The challenges of the HL-LHC for the Full and Fast Simulation of the CMS Experiment

Natascha Krammer[a,*] **on behalf of the CMS Collaboration**

[a] *Institute of High Energy Physics, Austrian Academy of Science, Vienna, Austria*

E-mail: natascha.krammer@oeaw.ac.at

Monte Carlo Simulation data for the CMS experiment are produced using two different software chains. Full Simulation, is a precise tool based on Geant4 detector simulation. The other tool, Fast Simulation, provides a faster but still reliable tool and is based on parametric particle-material interactions. Full Simulation for the LHC Run-3 has shown significant computing performance improvements compared to LHC Run-2. The challenging CMS detector upgrade plan for HL (High Luminosity)-LHC requires extra efforts due to the increased luminosity and a new and more complex detector geometry. Full Simulation plans to meet the requirements for HL-LHC, which includes continuous migration to newer versions of Geant4 as well as further physics improvements. In Fast Simulation, a more efficient treatment of the generator particles during their propagation through the detectors was achieved. The increasing use of machine learning (ML) techniques in simulation leads to an enhanced description of physics processes and detector responses, which reduces the need for computing capacity. This contribution reports the current Full and Fast Simulation performance innovations and developments to fulfill the significant higher Monte Carlo Simulation demands for HL-LHC. ML software tools already in use and new developments for Full and Fast Simulation and other promising simulation tools such as FlashSim are introduced.

12th Large Hadron Collider Physics Conference (LHCP2024)
3-7 June 2024
Boston, USA

*Speaker

1. Introduction

Future LHC runs will reach a new luminosity frontier and the experiments have to deal with huge data taking rates and pile ups. For the next experimental phase, the High-luminosity LHC, it is crucial to prepare the simulation software for the new challenges. Major upgrades of the CMS experiment [1] will be implemented, in particular the High Granularity Calorimeter (HGCAL) [2]. FullSim will need 2.2 times more run time due to the increased complexity of the geometry and envisaged higher measurement precision of physics parameters.

2. Improvements for Full and Fast Simulation

For Full and Fast Simulation, a wide range of improvements have been developed and introduced. New features were implemented in FullSim for Run-3 [3] and Phase-2 (HL-LHC runs) [4], as well as the migration to DD4hep geometry description and Geant4 to 11.1.2. In Fast Simulation (FastSim) the software and framework were optimized for a more efficient handling of the generated particles through the detectors. In addition, there is an ongoing effort of R&D of GPU usage for simulation such as Accelerated demonstrator of electromagnetic Particle Transport (AdePT) [5] and the Celeritas [6] project to provide complementary resources for the computationally intensive HL-LHC runs.

For Phase-2 the geometry description has been updated. There is a completely new geometry for the outer tracker, the Strip-Strip (2S) modules, which consist of two strip sensors and the Pixel-Strip (PS) modules, which are composed of a macro pixel and a strip sensor. New High Granularity Calorimeter (HGCAL) endcaps, new forward gas ionization detectors and the GEM detectors were also designed. The software framework CMSSW supports several scenarios for Phase-2. There are six major components which get modified independently: Tracker, Calorimeter, Muon detectors, Forward system, MIP Timing Detector (MTD) and Overall systems, which include the major partitions and the magnet system. The components are suitably combined to define scenarios of the Phase-2 configurations (D49, D76, D88, D95, D98, D110). The average CPU time evolution of the past 5 years (Figure 1) shows the success of the improvements for the standard model (SM) process $t\bar{t}$ for single threaded jobs [7–9].

3. Refining Fast Simulation using ML techniques

FastSim has a major advantage in speed, but compared to FullSim a decreased accuracy in some of the final observables. FastSim needs to develop tools to make it suitable for a large number of analyses in the collaboration. The refining method using ML techniques [10] was developed to improve accuracy. In the refining method, the analysis observables are simulated in the FastSim chains and the values are compared to the corresponding FullSim output. A fully-connected feed-forward neural network (NN) is trained, which results in a more accurate refined version of the FastSim data sample. The refining method focuses on jet flavor tagging for four DeepJet discriminators (B, CvB, CvL, QG) in the CMS NanoAOD data analysis format. The DeepJet algorithm [11] is a multi-class NN trained to distinguish jets originating from b, c, light quarks, and gluons. The application of the ResNet-like (Deep Residual Learning) architecture [12]

