Journal of Physics: Conference Series

ATLAS Geant4 Simulation Optimizations[1]

Mustafa Schmidt[1], Akanksha Vishwakarma[2], Andrei Sukharev[3], Benjamin Michael Wynne[2], Benjamin Morgan[4], Caterina Marcon[5], Dongwon Kim[6], Evangelos Kourlitis[7], Evgueni Tcherniaev[8], Guilherme Amadio[8], John Apostolakis[8], John Derek Chapman[9], Marilena Bandieramonte[10], Miha Muskinja[11], Mihaly Novak[8], Tommaso Lari[5], Walter Hopkins[7]

[1]Bergische Universität Wuppertal, [2]The University of Edinburgh (GB), [3]Budker Institute of Nuclear Physics (RU), [4]University of Warwick (GB), [5] INFN Sezione di Milano (IT), [6] Stockholm University (SE), [7]Argonne National Laboratory (US), [8] CERN (CH), [9]University of Cambridge (GB), [10]University of Pittsburgh (US), [11]Lawrence Berkeley National Laboratory (US)

E-mail: `muschmidt@uni-wuppertal.de`

Abstract. Several optimization techniques are implemented in the ATLAS Geant4 detector simulation to improve CPU and memory usage. These optimizations include tracking methods for gammas, the usage of static linking, adjustments to electromagnetic range cuts, a Russian Roulette process for reducing simulation steps, and advanced geometry management. The latest performance benchmarks show significant improvements in simulation speed and resource efficiency, crucial for future LHC high-luminosity operations. Consequently, more projects are initiated to improve the ATLAS full simulations even further, mainly concerning improving the CPU time. This paper summarizes the validated and ongoing tasks as well as the obtained performance increase.

1. Introduction

The ATLAS experiment at the Large Hadron Collider (LHC) relies heavily on simulated event samples produced using full Geant4 detector simulations [1]. Monte Carlo (MC) simulations based on Geant4 are major consumers of computing resources, a trend that is expected to continue in the High-Luminosity LHC (HL-LHC) era [2]. Therefore, optimizations in full Geant4 simulations are crucial for efficient resource utilization. This paper discusses various optimizations that are already implemented and validated or still ongoing. Throughout the past years, the computation time for the existing MC production campaigns, such as mc20, has continuously decreased. All presented benchmark values in the following sections are comparisons between the Run3 setup for mc23 and the unoptimized configuration applied for the Run2 configuration in mc16. The overall goal is an improvement of CPU and memory consumption without compromising physics accuracy. All optimizations are done in close collaboration with the Geant4 collaboration which always results in fruitful feedback loops between both groups. Many optimization results were already included in the main Geant4

[1] Copyright 2024 CERN for the benefit of the ATLAS Collaboration. Reproduction of this article or parts of it is allowed as specified in the CC-BY-4.0 license

software framework, while others are only used within the ATLAS offline simulation and reconstruction framework Athena [3].

2. Implemented & Validated Optimizations

2.1. Woodcock Tracking for Gammas

Woodcock tracking [4] was found to have the largest impact on the overall computation time. Its goal is to optimize the tracking of neutral particles, particularly photons, especially in highly segmented detectors due to the many simulation steps that have to be performed on each boundary. Woodcock tracking limits these simulation steps by performing the particle propagation in a unified geometry using the material with the highest macroscopic cross-section, such as lead (Pb). The interaction probability is then adjusted based on the ratio of cross-sections of the true material and Pb. This method is implemented as a special wrapper process for gammas directly in Geant4 and can be used in other experiments as well. The results of this optimization include a 50% reduction in steps within the region of the ATLAS electromagnetic endcap calorimeter (EMEC) [5] and an overall 17.5% speedup in Athena simulations.

2.2. Static Linking

The static linking optimization aims to use Geant4 as a static library to avoid lookup table delays. The approach involves defining a BigSimulation shared library by grouping all libraries from Athena packages that use Geant4. Since static linking does not change the way of computing numbers, no physics validation is required for this task. This optimization results in a speedup of 5-7%, depending on the compiler version.

2.3. EM Range Cuts

The motivation behind EM range cuts [6] is that different energy thresholds in Geant4 are relevant for the production of secondary particles. This tuning was already successfully applied for e^+/e^- processes in previous MC campaigns. The approach sets the secondary production energy threshold below which secondary particles are not created. Below these thresholds, the remaining energy is deposited at the end of the production step. Simulations related to Run3 enabled additionally the range cuts for gamma processes (Compton scattering, photo-electric effect, and conversion processes). Those in fact are turned off by default in Geant4. The benchmark of this optimization shows an 8% CPU speedup and a significant reduction of 60% in simulated low-energy electrons.

