Accelerating CMS Matrix-Element Event Generation for Drell-Yan and Top Pair Production with Madgraph4GPU

Jin Choi[1,*] and *Saptaparna* Bhattacharya[2,**]

on behalf of the CMS Collaboration

[1]Department of Physics & Astronomy, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Korea
[2]Department of Physics & Astronomy, Southern Methodist University, 6425 Boaz Lane, Dallas, Texas 75205, United States of America

Abstract. In this note, we present the first analysis of the computational speedup achieved using the GPU version of Madgraph, known as Madgraph4GPU (MG4GPU), in the CMS workflow. Madgraph is one of the most widely used event generators in CMS. This work represents the initial step toward benchmarking the improvements offered by both the GPU and vectorized CPU implementations. We demonstrate timing improvements across a broad range of physics processes relevant to CMS. Speedups are quantified for both gridpack production and event generation. A gridpack is a pre-defined package that encapsulates all the necessary components for effectively executing Monte Carlo event simulations, eliminating redundant computations of common elements for each event. Preliminary results indicate a speedup of approximately a factor of three with vectorized CPUs and an order-of-magnitude improvement with GPUs in gridpack production for the Drell-Yan and top quark pair production processes. For event generation, we observe a speedup of 1.5 times with vectorized CPUs and 7 times with GPUs when generating 10^5 events. These workflows were tested using a variety of computational resources, including CUDA-enabled NVIDIA GPUs and modern vectorized CPUs from Intel and AMD, accessible via CERN resources and HPCs.

1 Introduction

A new era of precision-focused collider physics has begun, driven by the LHC's unprecedented accuracy in measuring the W boson mass and related Standard Model parameters. However, the precision program comes at the cost of computing resources. In a typical sample production workflow in CMS, the first step is the simulation of a physics process, also referred to as event generation. Current generator workflows account for 9% of the total CPU usage as shown in the pie-chart in Figure 1. In the HL-LHC era, huge amounts of resources will need to be allocated for Monte Carlo (MC) generation, given the sheer amount of data that will be collected. To keep statistical uncertainties from MC simulations low, the production of sample sizes that are at least ten times larger than the amount of data is imperative.

*e-mail: jin.choi@cern.ch
**e-mail: saptaparnab@smu.edu

The CPU time requirements for the HL-LHC will grow rapidly as shown in the left panel of Figure 1 and to achieve the goals of the precision program, physics processes at high accuracy will need to be generated. Such workflows will lead to an increase in the CPU fraction for the generation step.

Over the last few years, substantial effort and resources have been invested into workflow optimization leading to the implementation of multithreading in the event generation step of the production workflow. Studies of the usage of GPUs for Matrix Element computation have begun. Using the GPU compatible version of Madgraph (the GPU version of aMC@NLO is in progress) and generating events with additional partons, significant improvements in the matrix element computation leading to a lower time per event have been observed [3–7]. In this article, we present the first benchmarking studies with MG4GPU in MC sample production workflows that are widely used in CMS.

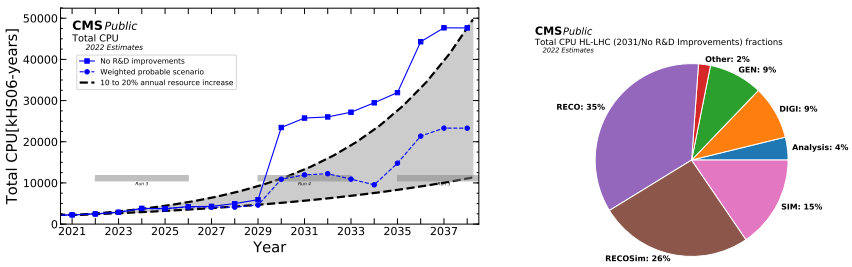

Figure 1. Left: CPU (disk) usage projected for the HL-LHC, where the CPU time requirements is quoted in kilo-HEPSpec06 years and provides an estimation of resources to be required annually for CMS processing and analysis needs. Right: The fraction of CPU time spent on each part of the simulation chain in CMS [1]. The event generation step is referred to as GEN (currently accounting for 9%), the simulation step is abbreviated as SIM, digitization as DIGI, reconstruction as RECO respectively. The analysis step refers to the production of files for physics analysis. For details on each component of the CMS MC production process are outlined in Ref. [2].

2 Event generators on GPUs

Matrix element calculations, which are usually the most CPU intensive part of higher order MC event generator computations, can be offloaded to GPUs. The CMS Collaboration is a major consumer of Madgraph. In this study, we chose processes that are widely used in CMS. These processes are Drell-Yan and top-pair (TT) production. The GPU version of Madgraph or MG4GPU currently includes implementation of leading order process. In order to align with CMS MC generation settings, the Drell-Yan (TT) process was generated with four (three) additional emissions or jets.

