# Accelerating 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 can be produced using two software tools. The first, the Full Simulation, is a more precise tool based on Geant4 detector simulation. The second, the Fast Simulation, provides a faster but still reliable tool and is based on parametric particle-material interactions. Full Simulation for the LHC Run-3 shows significant computing performance improvements compared to LHC Run-2 using the current Geant4 version (10.7.2), the new software package DD4hep for geometry description and the Vectorized Geometry run time library. A further optimization is achieved by the change of the computing platform operating system from CentOS 7 to Alma Linux 8. The challenging CMS detector upgrade plan for HL (High Luminosity)-LHC requires extra efforts due to the increased luminosity and the new and complex detectors geometry. Full Simulation plans to meet the requirements for HL-LHC with the new Geant4 version (11.1) as well as physics improvements including machine learning techniques to reduce compute capacity needs. Major progresses of Fast Simulation are reached by a more efficient treatment of the generator particles as they propagate through the detectors. Recent developments include the implementation of an increasing more accurate shower generation, improved track finding and tuning of physics processes. This contribution reports the current Full and Fast Simulation performance innovations and further plans to fulfill the significant higher Monte Carlo Simulation demands in LHC Run-3 and for HL-LHC.

*The Eleventh Annual Conference on Large Hadron Collider Physics (LHCP2023)*
*22-26 May 2023*
*Belgrade, Serbia*

*Speaker

PoS(LHCP2023)286

## 1. Introduction

Future LHC runs will reach new frontiers in energy and luminosity, which implicate increasing experimental data rate and high pileup interactions. For the next step the HL-LHC it is crucial to accelerate the simulation step. Due to the major upgrade of the CMS experiment [1], the high-granularity calorimeter (HGCAL), Full Simulation expects to be 3 times slower because of more complex geometry and more precise physics. Also the migration to CMS HL-LHC DD4hep geometry needs a speed up of the simulation time.

## 2. Accelerating Full and Fast Simulation

For Full and Fast Simulation a wide range of improvement options were analysed and introduced. New features in Full Simulation were implemented in Run-3 and Phase-2 [2] [3]: (i) the migration to DD4hep geometry description, (ii) the use of the LTO (Like Time Optimization) build method, (iii) the use of faster computations and less instructions: Geant4 Gamma General Process and VDT (VectorisD maTh) for fast and auto-vectorisable mathematical functions, (iv) the Geant4 Transportation with MSC (multiple scattering), (v) custom tracking managers to simplify the e-gamma transport in Geant4, (vi) G4HepEm external library, which focuses on the EM shower generation for GPU usage. In addition the operating systems was upgraded from CentOS 7 (SLC7) to Alma Linux 8 (EL8). The Geant4 version was migrated from 10.4 to 10.7 (CMSSW 11_3_X) and to 11.1.1 (CMSSW 13_1_X). Fast Simulation software and framework optimization achieves a more efficient handling of the generator 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) [4] und the Celeritas [5] project targeting the computationally intensive HL-LHC runs. The success of the improvements are shown in the evolution for the past 4 years in Figure 1 for the average CPU time per event for the standard model processes $t\bar{t}$ and the beyond standard model process BSM T1tttt (pp $\rightarrow$ gluino + gluino, gluino $\rightarrow$ $t\bar{t}$ + lightest neutralino) for single threaded jobs [6].

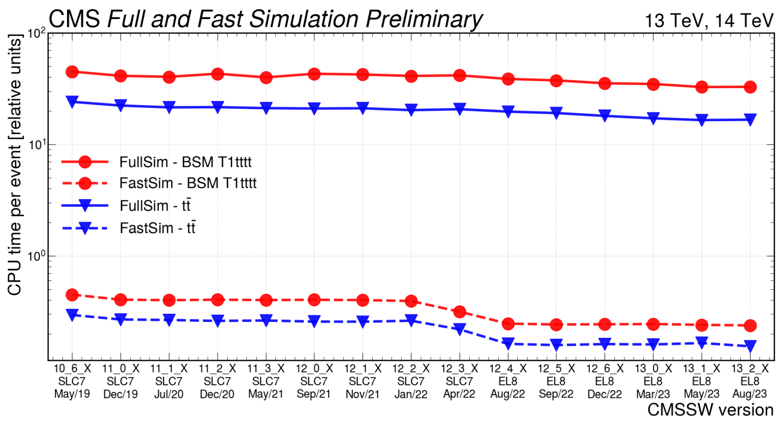

**Figure 1:** Full and Fast Simulation CPU time evolution for the past 4 years. Fast Simulation compared to Full Simulation is for BSM T1tttt 136 times and for $t\bar{t}$ 110 times faster using the latest CMSSW version.

## 3. Refining Fast Simulation using Machine Learning (ML) techniques

Fast Simulation (FastSim) compared to Full Simulation (FullSim) has a major advantage in speed with the price of decreased accuracy in some of the final observables. The refining method using ML techniques [7] was developed to increase the accuracy and promote FastSims wider usage. The refining method is to use the analysis observables simulated by the FastSim chains and compare them to the corresponding FullSim output. A fully-connected feed-forward neural network is trained to establish a refined version of the FastSim data sample, which is more accurate to the FullSim sample. The current refining FastSim method focus on jet flavor tagging for four DeepJet discriminators (B(b+bb+lepb), CvB(c/(c+b+bb+lepb)), CvL(c/(c+uds+g)), QG(g/(g+uds))) in the CMS NanoAOD data analysis format. The DeepJet algorithm [8] is a multi-class neural network trained to distinguish jets originating from b, c, light quarks and gluons. The network has six output nodes (b, bb, lepb, c, uds, g) activated with a softmax function.

