EPJ Web of Conferences **251**, 04017 (2021) https://doi.org/10.1051/epjconf/202125104017
CHEP 2021

Heterogeneous techniques for rescaling energy deposits in the CMS Phase-2 endcap calorimeter

Bruno Alves[1,*], *Andrea* Bocci[1], *Matti* Kortelainen[2], *Felice* Pantaleo[1], and *Marco* Rovere[1]

[1]CERN, 1211 Geneva 23, Switzerland
[2]Fermi National Accelerator Laboratory, Batavia, Illinois, U.S.A.

Abstract. We present the porting to heterogeneous architectures of the algorithm used for applying linear transformations of raw energy deposits in the CMS High Granularity Calorimeter (HGCAL). This is the first heterogeneous algorithm to be fully integrated with HGCAL's reconstruction chain. After introducing the latter and giving a brief description of the structural components of HGCAL relevant for this work, the role of the linear transformations in the calibration is reviewed. The many ways in which parallelization is achieved are described, and the successful validation of the heterogeneous algorithm is covered. Detailed performance measurements are presented, including throughput and execution time for both CPU and GPU algorithms, therefore establishing the corresponding speedup. We finally discuss the interplay between this work and the porting of other algorithms in the existing reconstruction chain, as well as integrating algorithms previously ported but not yet integrated.

1 Introduction

The operation of the High Luminosity LHC (HL-LHC) is expected to commence in 2027, achieving instantaneous luminosities of $\sim 5 \times 10^{34}\,\mathrm{cm}^2\,\mathrm{s}^{-1}$, more than two times LHC's current value. Over 10 years it will reach an integrated luminosity of $\sim 3\,\mathrm{ab}^{-1}$, with potentially up to 200 proton collisions (pileup) per bunch crossing. The goals of the HL-LHC include measuring the Higgs boson (self) couplings, vector boson fusion and vector boson scattering processes (also involving the Higgs boson), and B physics processes, among others [1].

In accordance with this programme, the upgrade of the CMS detector [2] foresees a High Granularity Calorimeter (HGCAL) [3] to replace the current endcap calorimeters. One of the challenges posed to CMS by the new calorimeter is writing reconstruction code allowing its full exploitation.

Present projections show a gap between projected CPU needs and availability at the start of the HL-LHC (Run4), as displayed in Fig. 1. The biggest contributor to CPU usage is event reconstruction (see Fig. 2), of which currently ~6% is used by HGCAL. CMS plans to port some parts of its reconstruction to Graphics Processing Units (GPUs), which represent one of the most promising accelerator technologies on the market. Its adoption would allow access to accelerators, which become more and more present on High-Performance Computing and traditional grid sites. It would also be in line with the direction taken by CMS to adopt a heterogeneous HLT farm already in Run 3. Finally, it potentially reduces the cost of the

