Heterogeneous Particle Flow Algorithms for Event Reconstruction in CMS

Andrea Bocci[1], Abhishek Das[2], Kenichi Hatakeyama[3], Florian Lorkowski[4], Marino Missiroli[5], Felice Pantaleo[1], Jonathan Samudio[3], Mark Saunders[3],
on behalf of the CMS collaboration

[1]CERN, European Organization for Nuclear Research, Meyrin, Switzerland
[2]University of Notre Dame, Notre Dame, Indiana, USA
[3]Baylor University, Waco, Texas, USA
[4]Deutsches Elektronen-Synchrotron, Hamburg, Germany
[5]Paul Scherrer Institut, Villigen, Switzerland

E-mail: jonathan_samudio1@baylor.edu

Abstract. The higher instantaneous luminosity expected during the upcoming years of LHC Run 3 operations will impose challenges for CMS event reconstruction, and this will be amplified in the HL-LHC era, where luminosity and pileup rates are expected to be significantly higher. One of the approaches for CMS to cope with this challenge is to utilize heterogeneous computing architectures in order to accelerate event reconstruction. In this presentation, we discuss an effort to utilize GPU accelerators for particle flow (PF) reconstruction. The PF algorithm, used for the vast majority of CMS data analyses for event reconstruction, provides a comprehensive list of final-state particle candidates and enables efficient identification and mitigation methods for simultaneous proton-proton collisions (pileup). The PF algorithm consists of multiple steps, and the clustering of calorimeter hits is one of its most time-consuming steps. As a first step toward accelerated PF reconstruction, a GPU version of PF clustering for hadronic showers has been developed. The cluster outputs and computational performance of the GPU-accelerated algorithm are compared to those of the CPU-based implementation.

1. Introduction

In response to increasing collision rates during Run 3 of the LHC and in preparation for the HL-LHC era, the CMS experiment [1] is in the process of optimizing data processing and event reconstruction. The High-Level Trigger (HLT) system [2] is used to filter the input data stream from the detector; therefore, the requirements on processing time are particularly strict. The Particle Flow (PF) algorithm [3] performs global event reconstruction using information from all CMS subdetectors to produce final-state particle candidates for each event, and it plays a critical role in the event reconstruction for CMS. The use of heterogeneous computing, in which some computations are performed on a Graphical Processing Unit (GPU), has been investigated to accelerate PF for Run 3 data taking and further increased luminosity in future runs of the LHC.

The simplified PF workflow is outlined in Fig. 1. The PF reconstruction utilizes two main classes of inputs: (1) charged particle tracks in the inner tracker and the muon system, which

are used to form *PF tracks*, and (2) calorimeter hits (*rechits*), *i.e.*, energy deposits in the electromagnetic (ECAL) and hadron calorimeters (HCAL), which are used to form *PF rechits* and subsequently *PF clusters*, *i.e.*, collections of PF rechits associated with a common particle shower in the calorimeters. The tracks and calorimeter clusters are then combined into blocks which are processed to produce the final PF candidates. It is critical for PF reconstruction to associate charged particle tracks to corresponding showers in calorimeters in order to avoid double-counting of contributions from the same particles, and reconstruction of calorimeter clusters is an integral part of this association. The PF cluster production is relatively slow on a CPU, currently taking up about half of the PF processing time and a few percent of the total HLT processing time. Since the reconstruction of hits in the CMS ECAL and HCAL is already performed on GPU at HLT [4], reconstruction of PF rechits and clusters can be performed using input calorimeter hits already on GPU without the need to transfer data from CPU to GPU and has been considered as one of the areas for GPU acceleration.

