Developing GPU-compliant algorithms for CMS ECAL local reconstruction during LHC Run 3 and Phase 2

Thomas Reis for the CMS Collaboration
STFC Rutherford Appleton Laboratory - Harwell Campus, Didcot OX11 0QX, UK
E-mail: thomas.reis@stfc.ac.uk

Abstract. The higher LHC luminosities foreseen in Run 3 (2022+) and during the Phase 2 of the LHC (2029+), and the consequently larger number of simultaneous proton-proton collisions per event, pose significant challenges for the CMS event reconstruction. In particular maintaining the strict time budget for reconstruction algorithms at the CMS software high level trigger will not be possible considering only the expected increase of processing power from conventional CPUs. Therefore, CMS is investigating the use of heterogeneous architectures, including GPU accelerators, to satisfy the needs of the Phase 2 reconstruction. The local reconstruction algorithm of the CMS electromagnetic calorimeter (ECAL) is among the first to be implemented on heterogeneous platforms for LHC Run 3. This article discusses the necessary changes to the CMS software framework that support the execution of code on accelerators. The current development status of the ECAL local reconstruction algorithm is described and first performance results of GPU enabled code are presented.

1. The case for heterogeneous computing in CMS

The increased LHC luminosity expected in Run 3 (2022+) and the associated larger number of simultaneous proton-proton collisions (pileup) per event pose significant challenges for the CMS event reconstruction. This is particularly important for event filtering at the CMS high level trigger (HLT), where complex reconstruction algorithms must be executed within a strict time budget of a few hundred milliseconds per event. This problem will become even more acute during the Phase 2 of the LHC (2029+), when significantly higher luminosity, pileup and input rates will be delivered to the HLT than foreseen during Run 3. The more than an order of magnitude higher processing power required to perform the online event reconstruction at the HLT during Phase 2 will exceed the expected increase in processing power for conventional CPUs [1], therefore requiring an alternative approach. CMS is investigating the widespread use of heterogeneous architectures, including GPU accelerators, to lower the per-event processing time and, therefore, increase the data throughput per compute node to satisfy the needs of the Phase 2 reconstruction. A prototype system, with GPU accelerator cards available at every HLT compute note, will be operated at the HLT during LHC Run 3 to validate the approach and to gain experience in optimising the reconstruction algorithms for different processor architectures. The local reconstruction algorithm of the CMS electromagnetic calorimeter (ECAL) [2] is among the first to be implemented on heterogeneous platforms since it is well suited for parallel execution.

2. The CMS ECAL local reconstruction

The lead tungstate (PbWO4) electromagnetic calorimeter of the CMS experiment is located inside the 3.8 T solenoid magnet and consists of 61200 crystals in the barrel section and 7324 crystals in each of the two endcaps, for a total of 75848 channels. The analogue pulse, arising from a crystal developing an electromagnetic shower, is sampled every 25 ns, and ten consecutive samples are read out for energy reconstruction when a trigger signal arrives. The ECAL local reconstruction calculates the energy for each individual crystal by reconstructing the signal pulse. The local reconstruction algorithm can be split into three steps:

(i) Unpacking of the raw data from the detector into digitized hits, or 'digis'. Digis are a collection of consecutive samples read out from a single ECAL channel.

(ii) Amplitude reconstruction. Uncalibrated reconstructed hits (RecHits) are calculated for each channel via the multifit algorithm [3]. Overlapping signals from previous or later LHC bunch crossings (termed out-of-time pileup) are subtracted from the central (in-time) amplitude via a template fit. Figure 1 shows how in-time and out-of-time pulses combine in a typical event.