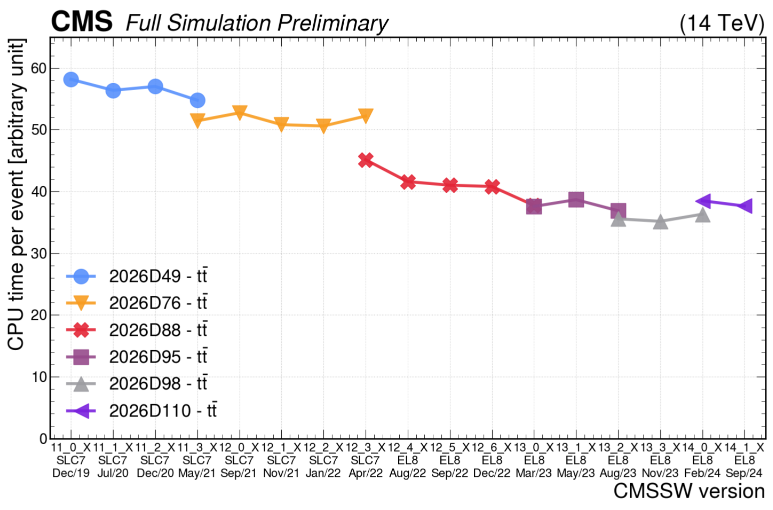


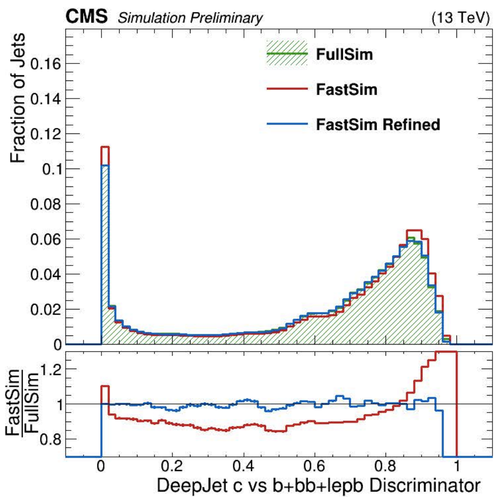
Figure 1: FullSim Phase-2 CPU time evolution. During the period of nearly 5 years the CPU time for the $t\bar{t}$ process has been improved by 35 % [8].

results in a good approximation of FastSim to FullSim output and needs only a residual correction. The network is implemented using the PyTorch package [13]. ResNet-like regression NN is used as post refinement layer to the FastSim output, which is already implemented in the FastSim software. This results in a considerably improved agreement with FullSim output, illustrated in Figure 2a.

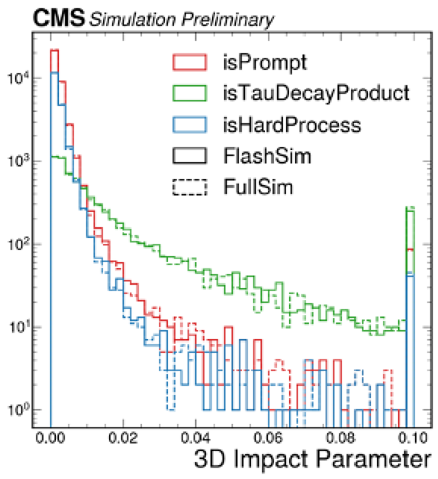
4. FlashSim - a ML simulation framework

FlashSim is a new simulation framework based on ML for a faster and more accurate simulation [14, 15]. New Normalizing Flows (NF) [16] use generic ML generative techniques, which directly produce CMS NanoAOD format samples from generator level information with the advantage to reduce the number of variables to simulate from several thousands down to few hundreds. The natural factorization is going successively through the various objects and to use generator level representation of those objects as conditioning information, which differs from typical AI/ML sample generation, e.g. image generation. The simulation of each object, the functional unit, is a transformation implemented by the NF algorithm. Only relevant physical information for the simulation of its target is taken into account and the various units are independent at first order. Additional correlation runs in a chain may be necessary to also access not only generator level information, but also reconstruction information of previous units. The advantage of this general, flexible simulator is to be not tailored to a specific analysis. As an example the goodness of conditioning comparing the impact parameter distributions for different status flags of the generated muons is verified in Figure 2b.