*e-mail: bruno.alves@cern.ch

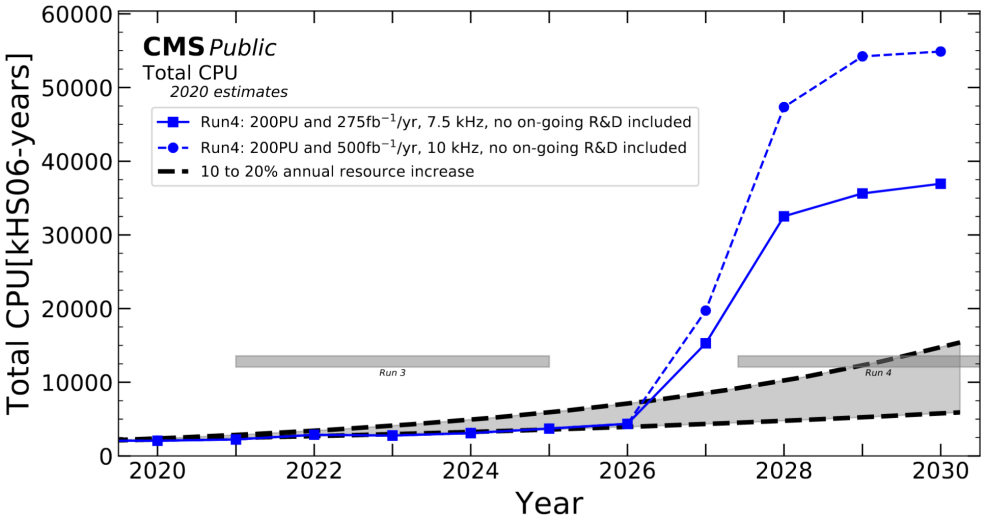

Figure 1. Estimated evolution of CPU computing needs, measured in thousands of HS06 computing capacity units. Without taking accelerator technologies into account, the available resources at Run4 and beyond will fall behind the ones required. Figure taken from [4].

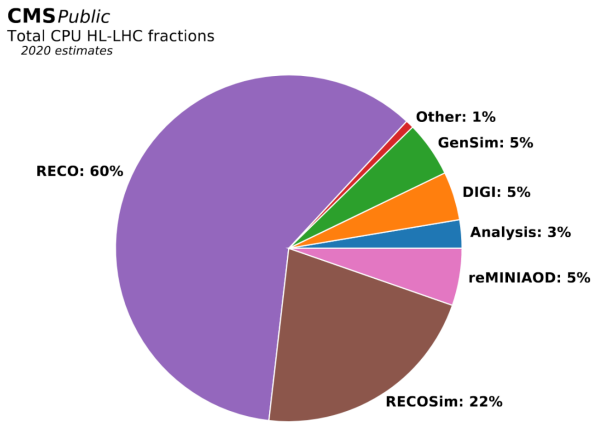

Figure 2. Breakdown of the projected resource needs for CPU usage expected during 2029 (Run 4). "Gen", "Sim", "DIGI" and "RECO" refer to the tasks employed by CMS to process real and simulated data (in order: generation, simulation, digitization and reconstruction). "reMINIAOD" refers instead to the production of the MINIAOD CMS data format for physics analysis with different CMSSW versions. AOD (Analysis Object Data) and more compact data subsets such as MINIAOD derive from the RECO format. Figure taken from [4].

computing capacity necessary to satisfy the CMS physics programme, since computation on GPUs might be cheaper than on CPUs.

In this paper we describe the first heterogeneous implementation, fully compatible with the CMS software (CMSSW) [5, 6], of one of HGCAL's reconstruction stages, namely the rescaling of energy deposits (also loosely called "calibration"). The implementation relies on

Nvidia's CUDA [7]. This effort follows the GPU porting of HGCAL's clustering algorithm (CLustering of Energy, CLUE) [8], which for the moment is implemented as a standalone version. The paper is structured as follows. We start by briefly describing HGCAL in Section 2. We cover the basic functionalities of HGCAL's reconstruction chain, and proceed to describe the methodology of the application of linear transformations to energy deposits, respectively in Sections 3 and 4. Finally, we assess the calibrations' validity and performance in Section 5, and draw conclusions in Section 6.

2 The High Granularity Calorimeter

The HGCAL will be a sampling calorimeter. The proposed design includes, as active material, silicon (Si) sensors in the front 28 layers of its electromagnetic section (CE-E). The hadronic section (CE-H) comprises 8 Si-only layers followed by 14 Si-scintillator hybrid layers, where the scintillation light is read out by Si photo-multipliers (see Fig. 3). The Si sensors are further subdivided into three types with varying thicknesses (120, 200 and 300 µm), capacitances and sizes, to withstand different fluence conditions. The absorbers will be made of CuW, Pb and Cu in the CE-E and stainless steel and Cu in the CE-H, and its thicknesses will vary across layers. The electromagnetic radiation and hadronic interaction lengths of CE-E are $25\,X_0$ and $1.3\,\lambda$ respectively, while the hadronic interaction length of CE-H is $8.2\,\lambda$. In total, the full HGCAL system has $\sim$620 m^2 of Si and $\sim$400 m^2 of plastic scintillators. The size of each Si sensor is 0.5 cm^2 to 1.0 cm^2 (120 µm Si sensors are smaller). Scintillators will range in size from 4 to 30 cm^2, and the number of Si (scintillator) channels is $\sim$6 million ($\sim$240 thousand). Each endcap weighs $\sim$215 t and measures $\sim$2 m ($\sim$2.3 m) in longitudinal (radial) direction. The full system operates at a temperature of $-35\,^{\circ}$C maintained by a CO_2 cooling system [3].

