EPJ Web of Conferences **245**, 05005 (2020) https://doi.org/10.1051/epjconf/202024505005
CHEP 2019

GPU-based Clustering Algorithm for the CMS High Granularity Calorimeter

Ziheng Chen[1,*], *Antonio* Di Pilato[2,3,], *Felice* Pantaleo[4,], and *Marco* Rovere[4,]
on behalf of the CMS Collaboration

[1]Northwestern University
[2]University of Bari
[3]National Institute of Nuclear Physics (INFN)
[4]European Organization for Nuclear Research (CERN)

Abstract. The future High Luminosity LHC (HL-LHC) is expected to deliver about 5 times higher instantaneous luminosity than the present LHC, resulting in pile-up up to 200 interactions per bunch crossing (PU200). As part of the phase-II upgrade program, the CMS collaboration is developing a new endcap calorimeter system, the High Granularity Calorimeter (HGCAL), featuring highly-segmented hexagonal silicon sensors and scintillators with more than 6 million channels. For each event, the HGCAL clustering algorithm needs to group more than 10^5 hits into clusters. As consequence of both high pile-up and the high granularity, the HGCAL clustering algorithm is confronted with an unprecedented computing load. CLUE (CLUsters of Energy) is a fast fully-parallelizable density-based clustering algorithm, optimized for high pile-up scenarios in high granularity calorimeters. In this paper, we present both CPU and GPU implementations of CLUE in the application of HGCAL clustering in the CMS Software framework (CMSSW). Comparing with the previous HGCAL clustering algorithm, CLUE on CPU (GPU) in CMSSW is 30x (180x) faster in processing PU200 events while outputting almost the same clustering results.

1 Introduction

The luminosity of the future High Luminosity Large Hadron Collider (HL-LHC) is going to achieve up to 7.5×10^{34} $\mathrm{cm}^{-2}\mathrm{s}^{-1}$ [1] in its ultimate scenario, which is 5 times that delivered at present. This leads to the production of high pile-up events containing up to 200 interactions in each bunch crossing (PU200). Current CMS endcap calorimeters [2] are designed with a lifetime radiation limit of 500 fb^{-1} [3], which will be reached at the end of LHC Run-III in 2023. During the third long shutdown period of LHC from 2024 to 2026, the CMS Collaboration is going to conduct the Phase-II upgrade of the CMS detector [3]. One of the major tasks in the CMS Phase-II upgrade is to replace the current endcap calorimeters, including both endcap electromagnetic and hadronic calorimeters, with a new high granularity calorimeter system (HGCAL) which is based on highly-segmented Silicon sensors and plastic scintillators.

*e-mail: zihengchen2015@u.northwestern.edu

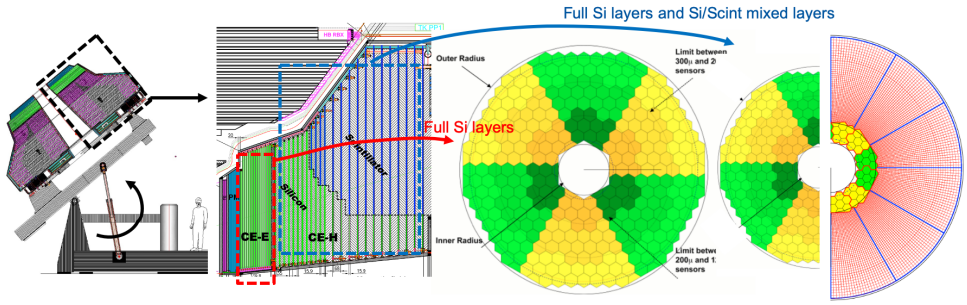

Figure 1. The design of HGCAL [4]. *Left:* Sketch of HGCAL endcap. *Mid-left:* The internal structure on the longitudinal-radial (z-r) plane. Red and blue rectangles indicate regions of CE-E and CE-H respectively, where CE-E has 28 full Si layers and CE-H has 8 full Si layers plus 14 Si-scintillators hybrid layers. *Mid-right, Right:* Layouts of CE-E and CE-H layer. Silicon wafers are shown as yellow and green hexagons and scintillators are shown as red mesh. Darker, medium and lighter shades of hexagons represent silicon wafers with thickness of 120, 200 and 300 μm respectively.