2.4. Russian Roulette

The Russian Roulette technique [6] is motivated by the fact that neutrons and photons consume most of the CPU time, particularly in the Barrel EM calorimeters and EMEC. The approach involves the Photon/Neutron Russian Roulette (PRR/NRR) method, which randomly discards particles below an energy threshold and weights the energy deposits of the remaining particles accordingly. This technique achieves a 10% speedup with a 2 MeV threshold for neutrons and 0.5 MeV for photons.

2.5. G4GammaGeneralProcess

The motivation for the G4GammaGeneralProcess [7] optimization is to reduce computation time caused by multiple gamma processes, such as photoeffect, compton scattering, or pair-production. To address this issue, a collective physics process for photons is introduced, which reduces the number of instructions and calculations required at geometry boundary crossings. This optimization results in an overall 3% CPU speedup.

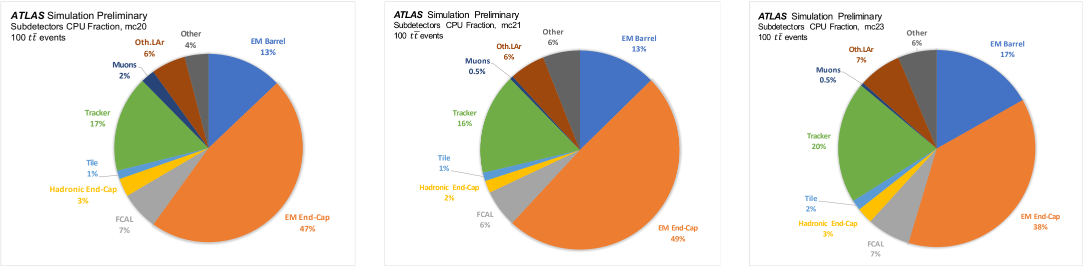

Figure 1. CPU time fractions for different subetectors of ATLAS for all MC campaigns.

2.6. Magnetic Field Optimizations

Tracking particles in a magnetic field is resource-intensive. However, the residual solenoid magnetic field within the bulk of the calorimeter volume is small. As a result, it is possible to ignore the magnetic field for the tracking of muons or high-energy electrons or positrons in the region of the liquid argon colorimeter (LAr) without affecting the shower shape. This optimization results in a 3% speedup.

2.7. EMEC Custom Solid

The EMEC custom solid optimization aims to improve the efficiency of EMEC, which was described using the solid class G4Polycone. Within this approach, G4Polycone is replaced with the class G4ShiftedCone, which is not a standard Geant4 shape, and the outer wheel is divided into conical-shaped sections. The G4ShiftedCone class is a copy from standard G4Cons and modified to represent a 2π cone and to have an arbitrary position along the z-axis, allowing for faster geometric calculations with fewer computational overheads by using optimized mathematical calculations. In addition, further subdivisions of the new wheel into thick slices along the z-axis were done which resulted in faster computation time due to higher tracking efficiency and usage of boundaries. This optimization results in a 5-6% speedup.

2.8. VecGeom Integration

The VecGeom integration involves optimizing the implementation of geometrical shapes using the VecGeom library [8], which leverages explicit and implicit vectorization. This approach only replaces polycons, cones, and tubes relevant to the geometry. The results of this optimization show a speedup of 2-7%.

3. Performance Studies

3.1. Computing Fractions

The performance studies analyze the CPU time distribution among different subdetectors during the MC campaigns mc20, which was the last Run2 MC campaign, mc21, which was the first Run3 MC campaign, and mc23 as the latest Run3 MC campaign. The largest CPU fraction is found in the EMEC, followed by the tracker and the Barrel Electromagnetic Calorimeter (EMB), while the Tile and the muon detector show the smallest impact on performance. An overview of the newest performance results can be found in Figure 1.

The analysis of CPU time focuses on the time spent per event simulating 100 $t\bar{t}$ events, which is an important benchmark for simulations. The study examines the time spent for each major subdetector and each particle type. Significant time is spent in the EMEC, as shown in Figure 2, due to the processing of electrons and gammas. However, a strong improvement since the MC campaign mc20 is visible. The total improvement achieved is a 50% reduction in CPU time compared to the Run2 configuration.