Due to HGCAL's hybrid detection technology, three subdetectors are considered independently for both the CPU and GPU implementation of the reconstruction algorithms:

- CE-E: comprises the first 28 layers made exclusively of Si;
- CE-H_{Si}: covers the Si part of the CE-H section;
- CE-H_{Sci}: covers the scintillator part of the CE-H section.

This partition is relevant to the GPU calibration algorithm described in Section 4.

3 Reconstruction Chain

The current reconstruction model envisioned for HGCAL, part of CMSSW and succinctly depicted in Fig. 4, is intended to be fast and flexible. It comprises a series of modules which transform raw data into physics objects. After the first stages described in [4], one obtains *UncalibRecHits*. They represent energy deposits whose amplitude is expressed in terms of the average number of minimum ionizing particles (MIPs), after being converted from analog-to-digital converter (ADC) counts by the previous Digi step, and taking the sensor thickness into account. This paper covers the following step, *i.e.*, rescaling the hits to produce a CMSSW collection of *RecHits* (see Section Section 4). Continuing along the chain, the software then clusters the *RecHits* into two-dimensional layer clusters, using CLUE [8]. Finally, taking the clusters as its input, The Iterative CLustering (TICL) framework [9] produces 3D objects and showers using a mixture of pattern recognition, energy regression and particle identification techniques. In parallel, a heterogeneous way of navigating through geometrical and topological information within the detector (such as information regarding Si sensors or plastic tiles) is being investigated, in order to accelerate and facilitate its access by different algorithms

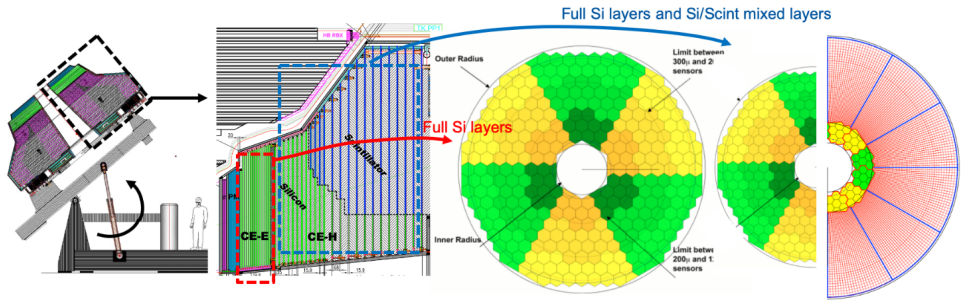

Figure 3. HGCAL's design [3]. *Left*: Sketch of one of HGCAL's endcaps. *Mid-left*: Cross-section of its internal structure. The proton beam line runs horizontally. Red and blue rectangles respectively indicate regions of CE-E and CE-H where CE-E has 28 full Si layers and CE-H has 8 full Si layers plus 14 Si-scintillators hybrid layers. *Mid-right, Right*: Layouts of CE-E and CE-H layers. Si wafers are shown as yellow and green hexagons and scintillators are shown as red mesh. Darker, medium and lighter shades of hexagons represent Si wafers with thickness of 120, 200 and 300 µm, respectively.