The design of HGCAL [4] is shown in Figure 1. Two HGCAL endcaps will be mounted on both sides of the CMS detector. Each endcap weighs about 215 tons and measures about 2 m in longitudinal direction and 2.3 m in radial direction, covering $1.5 < |\eta| < 3.0$. The full system operates at a temperature of $-35°$C maintained by a CO_2 cooling system. Each endcap consists of 50 layers, each of which combines passive absorber material and active sensor material. The front 28 layers are the electromagnetic part (CE-E), which uses Cu, CuW and Pb as absorber and Si wafers with 120, 200, 300 μm thickness as sensors. The back 22 layers are the hadronic part (CE-H), which uses stainless steel and Cu as absorber and includes 8 full Silicon layers plus 14 hybrid layers of Si sensors and plastic scintillators with SiPM readout. The electromagnetic radiation thickness and hadronic interaction thickness of CE-E are $25X_0$ and 1.3λ respectively, while the hadronic interaction thickness of CE-H is 8.2λ. In total, The full HGCAL system has 620 m^2 of Silicon and about 400 m^2 of plastic scintillators. The size of each Si sensor is 0.5-1.0 cm^2 and the number of Si channels is about 6 million. The size of the scintillators is 4-30 cm^2 and the number of scintillator channels is about 240 thousand.

As a consequence of both high pile-up in the HL-LHC and enormous number of channels in the HGCAL, the number of input hits to HGCAL clustering algorithm is huge, usually in the order of $n \sim O(10^5)$ in PU200 events, where n denotes the number of hits. The clustering algorithm aggregates hits in 2D clusters layer by layer, producing about $k \sim O(10^4)$ clusters, where k denotes number of clusters. The average number of hits in a cluster is about $m = n/k \sim 10$; therefore HGCAL clustering task is characterized by $n > k \gg m$. Since cells are small compared to shower lateral size, an "energy density" is defined to better hint regional energy blobs in the HGCAL clustering. After 2D clustering algorithm, 3D showers in HGCAL are reconstructed by collecting and associating 2D clusters on different layers using TICL algorithms [5].

The current trigger system in CMS consists of two levels: Level 1 Trigger (L1T) and High Level Trigger (HLT). L1T utilizes customized ASICs and FPGAs to reduce the event rate from 40 MHz LHC collision frequency to 100 kHz within a 4 μs time budget for decision; HLT is fully based on C++ software running on CPUs and further reduces event rate from 100 kHz to 1 kHz with a 300 ms time budget for decision. However, in the era of HL-LHC, CMS HLT expects 30 times more computing load: 1.3x from upgraded detectors with more

channels; 3x from increased number of pile-up interactions; 7.5x from improved L1T output rate. Among this 30x surge of the computing demand, improvement in the CPU performance by 2026 is expected to account for only 4x. Therefore, there will be a considerable deficit of computing power if the HLT architecture remains unchanged in 2026. In the HL-LHC era, the HLT time budget for CMS HGCAL clustering is roughly estimated to be less than a few tens of milliseconds. It is particularly a huge challenge of computing for the HGCAL clustering algorithm to process $n \sim 5 \times 10^5$ hits within such a limited time budget. To cope with this computing challenge, CMS is studying the feasibility of heterogeneous computing in HLT and offline reconstruction. With the support of CUDA [6] in the CMS software framework (CMSSW), it is possible to accelerate the HGCAL reconstruction with GPUs. In this paper, we present both CPU and GPU implementations of a fast fully-parallelizable density-based clustering algorithm, CLUsters of Energy or CLUE [7], in CMSSW for HGCAL reconstruction. In addition, we demonstrate that CLUE on CPU (GPU) is about 30x (180x) faster than the previous HGCAL clustering algorithm [8] in CMSSW for PU200 events.

2 HGCAL Clustering Algorithm

The previous clustering algorithm [8] used in the CMS HGCAL reconstruction was based on Clustering by Fast Search and Find Density Peak (CFSFDP) [9] and exploited a KD-Tree spatial index [10]. In the step of calculating local density ρ, KD-Tree provides a significant speedup comparing with not using any spatial index [8]. However, it has three crucial computing weaknesses: first, KD-Tree does not provide the optimal spatial index for HGCAL, because its window-query is of $O(n \log n)$ complexity and it is hard to construct or query on the GPUs; second, the calculation of separation δ does not take advantage of spatial index but still relies on a costly $O(n^2)$ loop; third, the expansion of clusters happens in sequential order of decreasing density, which is not only costly because of sorting but also hard to parallelize.