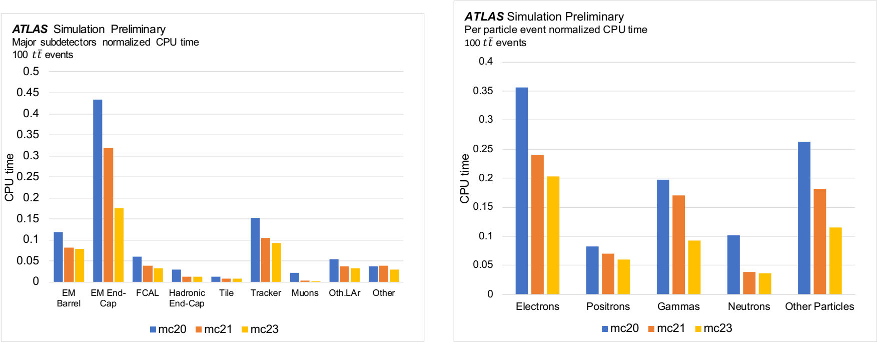

Figure 2. CPU time fractions for different subetectors (left) of ATLAS and particle species (right) comparing all MC campaigns.

In addition to the plots above, the number of steps required for the propagation of each particle species in all relevant subdetectors was computed. The results using all previously described optimizations are shown in Figure 3. Each two neighboring bars represent both important MC campaigns mc20 and mc23 used for a direct comparison with the latest obtained optimization results.

4. Ongoing & Future Tasks

4.1. EMEC Geometry Optimization

The current implementation of the EMEC uses custom solids that do not exist in Geant4. The optimization approach involves defining the geometry of EMEC with standard G4 shapes (G4GenericTrap) for faster simulation and compatibility with GPU usage. Those volumes perform much better than the originally used G4Polycones. Additional slices in the z-direction provide further improvements. Ongoing comparisons between different geometry options and configurations are being conducted. Preliminary results show an improvement in CPU time by 19% in a standalone simulation framework called FullSimLight [9]. Currently, the geometry optimizations are being transferred into the Athena software framework to validate the obtained results.

4.2. Advanced Compiler Optimization

The goal of advanced compiler optimization is to speed up simulations using two different approaches. This involves smarter usage of the compiler for more efficient use of computing resources. There are two possible ways (link-time-optimization and profile-guided optimization) to achieve this goal which are explained in the following. Both methods should not be treated as being fully independent and can influence each other under certain conditions due to synergy effects. However, when combined, the speedup might well be greater than either technique alone.

4.2.1. Link-Time Optimization (LTO) LTO [10] is already validated successfully and is mainly a compilation technique that optimizes the entire program at the linking stage. Unlike traditional compilation, which optimizes individual files separately, LTO considers the whole program and thus allows for more aggressive and comprehensive optimizations. This can result in significant performance improvements because the optimizer has a global view of the code. Preliminary benchmarks show a 3-4% speedup with LTO.

4.2.2. Profile-Guided Optimization (PGO) PGO [11] is a method where the compiler uses runtime profiling data to optimize the program. The process involves two steps: first, the

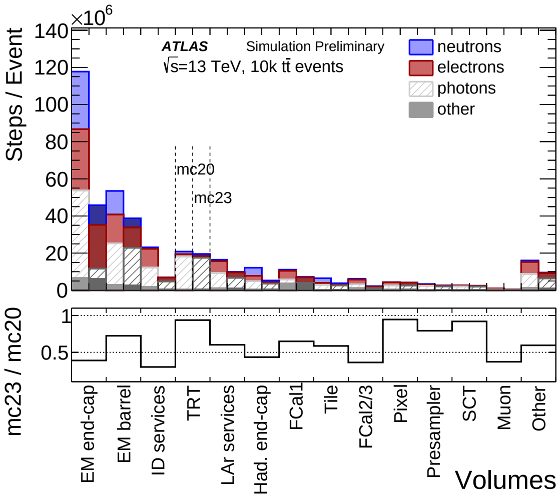

Figure 3. Number of processed steps for each particle species in all relevant ATLAS subdetectors comparing mc20 and mc23. Two neighboring bars represent both campaigns for a direct comparison with the latest obtained optimization results.

program is compiled and run with representative input to collect profiling data and after that, the program is recompiled using this data. The profiling information helps the compiler make informed decisions about inlining, branch prediction, loop unrolling, and other optimizations. This optimization can lead to better performance by tailoring the compiled code to actual usage patterns. In total, 3-5% speedup with PGO seems to be achievable according to preliminary results.

4.3. High-Eta Particle Rejection

The High-Eta Particle Rejection aims to eliminate primary particles that generate secondaries close to the beam pipe because these particles do not contribute to the physics analysis due to not being visible by important detector components, but they can increase the overall CPU time significantly. This involves generating a large sample of particles to map out relevant signals and using a new particle killer to drop irrelevant particles.