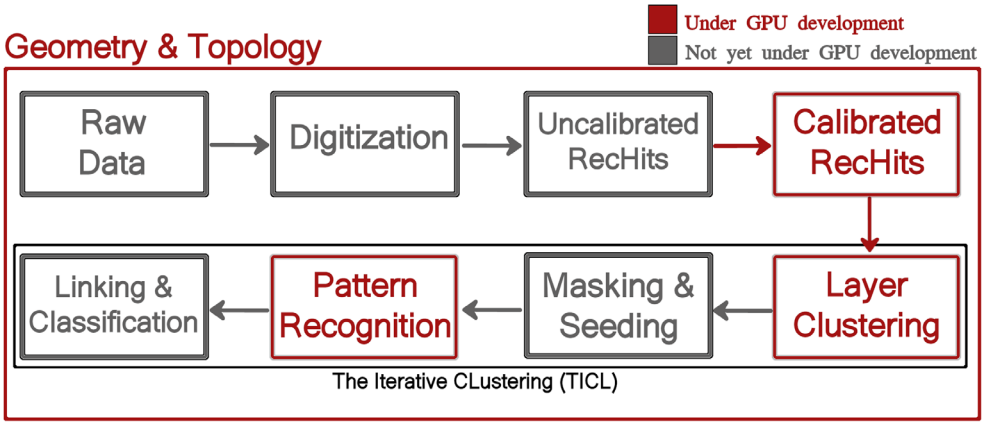

Figure 4. HGCAL's reconstruction chain, from raw data to physics objects. The geometry and topology of the detector are intertwined with all modules. The whole chain was already developed for CPU-only algorithms. We highlight in red the stages currently under development for heterogeneous computing, specifically using CUDA. The first three stages are yet to be completely settled upon. This works covers the top right module of the diagram.

in the chain. The constant need to retrieve the x and y coordinates (in HGCAL's transversal plane) in CLUE is an example of these navigation challenges.

4 Methodology

The energy calibration step consists in converting *UncalibRecHits*, measured in multiples of MIPs, to energy measurements in multiples of electron volts. In order to measure the energy of a particle shower in a sampling calorimeter with varying absorber thickness, the energy deposited in the sensors (*RecHits*) must be weighted on a per-layer level. The energy weights, with units MeV/MIP, are calculated with help from Geant4 [10]. Each *UncalibRecHit* is

thus rescaled by one of these weights, depending on the layer in which it is located, and it is further corrected with so-called "regional electromagnetic factors", calculated by comparing the incident energy of a simulated photon with the shower energy measured using rescaled hits. Currently seven of these factors are defined: six for the three Si thicknesses in both CE-E and CE-H, and one more for the scintillator in CE-H. Hits in the CE-E are additionally corrected by thickness-dependent charge collection efficiencies [11].

The code implementing this type of hit-dependent corrections can be straightforwardly converted to a heterogeneous version by assigning one CUDA thread to each hit. There is no need for hits to share information with others, allowing full hit parallelization. Attention is given to systematically allocating memory in multiples of CUDA's *warpSize* (currently 32 threads), and to access it such that coalesced reads are possible. This is known to increase throughput by potentially an order of magnitude [12].

Further parallelization is made possible due to HGCAL's topology, namely the three subdetectors presented in Section 2. They are considered independently of each other, both in the CPU and GPU implementations, and differ solely in the considered layers and regional correction factors. When using CUDA, this split allows the exploitation of independent CUDA streams, one per subdetector, as shown schematically in Fig. 5. The execution of actions (kernel execution, copies, and transfers, for instance) is guaranteed to be serialized within a CUDA stream. The same streams can also be used for other modules which do not require sharing of information at subdetector-level, such as the ones which produce *UncalibRecHits*.

The calibration module is developed as an integral part of CMSSW, and takes advantage of some of its CUDA-related functionalities [13, 14]. From a computational perspective, as shown in Fig. 6, the calibration starts by converting *UncalibRecHits* into a data structure more suitable for GPUs, namely a Structure of Arrays (SoA). The latter improves access to global memory in the GPU and allows CPU vectorization to be exploited [15]. The SoAs are then transferred to the GPU, together with some configuration constants, where the calibration is performed. The data will then be directly available on the GPU to be used by the next heterogeneous module, avoiding needless data transfers, which are a common GPU bottleneck [12]. To increase the framework's flexibility and to make its validation possible, the data can also be transferred back to the CPU. In total, each subdetector will have its calibration performed by a maximum of 4 modules (12 in total), event by event:

1. Conversion of the CMSSW input data collection (*UncalibRecHits*) into a SoA, followed by its transfer to the GPU;

2. Execution of CUDA kernels in the GPU;

3. Transfer the calibrated SoA back to the CPU (optional);

4. Conversion of the SoA to the *RecHits* CMSSW collection (optional).

The final goal of the heterogeneous efforts will be to perform the whole HGCAL reconstruction directly within the GPU (Step 2 only), where each module receives GPU data from the previous heterogeneous module, and feeds to the subsequent heterogeneous module the new data it has processed. Ideally, memory transfers to/from the GPU will only be performed at the start and end of the reconstruction chain. As an example, the linking between the calibration and clustering in the GPU is currently under development.

For completeness, we further mention that data-oriented structures (such as SoAs) are being considered to replace the current CMSSW object-oriented legacy format. This would allow the acceleration of CPU code and the removal of the fourth step during GPU-CPU transfers in heterogeneous modules.

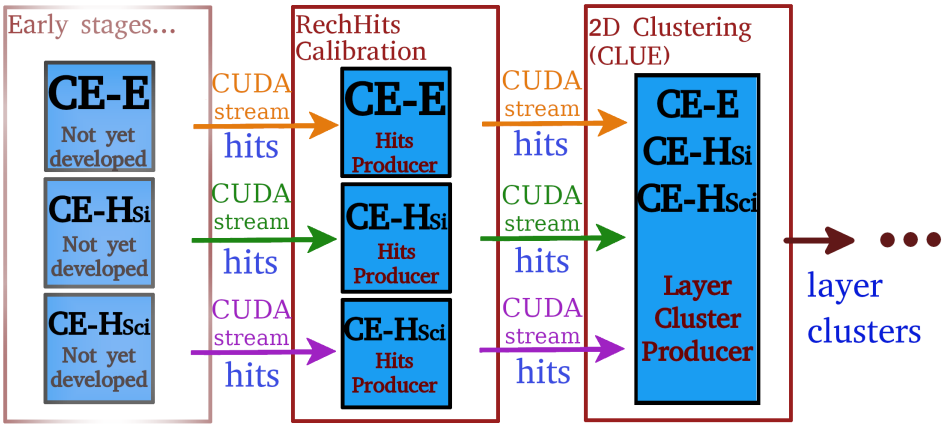

Figure 5. The rescaling of *UncalibRecHits* is done independently for each of the three subdetectors in HGCAL. Each of them gets assigned a different CUDA stream. The streams will be propagated and merged in the clustering stage to allow the creation of 3D objects out of 2D clusters. The box having "Early stages" as title refers to all modules which precede the *UncalibRecHits* rescaling.

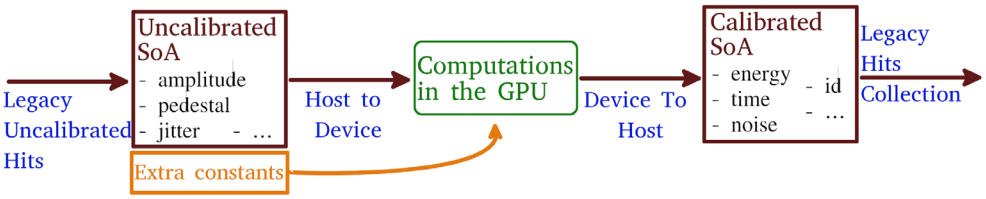

Figure 6. Diagram showcasing the operations involved in rescaling *UncalibRecHits* using GPUs, implemented using four independent modules per subdetector. Both the "host-to-device" and "device-to-host" transfers (where device refers to the GPU and host to the CPU) are optional, and can be replaced by a fully-heterogeneous reconstruction chain.

5 Validation and Performance

Version `12_0_0_pre1` of the CMSSW software [5] is used for both validation and performance measurements.