CLUsters of Energy (CLUE) [7] is a recently-proposed parallelizable high-speed clustering algorithm. It overcomes the above three computing weaknesses and achieves an average $O(n)$ computational complexity in the applications like HGCAL where $n > k \gg m$. CLUE uses a spatial index [11] for fast querying of neighbours. Figure 2 is a demonstration of CLUE procedure provided in [7]. Both the CPU and the GPU version of CLUE, referred as CLUE-CPU and CLUE-GPU in this paper, have been implemented in CMSSW for HGCAL reconstruction. CLUE-CPU is implemented in C++, while CLUE-GPU is implemented using CUDA. Figure 3 shows the workflow of CLUE-GPU within CMSSW: hits are offloaded from CPU to GPU after energy calibration; then CLUE steps are carried out on GPU; in the end, the clustering results are transported back to CPU for post processing and other downstream HGCAL reconstruction related to 3D linkage of CLUE clusters.

To validate the implementation of CLUE in CMSSW, results of CLUE-CPU and CLUE-GPU are compared with the previous clustering algorithm in CMSSW version 10.6, referred as CMSSW_10_6_X in this paper. Based on the simulated $t\bar{t}$ events, CLUE-CPU and CLUE-GPU completely agree with each other, while both of them show some rare disagreements with the previous clustering algorithm implemented in CMSSW_10_6_X. Such disagreements are caused by the different ordering of hits with exactly equal ρ or equal δ when using different data structures, namely grid in CLUE and KD-Tree in CMSSW_10_6_X. An example of clustering result is shown in Figure 4, where from left to right are results from CMSSW_10_6_X, CLUE-CPU and CLUE-GPU. In this example, CLUE-CPU and CLUE-GPU provide almost the same result as the clusters in CMSSW_10_6_X. However, a small notable difference is the blue cluster, which includes 4 hits in CMSSW_10_6_X but 2 in CLUE. This is because the hit at about (x=60, y=106) cm is equally close to the two neighbouring hits in orange cluster and blue cluster, and its two different assignments, caused by

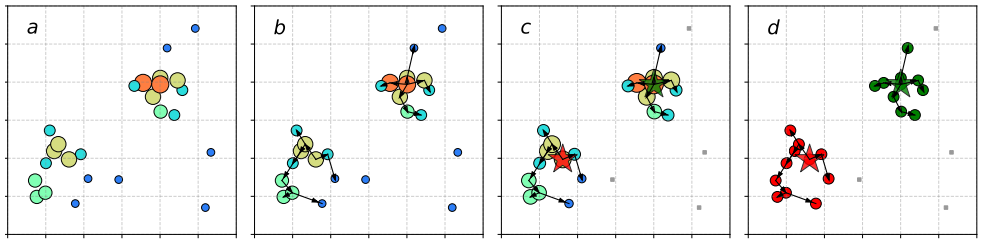

Figure 2. Demonstration of CLUE procedure [7]. The definitions of four internal variables $\{\rho, \delta, nh, followers\}$ are also given in [7]. Before the clustering procedure starts, a fixed-grid spatial index is constructed. In the first step, shown as (a), CLUE calculates the local density ρ for each point. The color and size of points represent their local densities. In the second step, shown as (b), for each point CLUE calculates its nearest-higher nh (defined as the nearest hit with higher density) and its separation δ (defined as the distance to nh). The black arrows represent the relation from the nearest-higher of a point to the point itself. If the nearest-higher of a point is -1, there is no arrow pointing to it. In the third step, shown as (c), CLUE promotes a point as a seed if ρ, δ are both large, or demote it to an outlier if ρ is small and δ is large. Promoted seeds and demoted outliers are shown as stars and grey squares, respectively. In the fourth step, shown as (d), CLUE propagates the cluster indices from seeds through their chains of followers. Noise points, which are outliers and their descendant followers, are guaranteed not to receive cluster ids from any seeds. The color of points represents the cluster ids. A grey square means its cluster id is undefined and the point should be considered as noise.