4.4. Voxel Density Optimization

The voxel density is a member variable of the Geant4 logical volume classes. This value was previously called artlessness and defines the trade-off between memory usage and computational overhead for internal calculations. Variations of this parameter are, therefore, not expected to influence the physics performance which speeds up the validation process. The related optimization aims to find optimal values for voxel density to improve CPU time and memory consumption, as previous studies with certain geometries have indicated promising results. This involves tuning the size and granularity of voxels and making improvements in memory consumption for geometry optimizations. Preliminary results indicate small improvements, however, these results are highly influenced by statistical and systematic uncertainties. Hence, follow-up studies have to be done to confirm and refine these results.

5. Conclusion & Outlook

Significant optimizations have been implemented in the ATLAS Geant4 simulation, improving CPU time and memory efficiency without sacrificing physics accuracy. For this purpose, a

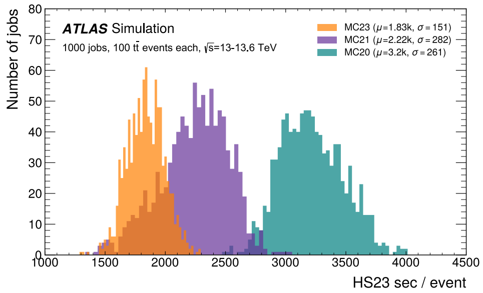

Figure 4. Overall Performance Improvements in simulations for Run3 [3] showing distributions of the HEPScore times [12]. μ represents the mean value of each distribution and σ the width of the distribution.

physics validation is frequently performed to check certain physics objects, such as particle multiplicity, flavor tagging, or missing energy distributions of dedicated benchmark channels. A total reduction of CPU time over 50% compared to Run2 samples has been achieved. The overall improvements of mc23 compared to mc20 and mc21 are shown in Figure 4 with μ indicating the mean value and σ the width of the distribution. The x-axis displays the HEPScore HS23 event time [12]. Key advancements include Woodcock tracking, EM range cuts, the optimized geometry, and the new G4GammaGeneralProcess. For that purpose, a close collaboration with the Geant4 team has been crucial to obtaining these achievements.

Ongoing improvements focus on leveraging modern computing architectures and refining simulation precision and resource usage. Continuous validation against Geant4 ensures high standards of physics accuracy. These efforts will help maintain the efficiency and accuracy of simulations as the ATLAS experiment progresses into the HL-LHC era.

References

[1] J. Allison et al. Recent developments in Geant4. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 835:186–225, 2016.

[2] John Apostolakis et al. HEP Software Foundation Community White Paper Working Group–Detector Simulation. *arXiv preprint arXiv:1803.04165*, 2018.

[3] ATLAS Collaboration. Software and computing for Run 3 of the ATLAS experiment at the LHC. *arXiv:2404.06335*, 2024.

[4] D. Legrady, B. Molnar, M. Klausz, and T. Major. Woodcock tracking with arbitrary sampling cross section using negative weights. *Annals of Nuclear Energy*, 102:116–123, 2017.

[5] ATLAS collaboration. The ATLAS experiment at the CERN Large Hadron Collider: a description of the detector configuration for Run 3. *Journal of Instrumentation*, 19(05):P05063, May 2024.

[6] Miha Muškinja, John Derek Chapman, and Heather Gray. Geant4 performance optimization in the ATLAS experiment. *EPJ Web Conf.*, 245:02036, 2020.

[7] Hahnfeld, Jonas et al. Geant4 electromagnetic physics for Run3 and Phase2 LHC. *EPJ Web of Conf.*, 295:03018, 2024.

[8] John Apostolakis, Gabriele Cosmo, Andrei Gheata, Mihaela Gheata, Raman Sehgal, and Sandro Wenzel. A vectorization approach for multifaceted solids in VecGeom. *EPJ Web Conf.*, 214:02025, 2019.

[9] Marilena Bandieramonte, Riccardo Bianchi, and Joseph Boudreau. FullSimLight: ATLAS standalone Geant4 simulation. *EPJ Web of Conferences*, 245:02029, 01 2020.

[10] T. Glek and J. Hubicka. Optimizing real world applications with GCC Link Time Optimization. *arXiv:1010.2196*, 2010.

[11] Baptiste Wicht, Roberto A. Vitillo, Dehao Chen, and David Levinthal. Hardware Counted Profile-Guided Optimization. *arXiv:1411.6361*, 2014.

[12] Domenico Giordano et al. HEPScore: A new CPU benchmark for the WLCG. *arXiv:2306.08118*, 2023.