The validation procedure is straightforward: the CPU-only and GPU calibration modules are run over the same input dataset, starting from three previously produced *UncalibRecHits* collections, one per subdetector. The obtained *RecHits* are then compared in terms of their underlying quantities, such as the rescaled energy. The difference is always found to be virtually zero for the three subdetectors ($\sim 10^{-10}$ for CE-E and smaller for CE-H_{Si} and CE-H_{Sci} at 200 pileup). This is true for 0, 50, 100, 140 and 200 pileup. The same pileup values are also considered for performance measurements.

Performance is assessed using a T4 Nvidia GPU and an Intel(R) Xeon(R) Silver 4114 CPU (40 logical cores), together with a benchmarking tool [16] by Patatrack [17]. It measures throughput, defined as the number of events processed per second. We perform all throughput measurements using 4 jobs with 10 CPU threads each and 1100 events, skipping the first 100 to mitigate GPU initialization costs. The events are stored in the CPU cache to avoid the observed negative I/O impact on measured throughput. Fig. 7 shows the measurements for

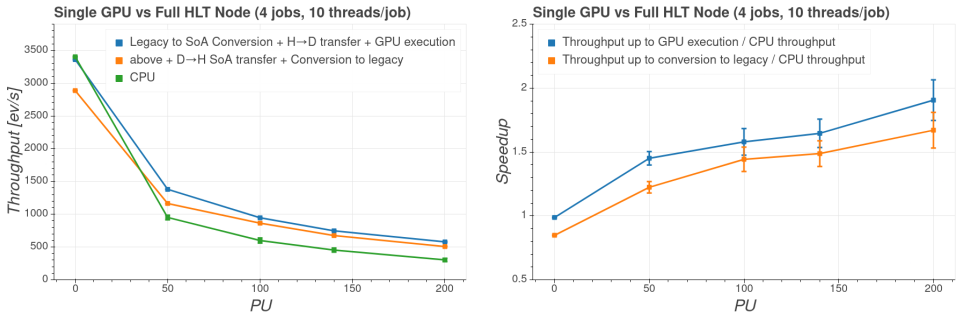

Figure 7. *Left*: Throughput measurements of the rescaling of energy deposits as a function of pileup, considering 4 jobs with 10 CPU threads each. The blue, orange and green curves represent the throughput of, respectively, everything up to GPU execution (steps 1 and 2 from Section 4), the full GPU chain and the CPU rescaling algorithm. 'H' and 'D' stand for, respectively, host and device. Vertical error bars as given by [16] are present but are too small to be seen. *Right*: Speedup obtained after dividing the blue and orange distributions on the left by the CPU throughput (green). The comparison is being done between a full node and a single GPU.

the CPU and two different GPU scenarios (left) and the obtained speedups (right), defined as the throughput of a GPU scenario divided by the CPU throughput. We stress that the comparison is being made between a full CMS node and a single GPU. We further observed an average GPU memory usage of ~50 MiB per CUDA stream.

It is currently not possible to measure the throughput of the *UncalibRecHits* rescaling alone, since at least the SoA host-to-device transfer must be included in the measurement. However, most conversions and transfers will be made optional once the full reconstruction chain has been deployed into GPUs. In order to estimate a conversion- and transfer-free speedup we use Nvidia's profiler `nvprof` [18] to extract the average kernel execution time per event. The CPU rescaling algorithm is instead timed with the C++ `std::chrono` library. Both CPU and GPU time measurements are performed with 1 job and 1 CPU thread. To maximize the occupancy of the GPU and thus expose its benefits over its serial version we consider the CE-E section, where ~500 K hits are expected per event, while the CE-H_{Si} and CE-H_{Sci} have, respectively, ~50 K and ~1 K. The speedup is here defined to be the CPU average time required to process an event, divided by the average time required by the GPU for the same task. Fig. 8 shows a speedup of around three orders of magnitude. By measuring average time for single events, any form of module concurrency is necessarily ignored in the final measurements. We therefore expect the heterogeneous rescaling algorithm to perform better in a real-world scenario (in terms of throughput), where it will have access to multiple CPU jobs and threads, and multiple CUDA streams.