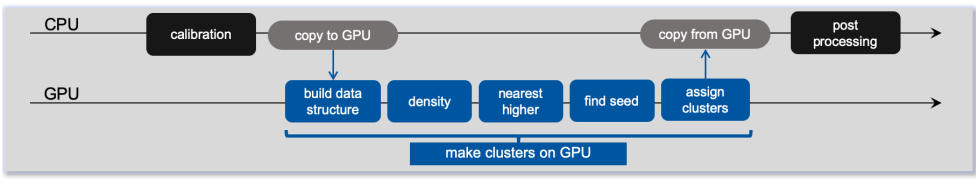

Figure 3. Workflow of CLUE-GPU in CMSSW. Hits are offloaded from CPU to GPU after energy calibration. Then CLUE process are carried out on GPU. In the end, the cluster indices of all hits are transported back to CPU for post processing.

different ordering of these two neighbors in spatial index, are equally correct. The topology of blue cluster in both cases are acceptable. Therefore, it is reasonable to conclude that CLUE in CMSSW gives almost the same clustering result as CMSSW_10_6_X with neglectable differences.

3 Performance of CLUE in CMSSW

The execution time of HGCAL clustering are tested using PU200 events. The testing platform is based on Intel i7-4770K CPU and NVIDIA GTX 1080 GPU. The average execution time is shown in Figure 5, where measured time includes all clustering steps and all necessary data transfer between CPU and GPU.

The previous clustering algorithm in CMSSW_10_6_X using a single thread CPU takes 6110 ms on average. In comparison, CLUE-CPU takes only 203 ms using the same single

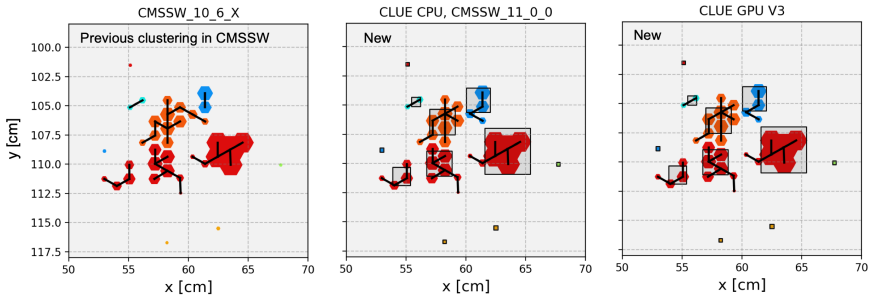

Figure 4. Example of clustering result from previous algorithm in CMSSW_10_6_X (*left*) and CLUE-CPU (*middle*) and CLUE-GPU (*right*). The example shows a small region on the 12^{th} layer of a simulated of $t\bar{t}$ event.

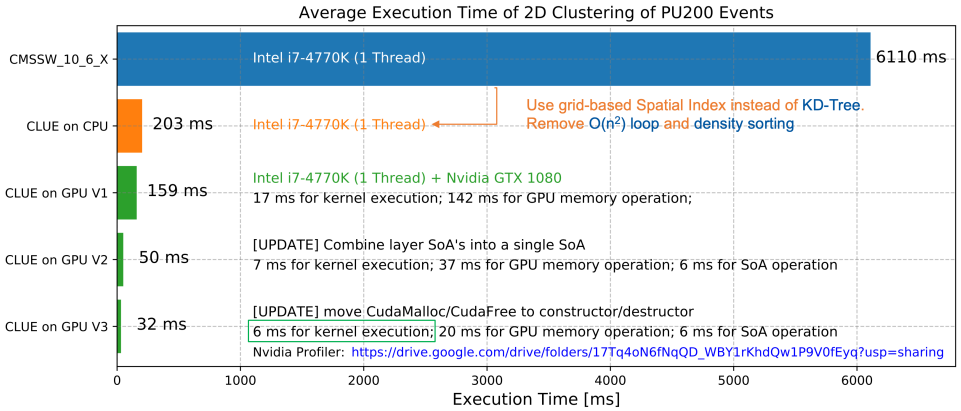

Figure 5. Average execution time of HGCAL clustering for PU200 events. The testing platform is based on Intel i7-4770K CPU and NVIDIA GTX 1080 GPU. Blue, orange and green bars represent execution time of CMSSW_10_6_X, CLUE-CPU and CLUE-GPU respectively. Both CMSSW_10_6_X and CLUE-CPU use a single CPU thread. Three green bars are three evolving versions of CLUE-GPU and the most updated one is version 3 with 32 ms execution time, shown as the bottom-most green bar.

thread CPU, producing almost the same result but 30x faster. The GPU implementation in CMSSW includes three versions. The first version is a plain CUDA implementation of CLUE-CPU and average execution time is 159 ms. The second version combines the data of all hits in the entire HGCAL as a single Structure of Array (SoA) to improve access to global memory and to allow parallelization of hits on different layers. The average execution time of the second version is reduced to 50 ms. The third version uses one-time GPU memory allocation and memory release before and after processing all events respectively. It further reduces execution time to 32 ms, which is decomposed into 6 ms for kernel execution, 20 ms for host-device data transportation and 6 ms for SoA conversion. The 6 ms total kernel execution time is comparable with that in [7]. The speedup factor of CLUE-GPU over CLUE-CPU is about 6x.