3 Computing architecture

We utilized a variety of computing resources with different architectures to benchmark the performance of MG4GPU. All tests were performed in isolated environments using appropriate batch systems to ensure consistent and reproducible results.

Several high-performance computing (HPC) platforms were used for our benchmarking tests. The CERN lxplus HTCondor pool [8] provided access to heterogeneous computing

nodes. These included AMD EPYC 7313 16-core processors paired with NVIDIA A100 GPUs, Intel Xeon Platinum 8468 processors with NVIDIA H100 GPUs, and CPU-only nodes equipped with Intel Xeon Silver 4216 processors. The Seoul National University HTCondor pool provided additional CPU resources with Intel Xeon E5-2698 v4 processors. To make the production of CPU based gridpacks faster, we utilized the NERSC Perlmutter supercomputer [9], which features AMD EPYC 7763 64-core processors and is managed through the SLURM batch system.

To ensure consistency across different computing platforms, we employed a standardized software environment. HTCondor and SLURM batch systems were used to isolate jobs and ensure reproducible performance measurements. A singularity container based on the EL8 image from OSG [10] provided a consistent operating system environment. At the time of this study, there was no publicly available cudacpp repository for MG4GPU [11], necessitating the use of the development version. Madgraph v5.3.6.0 was used in conjunction with the development repository of MG4GPU. The software was integrated into the centralized CMS event generator code repository [12] with additional modifications specific to our workflow. The software stack included EL8, GCC 12, and NVIDIA CUDA compiler (NVCC) version 12.2.

4 Process definitions

Two physics processes that are commonly used as backgrounds in CMS analyses are utilized in this study: Drell-Yan (DY) and top quark pair production (TT) at leading-order accuracy in quantum chromodynamics (QCD). These processes were chosen for their complementary characteristics and relevance to CMS physics program.

The DY process was selected because it is one of the most relevant background processes in many CMS analyses, from Standard Model tests to searches for new physics. Its clean dilepton final state makes it essential for detector calibration and performance studies. Furthermore, by adding jets to the final state, we can easily manage the complexity of the process, providing a scalable test environment with increasing computational demands. This property allows us to benchmark performance across different levels of computational complexity. Additionally, DY has no external dependencies in our implementation, allowing us to reuse the central production settings (or Madgraph input cards) without modification.

The TT process was included because previous studies by the Madgraph development team had indicated significant performance improvements when running this process on GPUs. Our goal was to verify these improvements within the context of the CMS production environment. Since TT processes involve mostly QCD interactions at leading order event generation, we can isolate our benchmarking from other types of interactions.

All production cards used in this study were synchronized with the CMS Run 3 production setup to ensure consistency with current CMS workflows. It should be noted that Madspin functionality was not included in our tests, as it is not yet supported by the MG4GPU implementation. This limitation does not affect the core matrix element calculations that are the focus of our benchmarking study.

5 First benchmarking studies

5.1 Experiment setup

The benchmarking studies follow the standard CMS hard scattering calculation procedure [2], which consists of two main steps, gridpack production and event generation. For each step, we compare the computational performance across different backend implementations of the MG4GPU cudacpp madevent_gpu plugins.

Table 1. Number of Feynman diagrams and processes for Drell-Yan with varying jet multiplicities, illustrating the increasing computational complexity. Here, "diagrams" refers to the number of Feynman diagrams for each process, while "processes" refers to the group of diagrams with the same initial and final state that will be merged in the actual gridpack production.

	DY+0j	DY+1j	DY+2j	DY+3j	DY+4j	DY+4j (Simplified)
Diagrams	30	180	3120	27600	412560	9856
Processes	15	45	285	435	1455	13

Table 2. Number of Feynman diagrams and processes for top quark pair production with varying jet multiplicities. The +*n*j, where *n*=0, 1, 2, 3, and 4 refers to the numbers of additional partons, which hadronize to jets, that are produced in association with the DY and TT processes.

	TT+0j	TT+1j	TT+2j	TT+3j
Diagrams	8	91	1473	17660
Processes	6	16	96	146

- FORTRAN: Legacy fortran code implementation without CPU vectorization
- CPP-AVX2: A vectorized CPU implementation using Advanced Vector Extensions 2 (AVX2)
- CUDA: The GPU implementation utilizing NVIDIA CUDA framework

Our benchmarking approach includes both inclusive and jet-binned production studies. The inclusive production represents the standard physics workflow used in CMS production, providing direct measurements of speedup achievable in actual analysis conditions. Additionally, we study processes with increasing numbers of final-state jets to systematically evaluate performance scaling and identify potential bottlenecks as the computational complexity increases.