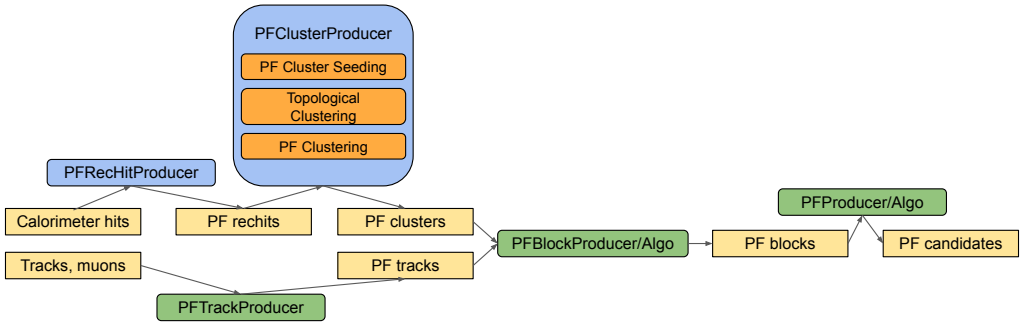

Figure 1: Diagram of the PF workflow. Yellow blocks correspond to the different data stages along the process. Blue and green blocks are the individual modules of the PF algorithm, where the modules in blue are the current target of GPU acceleration in CUDA.

2. Workflow to Produce PF Clusters

The GPU acceleration of PF clustering for hadronic showers using CUDA has been the initial target of optimization for PF. The general workflow in PF to process calorimeter rechits and produce PF clusters, which is common in both the CPU and GPU implementations, are discussed below.

PF RecHit Production: Input calorimeter rechits are initially used to produce PF rechits by selecting rechits passing energy thresholds in order to suppress hits arising from detector noise. We also associate geometry information to each PF rechit, including references to rechits in neighboring calorimeter cells, which assists subsequent PF clustering steps.

PF Cluster Seeding: The first step in the production of PF clusters is the identification of cluster seeds. The cluster seeds are identified based on the local energy maxima of neighboring input PF rechits. Hadronic (HCAL) and electromagnetic (ECAL) rechits use calorimeter-dependent seeding conditions [3].

Topological Clustering: Topological clusters are sets of connected PF rechits where all cells have a common side, as shown in Fig. 2 (left). The resulting topological clusters could have multiple cluster seeds and are fed into the main PF clustering algorithm where the final PF cluster energy and position are determined.

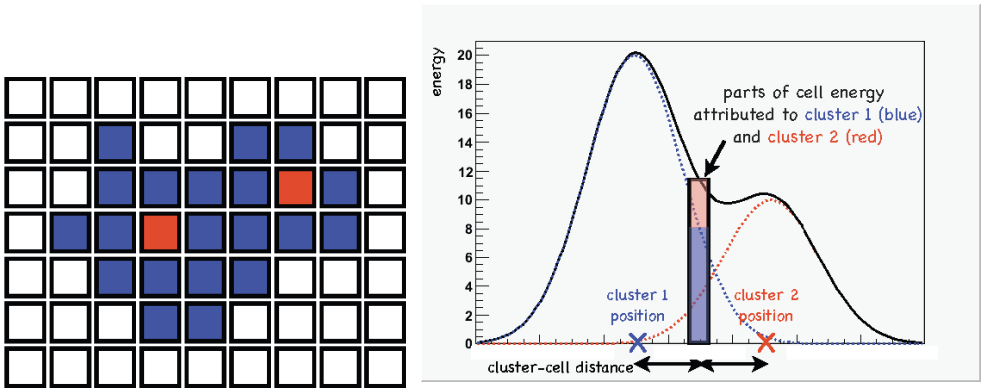

Figure 2: (Left) Diagram representing a topological cluster containing multiple seeds, shown in red. Each seed will become its own PF cluster. (Right) Example of energy sharing between two clusters.

PF Clustering (Energy and Position Determination): The last step of clustering is the formation of output PF clusters. This step takes input topological clusters and applies an expectation-maximization algorithm based on a Gaussian-mixture model to each topological cluster [3]. The model assumes that the energy from M topological cluster cells results from N Gaussian energy deposits, where N is the seed multiplicity. Each PF rechit will contribute some fraction (f_{ji}) of its energy to each cluster in the case of multiple seeds. This fraction is calculated based on the rechit distance to the cluster position as illustrated in Fig. 2 (right) and can be expressed by:

$$f_{ji} = \frac{A_i \mathrm{e}^{-(\vec{c}_j - \vec{\mu}_i)^2/(2\sigma)^2}}{\sum_{k=1}^{N} A_k \mathrm{e}^{-(\vec{c}_j - \vec{\mu}_k)^2/(2\sigma)^2}}, \tag{1}$$

$$A_i = \sum_{j=1}^{M} f_{ji} E_j, \vec{\mu}_i = \sum_{j=1}^{M} f_{ji} E_j \vec{c}_j. \tag{2}$$

where E_j and c_j are the energy and position of cell j, A_i and μ_i are the cluster energy and position, and σ is the assumed cluster Gaussian width. The cluster position and energy estimation are iterated with this model until convergence.

