	4.2	9.6
Geantino	3.0 ± 0.1	1.99×10^4	5.6
	Static library (10 GeV, 20 runs)		
p	546 ± 6	—	—
π^-	538 ± 8	1.5	—
π^+	526 ± 4	3.9	—
Geantino	3.2 ± 0.1	1.7×10^4	—
	Dynamic library (20 GeV, 100 runs)		
p	1130 ± 14	—	10.5
π^-	1083 ± 19	4.3	9.4
π^+	1079 ± 18	4.7	9.7
Geantino	3.1 ± 0.1	3.64×10^4	4.7
	Static library (20 GeV, 100 runs)		
p	1023 ± 12	—	—
π^-	990 ± 14	3.3	—
π^+	983 ± 12	4	—
Geantino	3.0 ± 0.1	3.4×10^4	—

Table 3: Execution times per run for p and $\pi^{\pm}$ at 10, 20 and 50 GeV, tested with static and dynamic GEANT4 builds. The `GeoModel` geometry (including EMEC) is used with 5000 primary particles.

Particle	**Simulation time (s)**	**Decrease w.r.t. protons (%)**	**Increase w.r.t. static case (%)**
	Dynamic library (10 GeV, 45 runs)		
p	647 ± 9	—	6.2
π^-	657 ± 9	−1.5	6.3
π^+	640 ± 10	1.1	7.0
	Static library (10 GeV, 45 runs)		
p	609 ± 10	—	—
π^-	618 ± 9	−1.5	—
π^+	598 ± 7	1.8	—
	Dynamic library (20 GeV, 45 runs)		
p	1212 ± 13	—	6.5
π^-	1209 ± 18	0.2	6.7
π^+	1181 ± 14	2.6	6.1
	Static library (20 GeV, 45 runs)		
p	1138 ± 11	—	—
π^-	1133 ± 14	0.4	—
π^+	1113 ± 13	2.2	—
	Dynamic library (50 GeV, 45 runs)		
p	2797 ± 46	—	6.1
π^-	2752 ± 40	1.6	6.9
π^+	2715 ± 39	3.0	6.9
	Static library (50 GeV, 45 runs)		
p	2636 ± 30	—	—
π^-	2573 ± 28	2.4	—
π^+	2539 ± 37	3.8	—

4 Conclusions and outlook

Several factors can significantly affect the full GEANT4 simulations execution times. This study has shown that unsafe math optimizations as well as certain compilers, namely the Clang family and older GCC versions, may have a negative impact on the quality of the physics results.

Tests with the single dynamic library resulted in a $\sim 10\%$ increase in the execution time, and this can be ascribed to the trampoline/lookup table mechanism of dynamic linking. This new build type, tested here for the first time, is, thus, not recommended as a viable choice for improving the time performance of the full simulations.

All the investigations carried out with the static build type with the GDML geometry (without EMEC) have shown a gain of about 10% with respect to the reference multi-library dynamic case. Evaluations with the `GeoModel` geometry, representing the full ATLAS detector (including EMEC), resulted in a smaller gain of about 7%. Despite this reduction in gain with the full geometry, the static build is confirmed as the most effective technique for optimizing the full simulations execution time. It is, therefore, advisable to expand the investigations on this build type by evaluating the performance of a single static library combined with the full `GeoModel` geometry. Eventually, these studies should also be integrated and tested in the environment of the Athena framework.

References

[1] ATLAS Collaboration, JINST **3**, S08003 (2008)
[2] J. Albrecht et al., Computing and Software for Big Science **3** (2019)
[3] P. Calafiura, J. Catmore, D. Costanzo, A. Di Girolamo, Tech. rep., CERN (2020), CERN-LHCC-2020-015 ; LHCC-G-178
[4] S. Agostinelli et al., Nucl. Instrum. Meth. A **506**, 250 (2003)
[5] C. Marcon, O. Smirnova, S. Muralidharan, EPJ Web Conf. **245** (2020)
[6] A. Dotti, *Geant4 benchmark simulation*, https://gitlab.cern.ch/adotti/HepExpMT (2018), accessed: 2021-02-25
[7] R.E. Bryant, D.R. O'Hallaron, *Computer Systems: A Programmer's Perspective (2nd Edition)* (Prentice Hall, 2010), ISBN 9780136108047
[8] C. Marcon, B. Morgan, *Building Geant4 as a single dynamic or static library*, https://gitlab.cern.ch/atlas-simulation-team/geant4/-/commit/89704795358426a25d6518f5197762bdab19d007 (2020), accessed: 2021-02-25
[9] M. Bandieramonte, J. Boudreau, R.M. Bianchi, Tech. rep., Geneva (2020)
[10] R. Chytracek, J. Mccormick, W. Pokorski, G. Santin, IEEE Trans. Nucl. Sci. **53**, 2892 (2006)
[11] J. Allison, K. Amako, J. Apostolakis, P. Arce, M. Asai, T. Aso, E. Bagli, A. Bagulya, S. Banerjee, G. Barrand et al., Nucl. Instrum. Meth. A **835**, 186 (2016)
[12] E. Elén, *Optimization and validation of Geant4 detector simulation software for the ATLAS experiment at the LHC* (2020), Student Paper, Lund University
[13] M. Matsumoto, T. Nishimura, ACM Transactions on Modeling and Computer Simulation **8**, 3 (1998)
[14] V. Agrawal et al., *Architectural Support for Dynamic Linking*, in *Proceedings of the Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems* (Association for Computing Machinery, New York, NY, USA, 2015), Asplos '15, pp. 691–702, ISSN 9781450328357
[15] A. Lechner, CERN Yellow Rep. School Proc. **5** (2018)
[16] P.A. Zyla et al., PTEP **2020** (2020)
[17] R. Jigström Madeira, *Impact of different primary particles on geant4 simulation execution time - study on protons and pions* (2021), Student Paper, Lund University