A limitation in our comparative analysis arises from the implementation differences between the traditional Madgraph and MG4GPU. Standard Madgraph employs a technique called "parton grouping," which combines partons with the same properties, such as leptons or quarks, to avoid recalculating identical matrix elements with different partons. This optimization is not implemented in the development version of the MG4GPU cudacpp plugin used in this study. To isolate the performance improvements from the GPU and vectorized CPU implementations without being affected by these structural differences, we focused our comparisons solely on switching backends while keeping all other aspects of the calculation identical.

The computational complexity of the matrix element calculation grows significantly with the number of final-state particles. Table 1 and Table 2 show the number of Feynman diagrams and processes for Drell-Yan and top quark pair production with increasing jet multiplicity. For the most complex case studied, DY+4j, the number of diagrams exceeds 400,000, presenting a substantial computational challenge. Due to the computational demands of the full DY+4j process, we used a simplified configuration for our actual testing rather than the complete DY+4j process. For the simplified DY+4j configuration, we restricted the initial and final state quarks to only up quark and up antiquark($u, \bar{u}$) and the final state leptons to electron-positron pairs (e^+, e^-).

5.2 Gridpack production

Gridpack production represents the first computational step in the CMS Monte Carlo event generation workflow, where matrix elements for all relevant processes are calculated and

Table 3. Timing for gridpack production of inclusive processes. As varying levels of parallelism were used for producing gridpacks, the reported times are normalized to a configuration using 16 madevents in parallel. Results taken from CMS-DP-2024-086 [14]

Production Time			
Process	FORTRAN	CPP-AVX2	CUDA
DY+0j	7m	6m	5m
DY+1j	10m	10m	12m
DY+2j	1h 12m	1h 10m	51m
DY+3j	22h 40m	9h 4m	4h 18m
DY+4j (Simplified)	440h 46m	141h 20m	9h 17m
DY+01234j (Simplified)	424h 36m	133h 38m	9h 32m
TT+0j	6m	7m	5m
TT+1j	11m	11m	7m
TT+2j	1h 15m	38m	22m
TT+3j	262h 11m	79h 19m	3h 4m
TT+0123j	253h 36m	155h 28m	3h 9m

stored for later use in event generation. The computational requirements for this step increase dramatically with process complexity, particularly with higher jet multiplicities.

For a comprehensive performance evaluation, we measured gridpack production times across different computational backends for both individual jet bins and inclusive processes. Table 3 presents timing measurements for each processes. For the Drell-Yan inclusive process, the CUDA implementation on an A100 GPU achieves a remarkable factor of 45 speedup compared to the traditional FORTRAN backend. The vectorized CPU implementation (CPP-AVX2) also shows significant improvement with a speedup of a factor of three.

The acceleration is even more pronounced for the inclusive top quark pair production process, where the CUDA implementation achieves a speedup by a factor of 85 compared to the FORTRAN backend. It's important to note that this speedup can be achieved with the same level of parallelism. However, this dramatic acceleration comes with hardware considerations. The dominant subprocess in TT production, $gg \to t\bar{t}ggg$, requires approximately 6 GB of GPU memory, which can constrain parallel execution on GPUs. By contrast, CPU-only HPC environments often provide larger memory capacities (around 100 GB), allowing for higher degrees of parallelism with more concurrent processes.

5.3 Event generation

Event generation is the second computational step in the CMS Monte Carlo workflow, where collision events are produced using the matrix elements stored in the gridpack. This step is typically less computationally intensive than gridpack production but still represents a significant computational load when generating large datasets.

For benchmarking event generation performance, we adopted a systematic approach with increasing event counts to evaluate how each backend scales with larger workloads. Each process was tested with various event counts (5×10^3, 10^4, 2×10^4, 5×10^4, 10^5, and 2×10^5) using a single computational core to provide a clear comparison of raw performance. Tests were conducted on CPU-only nodes for the FORTRAN and CPP-AVX2 implementations, while GPU-enabled nodes were used for the CUDA backend tests.

To ensure statistical significance and account for potential system variability, each test configuration was executed eight times in parallel. The mean execution time and standard