3. Implementations with CUDA

The GPU version of the PF rechit and cluster producers have been developed using CUDA, keeping the underlying clustering algorithm logic the same as the CPU version [5]. These producers utilize the data in the Structure-of-Array (SoA) format which is compatible with the highly parallel architecture of GPUs. The CUDA implementations of various steps for producing PF rechits and clusters are presented below.

PF RecHit Production and Cluster Seeding: The PF rechit producer and PF cluster seed finder both lend themselves well to parallelism. In both steps, rechits are assigned to individual threads of a GPU and the computation can be performed in parallel.

Topological Clustering: The topological clustering can be performed with a connected-component labeling algorithm which has been widely studied in the context of computer vision and image recognition. We adopted one implementation of the connected-component labeling algorithm, ECL-CC [6], which has been specifically designed for acceleration with CUDA. The algorithm operates on graph data consisting of vertices, v, edges made from neighboring vertices (u, v), and labeling information. ECL-CC is unique in its implementation of a path-halving technique for connected vertices and asynchronous edge processing for use in CUDA. The edge processing is split into three kernels which operate at thread level, warp level, and block level granularity for increasing degrees of vertices. A flattening step is performed at the end of the algorithm to reduce all paths to a single layer of connection, and these are the output topological clusters. This implementation of topological clustering improves on previous methods due to its asynchronous parallel operation and built-in minimization of load imbalance through the use of multiple edge processing kernels.

PF Clustering: The PF clustering runs the expectation-maximization algorithm based on a Gaussian-mixture model iteratively on each topological cluster of multiple PF rechits in order to find stable solutions for PF clusters. We assign a single block of GPU to process each topological cluster and GPU threads within a block are used to process rechits in each topological cluster. Using a single block per topological cluster allows for the use of shared memory in all iterations involved in determining PF cluster kinematics.

4. Results

The performance evaluation of the GPU-based PF clustering workflow has been performed by validating cluster outputs and event throughput against the established CPU version on Run 3 data from 2022. The use of the GPU algorithms at HLT would be highly motivated, if it does not saturate GPU resources available and increases the average event throughput. Fig. 3 shows the event throughput of three different workflows, where HCAL reconstruction and the PF clustering are performed entirely on the CPU, entirely on the GPU, or with HCAL reconstruction on the GPU and PF clustering on the CPU. There is a factor 3 improvement between running a workflow on 1 GPU vs 64 CPU cores, when a machine with one NVIDIA Tesla T4 GPU and two Intel "Skylake" Xeon Gold 6130 processors is used for comparisons. PF cluster output validation is shown in Fig. 4 where the computed PF cluster energy is compared between the CPU and GPU versions for the same dataset and shows a good agreement.

5. Conclusions and outlook

The GPU acceleration of particle flow reconstruction has been explored by the CMS Collaboration. The first implementation targets the processing of HCAL rechits into PF clusters. The GPU based PF clustering has shown a factor 3 improvement over the CPU implementation, when 1 GPU and 64 CPU cores are used for comparisons. Similar GPU acceleration can be used for other aspects of the particle flow algorithm such as ECAL rechit clustering. Moving forward, these optimizations will be written using a flexible portability library to be used on a variety of hardware configurations.

References

[1] CMS Collaboration 2008 *JINST* **3** S08004 URL `https://dx.doi.org/10.1088/1748-0221/3/08/S08004`
[2] CMS Collaboration 2017 *JINST* **12** P01020 URL `https://arxiv.org/abs/1609.02366`
[3] CMS Collaboration 2017 *JINST* **12** P10003 URL `https://arxiv.org/abs/1706.04965`
[4] CMS Collaboration 2021 URL `https://cds.cern.ch/record/2759072`
[5] CMS Collaboration 2022 URL `https://cds.cern.ch/record/2842374`
[6] Jaiganesh J and Burtscher M 2018 URL `https://doi.org/10.1145/3208040.3208041`