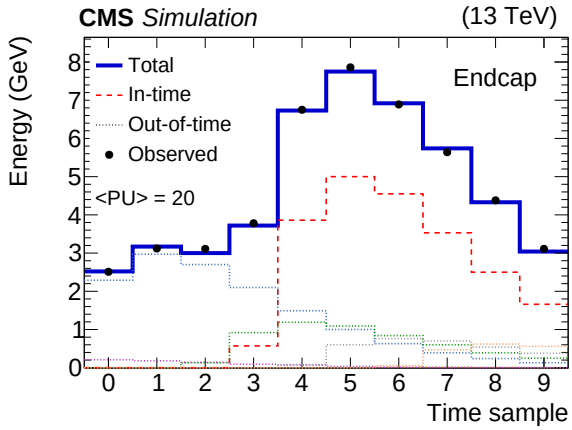

Figure 1. Simulated signal pulse showing the individual components and the fitted sum of in-time and out-of-time pulses.

(iii) Calculation of energies from the uncalibrated RecHits. Various scale factors and calibrations are applied [4] and RecHits in GeV units are obtained that can then be used to form energy clusters and, together with the information from other subdetectors, ultimately reconstruct the final state particle properties.

The multifit algorithm is the central and most computationally intensive part of the ECAL local reconstruction. It involves the iterative minimisation of a χ^2 function and outputs amplitudes for one in-time pulse and up to nine out-of-time pulses.

3. Adapting the CMS software framework for GPU algorithms

The CMS event reconstruction, both offline as well as at the HLT, uses the CMS software framework (CMSSW) [5]. This architecture provides different types of plugin modules that can contain algorithms for reconstruction, analysis, or event filtering. For event reconstruction, a module of the *EDProducer* type typically takes some data product from the event as input, as well as some additional parameters, e.g. calibration constants, and produces a new data product as its output, which is then put back into the event. To facilitate the offloading of work from the CPU to GPUs or other accelerators a number of changes were made to the CMSSW framework:

- **External workers**. To accommodate the offloading of work from the CPU to GPUs or other accelerators, a modification of the *EDProducer* for external workers is performed. This splits the process into two steps. The *acquire()* function loads the input data, starts an asynchronous execution on the GPU, and returns. The *produce()* function is called when the CMSSW framework is notified about the completion of the GPU task and puts the result back into the event. While the external work is being performed, the CMSSW task scheduler can execute other tasks on the CPU.
- **SwitchProducers**. SwitchProducers are used to detect the presence of a GPU during runtime. They will execute the GPU enabled producers if possible, falling back to the CPU producers if the job runs on a machine without GPUs [6].
- **GPU-friendly data formats**. New data formats for digis and (uncalibrated) RecHits are implemented as Structs-of-Arrays (SoA) to exploit the high memory bandwidth of GPUs to read variables from several objects in one operation. Conversion modules produce traditional Array-of-Structs (AoS) data formats for downstream CPU modules.
- **Event Setup (ES) data formats for GPU algorithms**. ES data contain conditions present during data taking and are loaded from a database during the execution of the reconstruction module. GPU optimized SoA versions of the various ECAL conditions formats were implemented. The conditions needed for the algorithm running on the GPU are packaged in the *acquire()* function, and are transferred to the device together with the event data.

4. Implementation of the ECAL local reconstruction on GPUs

All three steps of the ECAL local reconstruction described in Section 2 are implemented in a consecutive way as GPU algorithms using the CUDA C++ programming language, developed for NVIDIA GPUs [7]. This pipelining of GPU modules avoids time consuming data transfers between host and device, and data reformatting between the SoA data format and the legacy AoS formats. Figure 2 shows the modules and data products of the ECAL local reconstruction on the CPU and on the GPU. During normal running, only one of the two pipelines is executed