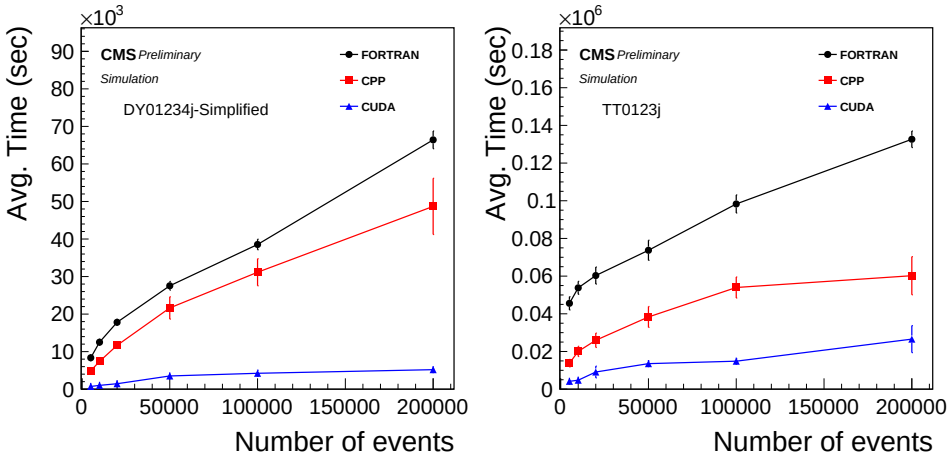

Figure 2. Event generation performance comparison across different computational backends. Left: Drell-Yan process. Right: Top quark pair production process. In both cases, the CUDA implementation (blue) significantly outperforms the traditional FORTRAN backend (black), while the vectorized CPU implementation (CPP-AVX2, red) provides intermediate performance. The 0-4 or 0-3 refer to the total number of additional partons, which hadronize to jets, that are generated in association with the DY and TT processes. Results taken from CMS-DP-2024-086 [14]

deviation were calculated to provide robust performance metrics. This approach allows us to quantify both the absolute performance and the consistency of each backend.

The results, as shown in Figure 2, demonstrate significant performance differences between the backends. For both Drell-Yan and top quark pair production processes, the CUDA implementation achieves substantial speedups compared to the traditional FORTRAN backend, while the CPP-AVX2 implementation provides moderate improvements. When generating 100K events, we observe a speedup of approximately 1.5 times with the vectorized CPU implementation and 7 times with the GPU implementation compared to the traditional FORTRAN backend. This trend continues with larger event counts, making the CUDA backend particularly advantageous for large-scale event generation tasks required for CMS physics analyses.

The improved performance of both the CPP-AVX2 and CUDA implementations can be attributed to their efficient utilization of modern computing architectures. The CPP-AVX2 implementation leverages vector instructions available in modern CPUs, while the CUDA implementation exploits the massive parallelism offered by GPUs. These improvements are particularly beneficial for CMS workflows that require generating billions of events for physics analyses.

6 Summary and future directions

We have presented the first benchmarking of event generator workflows using MG4GPU in CMS with widely used physics processes. We have observed significant improvements in computing time, up to a factor of 85 in gridpack production and a factor of seven in event generation for top-pair production. The eventual goal of this study is to produce events with MG4GPU for data analysis in CMS, replacing traditional CPU-based workflows for large scale event generation.

References

[1] CMS Offline and Computing public results, https://twiki.cern.ch/twiki/bin/view/CMSPublic/CMSOfflineComputingResults

[2] G. Boudoul et al., *Monte Carlo Production Management at CMS*. PoS, ICHEP2015, **vol. 664**, 072018 (2015). https://doi.org/10.1088/1742-6596/664/7/072018

[3] J. Alwall et al., *Madgraph 5 : Going Beyond*. JHEP06(2011)128. https://doi.org/10.1007/JHEP06%282011%29128

[4] S. Hageboeck et al., *Madgraph5_aMC@NLO on GPUs and vector CPUs Experience with the first alpha release*, 26th International Conference on Computing in High Energy and Nuclear Physics. https://doi.org/10.48550/arXiv.2312.02898

[5] A. Valassi et al., *Speeding up Madgraph5_aMC@NLO through CPU vectorization and GPU offloading: towards a first alpha release*, 21th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: AI meets Reality. https://doi.org/10.48550/arXiv.2303.18244

[6] Z. Wettersten et al., *Acceleration beyond lowest order event generation: An outlook on further parallelism within MadGraph5_aMC@NLO*, 26th International Conference on Computing in High Energy and Nuclear Physics. https://doi.org/10.48550/arXiv.2312.07440

[7] A. Valassi et al., *Madgraph on GPUs and vector CPUs: towards production*, in Proceedings of the 27th International Conference of Computing in High Energy and Nuclear Physics

[8] CERN lxplux batch system, *site documentation*. https://batchdocs.web.cern.ch/index.html

[9] NERSC Perlmutter, *site documentation*. https://www.nersc.gov/about

[10] Open Science Grid, *site documentation*. https://osg-htc.org/docs/

[11] madgraph4gpu, *github code repository*. https://github.com/madgraph5/madgraph4gpu

[12] CMS genproduction, *github code repository*. https://github.com/cms-sw/genproductions

[13] A. Valassi et al. *Development in Performance and Portability for Madgraph5_aMC@NLO*. PoS, ICHEP2022, **vol. 414**, 212 (2022). https://doi.org/10.22323/1.414.0212

[14] CMS Collaboration, *Quantifying the computational speedup with madgraph4gpu for CMS workflow*, CMS Detector Performance Summary, CMS-DP-2024-086, October 2024. https://cds.cern.ch/record/2914584/files/DP2024_086.pdf