As training sample the beyond standard model process BSM T1tttt is simulated with FastSim and FullSim. The jets, which match the $\delta R$ angular criterion result in the jet triplets (Generation, FastSim, FullSim). The application of the ResNet-like (Deep Residual Learning) architecture [9] results in a good approximation of FastSim to FullSim output and needs only to apply a residual correction. The pre-processing transforms the input variables/parameters and the post-processing transforms back and enforce DeepJet constraint that the refined DeepJet discriminators (B, CvB, CvL, QG) have to be constructed such that the sum of the original DeepJet output nodes (b, bb, lepb, c, uds, g) is equal to unity. The network is implemented using the PyTorch package [10]. Loss terms have to be considered, the primary loss MMD (distributed-based) compares ensembles of jets not jet-jet pairs to cope with independent stochastic in both simulations chains and an additional loss term MSE/Huber (output-target pair-based) for the correction for deterministic FastSim biases. The two loss terms are combined via MDMM algorithm. ResNet-like regression NN can be used as post-hoc refinement layer to FastSim output, which results in a considerably improved agreement with FullSim output, illustrated in Figure 2.

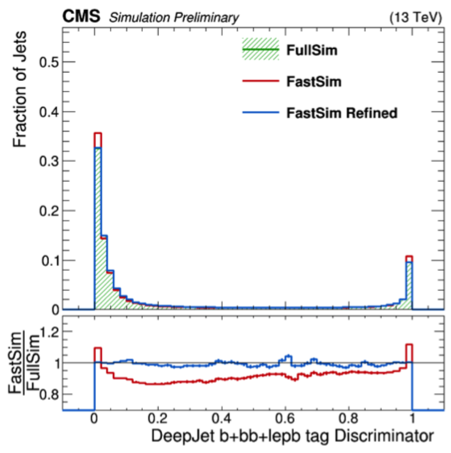

**Figure 2:** The distribution of the DeepJet discriminator B(b+bb+lepb) for FullSim, FastSim and the refined version of FastSim.

## 4. FlashSim - a ML simulation framework

A new simulation framework, named *FlashSim* [11] uses the advantage of ML for a faster and accurate simulation. It is based on new Normalizing Flows (NF) [12] using generic ML generative techniques, which directly produce CMS NanoAOD format samples from generator level information. The advantage of the NanoAOD format for analysis is to reduce the number of variables to simulate from several thousands down to few hundreds. The difference to typical AI/ML sample generation e.g. image generation, is that we need to condition the generation on some previous information. Not *a generic* CMS event should be simulated, but *the* CMS event corresponding to some given generator level input. For the simulation the natural factorization is to go one by one on the various objects and to use generator level representation of those objects as conditioning information. The simulation of each object is a so-called functional unit, which is a transformation implemented by the Normalizing Flows algorithm. As input only the relevant physical information for the simulation of its target is taken into account and the various units are independent at first order. It may still be necessary to include additional correlation runs in a chain 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.

In a real-world scenario analysis test the feasibility of the model is performed with generated datasets of $t\bar{t}$, Drell-Yan, EWK LLJJ and signal ($H \to \mu^+\mu^-$) processes. Higgs to di-muon VBF (vector boson fusion) analysis [13] uses events with muons and jets simulated by FlashSim to verify the usability of the approach at the analysis level. The flash-simulated quantities are calculated using Deep Neural Network (DNN) to separate signal from background and are compared with FullSim results. Multiple derived input variables are used by DNN, including some that are correlating the di-jets part of the VBF event with the di-muon part of the Higgs decay. The consistency of FlashSim and FullSim results are shown in Figure 3.

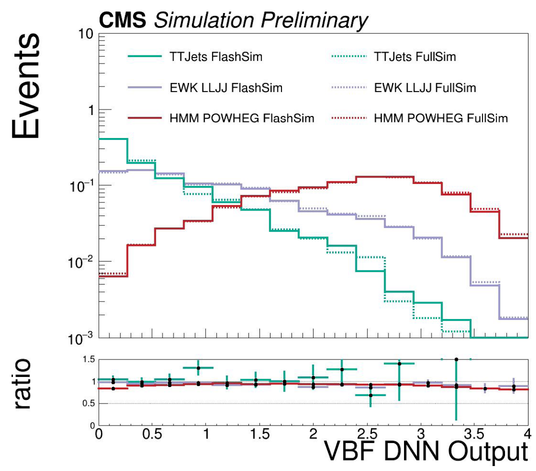

**Figure 3:** Flash and Full Simulation comparison of analysis DNN output using muons and jets information of $t\bar{t}$ and EWK LLJJ samples (same process used in training, different events) and signal events (not seen during training).

## References

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

[2] 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 03016 (2021)

[3] N. Srimanobhas, S. Banerjee, J. Hahnfeld, V. Ivantchenko, N. Krammer, S. Muzaffar,, K. Pedro, D. Piparo, Full Simulation of CMS for Run-3 and Phase-2, CERN-CMS-CR-2023/152 (2023)

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

[5] 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)

[6] CMS collaboration, CPU performance evolution of Full and Fast simulations from Run-2 to Run-3/CMSSW_13_2_0, CERN-CMS-DP-2023-063 (2023)

[7] S. Bein, P. Connor, K. Pedro, P. Schleper, M. Wolf, Refining fast simulation using machine learning, CERN-CMS-CR-2023/128 (2023)

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

[9] 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)

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga 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)

[11] F. Vaselli, FlashSim: accelerating HEP simulation with an end-to-end Machine Learning framework, CERN-CMS-CR-2023/090 (2023)

[12] 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)

[13] A. Sirunyan et al. (CMS), Evidence for Higgs boson decay to a pair of muons, JHEP 01 148, arXiv:2009.04363 (2021)