If the number of *RecHits* is low (as in the case of the CE-H_{Sci} section), GPU-related efforts can potentially be counter-productive, since memory allocations and transfers might actually be slower than the CPU algorithm. We test this by considering the CE-H_{Sci}-related GPU modules only, and by adapting the established CPU calibration to run solely on CE-H_{Sci} hits. We do not observe any significant impact in the throughput. If present, it nevertheless implies an optimization of negligible time intervals, where there is very few data being processed.

Additionally, we again note that the final chain will include multiple modules accessing data directly in the GPU; memory transfers and allocations will therefore not be requested on a per-module basis, significantly reducing their impact on the overall throughput. Moreover, as mentioned in Section 4, we are currently investigating SoA-like formats for data residing

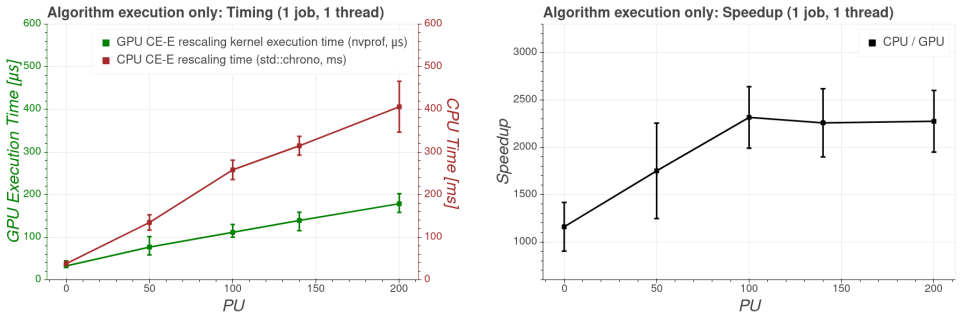

Figure 8. *Left*: Average time measurements per event of the rescaling algorithm only, as a function of pileup, considering 1000 events, 1 job and 1 CPU thread. Note the two vertical axis: CPU (GPU) timing results in brown (green) are displayed in milliseconds (microseconds). CPU (GPU) average times are measured with `nvprof` (`std::chrono`). The GPU vertical errors are obtained from `nvprof` maximum and minimum kernel execution times. The CPU errors are instead calculated using the standard deviation of all events. *Right*: Speedup obtained after dividing the CPU results by the GPU results on the left. A difference of three orders of magnitude is present, increasing for the first three pileup values.

on the CPU to decrease or eliminate the time required for conversions between SoA and legacy formats.

6 Conclusions

This work presents the first HGCAL heterogeneous algorithm to be fully integrated within CMSSW, and the second to be developed (CLUE is for the moment available as a standalone heterogeneous algorithm). It describes the algorithm's implementation, validation and performance measurements. Its motivation is clear: CMS will have to cope with the very significant radiation and pileup increases expected for the HL-LHC. These challenges might be tackled by the widespread adoption of accelerator technologies, GPUs in particular.

Performance measurements indicate that the GPU provides a throughput up to twice as large as the CPU one, including data conversions and transfers which will not be present in the final version of the reconstruction chain. Timing measurements of the rescaling algorithm alone suggest a speedup of three orders of magnitude. Both estimates might be conservative, given that a better result is foreseen as soon as a full GPU reconstruction chain has been developed and is ready for production, taking advantage of multiple CPU jobs, streams, threads and subdetector parallelism. The rescaling, given its speed and relative simplicity, will surely not represent the bottleneck of the chain.

In addition to the speedup, the implementation of the rescaling algorithm paves the way for future accelerator developments in HGCAL. Being essential for allowing the entire reconstruction to be ported to GPUs, it provides a strong motivation for the development of the CMSSW heterogeneous rescaling-clustering linking and grants the opportunity to assess CLUE's heterogeneous performance. It further facilitates the deployment of the calibration to multiple accelerators using abstraction libraries such as `alpaka` [19]. Finally, GPU computing has the added benefit of being potentially cheaper than its CPU counterpart.