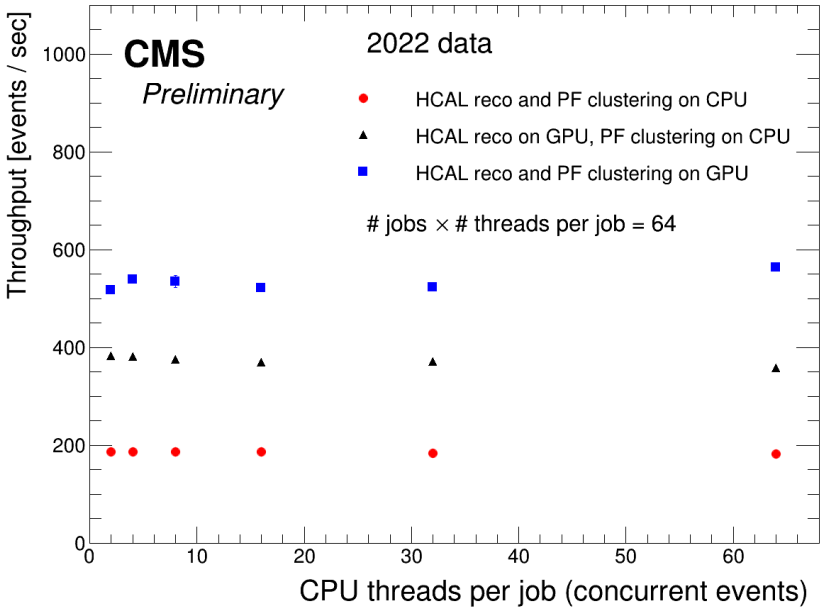

Figure 3: The throughput in events per second for three workflows running the HCAL local reconstruction and PF reconstruction, either on CPU or on GPU, measured using Run 3 2022 data: (red) both HCAL local reconstruction and PF clustering on CPU, (black) HCAL local reconstruction on GPU and PF clustering on CPU, and (blue) both HCAL local reconstruction and PF clustering on GPU. These measurements are performed on a machine with one NVIDIA Tesla T4 GPU and two Intel "Skylake" Xeon Gold 6130 processors (64 logical cores and hardware threads in total) and with the PF clustering configuration used for the current offline reconstruction. The different points correspond to measurements with a different number of jobs and threads per job, keeping the total number of threads constant.

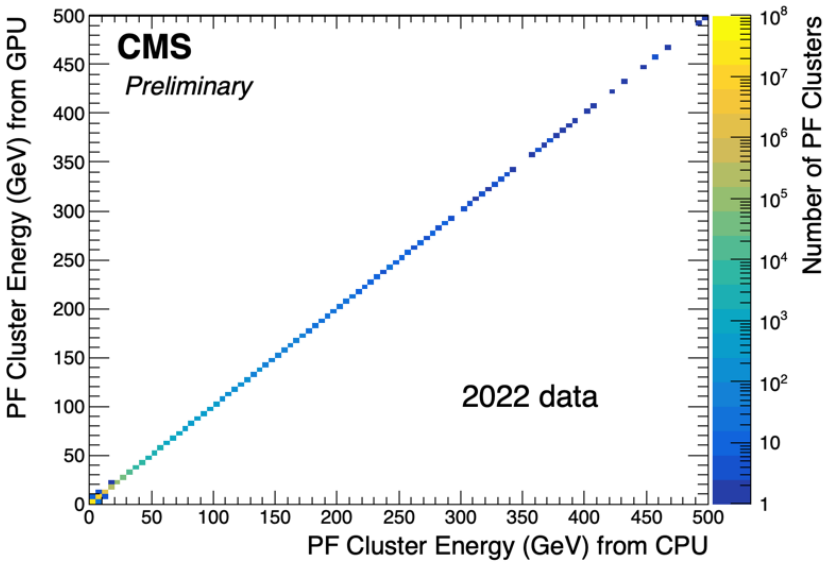

Figure 4: Comparisons of reconstructed PF cluster energies with CPU and GPU clustering algorithms as measured in Run 3 2022 data. There is a good agreement between PF cluster energies reconstructed with CPU and GPU.