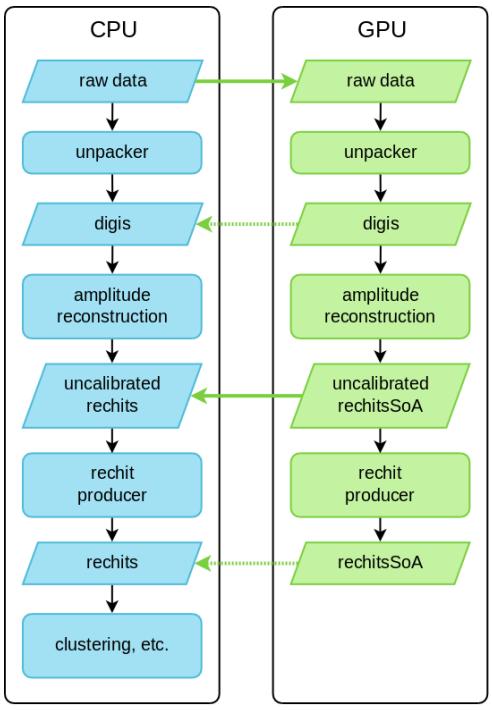

Figure 2. Data products and CMSSW framework producers of the ECAL local reconstruction on CPUs and GPUs. The dotted and solid arrows between the CPU and GPU data formats represent possible and currently used conversions between data formats, respectively.

but, for validation purposes, it is possible to run the CPU and the GPU side of the pipeline to obtain a CPU RecHit collection and a GPU one. In the following, implementation details are given for the different GPU reconstruction modules.

4.1. Unpacking module

The packed raw data from the 54 ECAL front end drivers (FEDs) is accumulated on the host in one piece of memory that is then transferred to the device. The kernel to unpack the raw data is executed and the number of unpacked digis is transferred back to the host. The unpacked digis could be transferred back to the host by a separate module if needed. There are as many blocks used as there are non-empty FEDs in the event, each having 32 threads. A loop over all channels in a FED with a stride of 32 unpacks 32 channels in parallel.

4.2. Amplitude reconstruction module

The minimisation and amplitude reconstruction is split into several kernels, rather than one monolithic kernel, to improve the performance. The kernels are chosen to maximise the parallelism at different stages of the minimisation and to avoid recomputation of common variables. Kernels for the reconstruction of the signal pulse time exist but are currently not executed at the HLT. Depending on physics analysis needs, the pulse time reconstruction may be activated in the future.

4.3. RecHits producer module

The various calibration constants and scale factors are provided by the ES, and are applied to the uncalibrated RecHits to obtain the correct energies deposited in the crystals. Some features of the CPU algorithm that involve data from neighbouring channels still need to be implemented in the GPU algorithm. Due to this discrepancy of the CPU and GPU algorithms, the GPU RecHits producer module is currently not executed in the default reconstruction. RecHits are generated on the CPU only until the ongoing work on the full implementation of the GPU version is completed.

5. Performance

A comparison between the CPU version and the GPU version of the ECAL local reconstruction was performed on an Intel Xeon Gold 6148 machine (32 CPU cores) with an NVIDIA V100 GPU [1]. A 4x speed-up was measured, with a throughput of around 2000 events/s for the GPU algorithm processing 8 events in parallel compared to around 500 events/s for the CPU-only algorithm using all 32 cores.

An event-by-event and channel-by-channel comparison of the reconstructed energies shows a relative difference between CPU and GPU values of less than 10^{-4}. For the disagreements between CPU and GPU reconstructions, the input signal typically shows high noise or pedestal issues and the amplitude reconstructed by CPU and GPU algorithms does not correspond well to the actual pulse amplitude. One of the suspected reasons for the discrepancies is the different use of single precision and double precision variables in the two algorithms.

Tests with a Run 3 HLT menu with GPU enabled algorithms for ECAL, hadron calorimeter, and pixel reconstructions, using data from 2018 on a 2x AMD EPYQ 7502 processor machine equipped with an NVIDIA Tesla T4 GPU have been performed. These tests show an overall CPU time usage reduction of 24% when using the GPU-enabled algorithms compared to CPU-only versions. The ECAL algorithm runtime is reduced from 24.5 ms for the CPU-only version to 7.4 ms when utilising the GPU as shown in Figure 3.

The ECAL local reconstruction on GPU was successfully used for data taking at the CMS HLT during collisions runs at LHC injection energy in October 2021.