Acknowledgments: Bruno Alves is supported by *Fundação para a Ciência e Tecnologia* under reference SFRH/BEST/150186/2019. Matti Kortelainen is supported by the Fermi National Accelerator

Laboratory, managed and operated by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy.

References

[1] D. Contardo, M. Klute, J. Mans, L. Silvestris, J. Butler, Tech. Rep. CERN-LHCC-2015-010. LHCC-P-008. CMS-TDR-15-02, Geneva (2015), https://cds.cern.ch/record/2020886

[2] S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A.M. Sirunyan, R. Adolphi, G. Anagnostou, R. Brauer, W. Braunschweig, H. Esser, L. Feld et al. (CMS Collaboration), JINST **3**, S08004. 361 p (2008), also published by CERN Geneva in 2010

[3] Tech. Rep. CERN-LHCC-2017-023. CMS-TDR-019, CERN, Geneva (2017), https://cds.cern.ch/record/2293646

[4] C.O. Software, Computing, Tech. Rep. CMS-NOTE-2021-001. CERN-CMS-NOTE-2021-001, CERN, Geneva (2021), https://cds.cern.ch/record/2751565

[5] The CMS Collaboration, *CMS Software*, http://cms-sw.github.io/ (2021), accessed: 2021-02-18

[6] C.D. Jones, M. Paterno, J. Kowalkowski, L. Sexton-Kennedy, W. Tanenbaum, *The New CMS Event Data Model and Framework*, in *Proceedings of International Conference on Computing in High Energy and Nuclear Physics (CHEP06)* (2006)

[7] NVIDIA, P. Vingelmann, F.H. Fitzek, *Cuda, release: 11.2.152* (2021), https://developer.nvidia.com/cuda-toolkit

[8] Z. Chen, A. Di Pilato, F. Pantaleo, M. Rovere, EPJ Web of Conferences **245**, 05005 (2020)

[9] C.H. DPG, *HGCAL Reconstruction Website, TICL*, https://hgcal.web.cern.ch/Reconstruction/TICL/, accessed: 2021-02-17

[10] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand et al., Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506**, 250 (2003)

[11] M. Rovere, *HGCAL Reconstruction Website, Hit Calibration: Procedure*, https://hgcal.web.cern.ch/HitCalibration/hitCalibration/, accessed: 2020-12-25

[12] NVIDIA, *CUDA C++ Programming Guide* (2021), accessed: 2021-02-17, https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[13] A. Bocci, D. Dagenhart, V. Innocente, C. Jones, M. Kortelainen, F. Pantaleo, M. Rovere, EPJ Web Conf. **245**, 05009 (2020)

[14] M. Kortelainen, *CUDA algorithms in CMSSW, CUDA ESProduct*, https://github.com/cms-sw/cmssw/blob/CMSSW_11_2_1/HeterogeneousCore/CUDACore/README.md, accessed: 2020-12-25

[15] V. Innocente, *Efficient Memory Management*, https://agenda.infn.it/event/17237/contributions/35945/ (2019), accessed: 2021-02-17

[16] A. Bocci, *Patatrack scripts: Benchmark*, https://github.com/cms-patatrack/patatrack-scripts/blob/master/benchmark, accessed: 2021-05-12

[17] A. Bocci, *Patatrack*, https://patatrack.web.cern.ch/patatrack/, accessed: 2021-05-12

[18] NVIDIA, *nvprof* (2012–2020), accessed: 2021-05-14, https://docs.nvidia.com/cuda/profiler-users-guide/index.html#nvprof-overview

[19] A. Matthes, R. Widera, E. Zenker, B. Worpitz, A. Huebl, M. Bussmann, *Tuning and Optimization for a Variety of Many-Core Architectures Without Changing a Single Line of Implementation Code Using the Alpaka Library*, in *High Performance Computing*, edited by J.M. Kunkel, R. Yokota, M. Taufer, J. Shalf (Springer International Publishing, Cham, 2017), pp. 496–514, ISBN 978-3-319-67630-2