Conclusions The accelerating efforts for the simulation over the past 5 years for Phase-2 for FullSim yields in a strong reduction of the CPU time for SM processes, for $t\bar{t}$ by 35%. The agreement of the ML-based refinement with FullSim has been further improved with modest increase of the



(a) The distribution of the DeepJet discriminator Cvb(c vs (b+bb+lepb)) for FullSim, FastSim and the refined version of FastSim [10].



(b) Flash and Full Simulation comparison of the 3D impact parameter for muons from different processes generating the muon [14, 15].

Figure 2: Test sample results of Refining FastSim (a) using ResNet-like (Deep Residual Learning) architecture and FlashSim (b) using Deep Neural Network techniques.

simulation time. FlashSim continues to work on a complete ML simulation framework and shows a promising method to achieve a significant reduction of the simulation time.

References

[1] CMS Collaboration, The CMS experiment at the CERN LHC, JINST 3 S08004 (2008)

[2] CMS Collaboration, The Phase-2 Upgrade of the CMS Endcap Calorimeter, CERN-LHCC-2017-023, CMS-TDR-019 (2017)

[3] V. Ivanchenko, S. Banerjee, G. Hugo, S. L. Meo, I. Osborne, K. Pedro, D. Piparo, D. Sorokin, N. Srimanobhas, C. Vuosalo, CMS Full Simulation for Run 3, EPJ Web Conf. 251 (2021) 03016

[4] N. Srimanobhas, S. Banerjee, J. Hahnfeld, V. Ivantchenko, N. Krammer, M. Muzaffar, K. Pedro, D. Piparo, Full Simulation of CMS for Run-3 and Phase-2, EPJ Web of Conf. 295 03017 (2024)

[5] G. Amadio, J. Apostolakis, P. Buncic et al., Offloading electromagnetic shower transport to GPUs, J. Phys.: Conf. Ser. 2438 012055 (2023)

[6] S. C. Tognini, P. Canal, T. M. Evans et al., Celeritas: GPU-accelerated particle transport for detector simulation in High Energy Physics experiments, FERMILAB-FN-1159-SCD, arXiv 2203.09467 (2022)

[7] CMS Collaboration, CPU performance evolution of Full simulations using Phase-2 geometry from Run-3 CMSSW 11_0_4 to 14_0_1, CERN-CMS DP-2024/031 (2024)

[8] CMS Collaboration, Validation and monitoring of Geant4 simulation for CMS Phase2 detector configuration, CERN-CMS DP-2024/087 (2024)

[9] N. Krammer, Accelerating Full and Fast Simulation of the CMS Experiment, PoS(LHCP2023)286

[10] S. Bein, P. Connor, K. Pedro, P. Schleper, M. Wolf, Refining fast simulation using machine learning, EPJ Web of Conf. 295 (2024) 09032

[11] E. Bols, J. Kieseler, M. Verzetti, M. Stoye, A. Stakia, Jet flavour classification using DeepJet, JINST 15 P12012, arXiv:2008.10519 (2020)

[12] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770-778 (2016)

[13] A. Paszke, S. Gross, F. Massa et al., PyTorch: An Imperative Style, High-Performance Deep Learning Library, Part of Advances in Neural Information Processing Systems (NeurIPS) 32 pp. 8024–8035 (2019)

[14] F. Vaselli, A. Rizzi, F. Cattafesta, G. Cicconofri, FlashSim prototype: an end-to-end fast simulation using Normalizing Flow, CERN-CMS NOTE-2023/003 (2023)

[15] F. Vaselli, A. Rizzi, F. Cattafesta, G. Cicconofri, FlashSim: Accelerating HEP simulation with an end-to-end Machine Learning framework, EPJ Web of Conf. 295 (2024) 09020

[16] G. Papamakarios, E. Nalisnick, D.J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, Journal of Machine Learning Research 22(57) 1-64, arXiv:1912.02762 (2021)