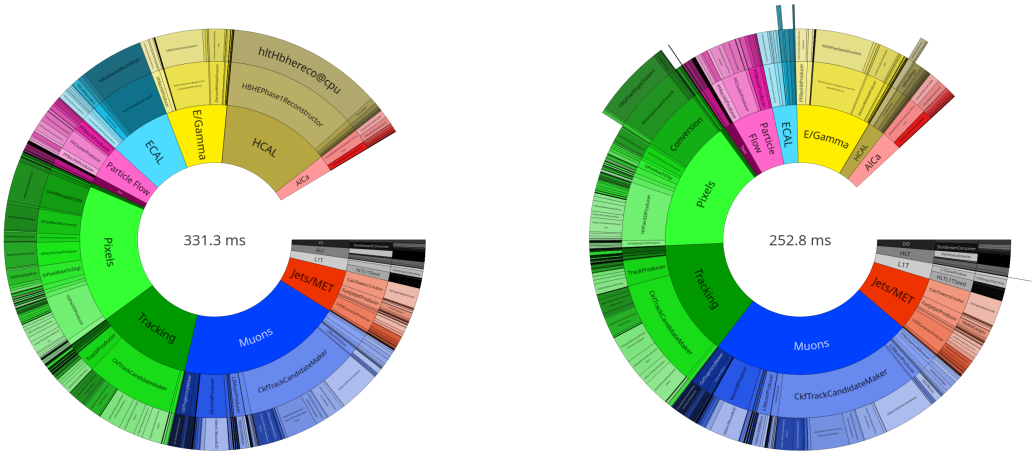

Figure 3. Runtime fractions of different object reconstruction algorithms for a representative Run 3 HLT menu. Both CPU-only (left) and GPU-enabled (right) reconstruction algorithms are tested. The ECAL local reconstruction algorithms are highlighted in cyan.

6. Summary

New technologies beyond the use of CPUs are required to continue, and go beyond, the use of sophisticated energy reconstruction and pattern recognition algorithms during LHC Run 3, and particularly the subsequent high luminosity LHC era. The CMS Collaboration is investigating the offloading of computationally intensive tasks to GPU accelerators for its software based high level trigger. The CMS software framework has been extended to allow the execution of certain tasks asynchronously on GPUs or other accelerators. The ECAL local reconstruction algorithm was among the first to be ported to a GPU architecture, using the NVIDIA CUDA toolkit, to exploit the parallelisability of the task. The reconstruction pipeline runs the code for unpacking of the raw data, pulse amplitude reconstruction, and the energy reconstruction on the GPU, and the resulting collection of reconstructed hits is transferred back to the CPU for further processing. A reduction of the per-event execution time for the ECAL local reconstruction from 24.5 ms for CPU-only code to 7.4 ms has been measured on a test system equipped with an NVIDIA Tesla T4 GPU. The GPU enabled CMS high level trigger, including the ECAL local reconstruction algorithm, was used for data taking during proton-proton collision runs in October 2021. It will be used throughout LHC Run 3, providing valuable experience for its widespread adoption during the high luminosity LHC era.

References

[1] CMS Collaboration 2021 The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger Tech. rep. CERN Geneva URL `https://cds.cern.ch/record/2759072`

[2] CMS Collaboration 1997 The CMS electromagnetic calorimeter project: Technical Design Report Tech. rep. CERN Geneva URL `https://cds.cern.ch/record/349375`

[3] CMS Collaboration 2020 *JINST* **15** P10002

[4] CMS Collaboration 2013 *JINST* **8** P09009

[5] CMS Collaboration CMS-SW `https://github.com/cms-sw/cmssw/` accessed: 2021-11-23

[6] Bocci A, Dagenhart D, Innocente V, Jones C, Kortelainen M, Pantaleo F and Rovere M 2020 *EPJ Web Conf.* **245** 05009. 7 p

[7] NVIDIA CUDA C++ Programming Guide `https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html` accessed: 2021-11-23