In the future, the latency due to data traffic and SoA conversion can be shared with other reconstruction processes if more processes are also offloaded to GPU. Such latency can also

be partially hidden if multiple CUDA streams work on different events simultaneously to keep the GPU occupied.

4 Conclusion

CLUE has been integrated into CMSSW for the 2D clustering of HGCAL reconstruction. Thanks to the support of heterogeneous architecture in CMSSW, CLUE is able to run not only on CPUs but also be offloaded to GPUs via CUDA. Comparing with performance of previous CPU-based clustering algorithm in CMSSW_10_6_X, CLUE produces almost the same clustering result but 30x (180x) faster when running on CPU (GPU). The average execution time of CLUE-GPU for PU200 events is reduced to 32 ms, which is promising to satisfy the time budget for HGCAL clustering in HLT. Within 32 ms of CLUE-GPU, kernels for algorithmic computation take only 6 ms in total, while the latency due to data transfer (20 ms) and SoA conversion (6 ms) are the bottlenecks. However, in the future, the latency due to data traffic and SoA conversion can be shared with other reconstruction steps and can also be partially hidden if multiple CUDA streams work on different events simultaneously.

References

[1] Large Hadron Collider, *Schedules and luminosity forecasts* (2015), https://lhc-commissioning.web.cern.ch/lhc-commissioning/schedule/HL-LHC-plots.htm

[2] S. Chatrchyan et al. (CMS), *The CMS Experiment at the CERN LHC* (2008), Vol. 3, p. S08004

[3] D. Contardo, M. Klute, J. Mans, L. Silvestris, J. Butler (CMS), *Technical Proposal for the Phase-II Upgrade of the CMS Detector* (2015)

[4] CMS Collaboration (CMS), *The Phase-2 Upgrade of the CMS Endcap Calorimeter* (Geneva, 2017), CERN-LHCC-2017-023. CMS-TDR-019, technical Design Report of the endcap calorimeter for the Phase-2 upgrade of the CMS experiment, in view of the HL-LHC run., https://cds.cern.ch/record/2293646

[5] CMS HGCAL DPG, *Hgcal reconstruction website ticl* (2020), http://hgcal.web.cern.ch/hgcal/Reconstruction/TICL

[6] Nvidia CUDA, *Nvidia cuda c programming guide* (2011), Vol. 120, p. 8

[7] M. Rovere, Z. Chen, A. Di Pilato, F. Pantaleo, C. Seez, *CLUE: A Fast Parallel Clustering Algorithm for High Granularity Calorimeters in High Energy Physics* (2020), arXiv:2001.09761

[8] Z. Chen, C. Lange, E. Meschi, E. Scott, C. Seez (CMS), *Offline Reconstruction Algorithms for the CMS High Granularity Calorimeter for HL-LHC*, in *Proceedings, 2017 IEEE Nuclear Science Symposium and Medical Imaging Conference and 24th international Symposium on Room-Temperature Semiconductor X-Ray & Gamma-Ray Detectors (NSS/MIC 2017): Atlanta, Georgia, USA, October 21-28, 2017* (2018), p. 8532605

[9] A. Rodriguez, A. Laio, *Clustering by fast search and find of density peaks* (American Association for the Advancement of Science, 2014), Vol. 344, pp. 1492–1496, ISSN 0036-8075, https://science.sciencemag.org/content/344/6191/1492

[10] J.L. Bentley, *Multidimensional Binary Search Trees Used for Associative Searching* (ACM, New York, NY, USA, 1975), Vol. 18, pp. 509–517, ISSN 0001-0782, http://doi.acm.org/10.1145/361002.361007

[11] J.L. Bentley, J.H. Friedman, *Data structures for range searching* (ACM, 1979), Vol. 11, pp. 397–409