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

T. Reis for the CMS Collaboration

STFC Rutherford Appleton Laboratory - Harwell Campus, Didcot, OX11 0QX, United Kingdom

ACAT 2021 – 29th Nov. - 3rd Dec. 2021, Daejeon (Republic of Korea), virtual

The case for heterogeneous computing in CMS

The **higher LHC luminosity expected in Run 3** (2022+) and the consequently 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**.

This problem will become even more acute during **Phase 2 of the LHC** (2027+), where significantly higher luminosity, pileup and input rates will be delivered to the HLT than foreseen during Run 3. **The processing power** required to perform online event reconstruction in this environment **exceeds the expected increase in processing power for conventional CPUs**, requiring an alternative approach.

CMS is investigating the widespread use of **heterogeneous architectures, including GPU accelerators**, to satisfy the needs of the Phase 2 reconstruction. **A prototype system will be operated at the HLT during Run 3** to validate the approach and to gain experience in optimising the reconstruction algorithms to the different processor architectures.

The **local reconstruction algorithm of the CMS electromagnetic calorimeter (ECAL)** [1] is among the first to be implemented on heterogeneous platforms since it is well suited for parallel execution.

Adapting the CMS Software framework for GPU algorithms

The reconstruction, offline and at the HLT, is preformed with the **CMS Software (CMSSW) framework** [3]. Its collision event data based architecture provides different types of modules that can contain algorithms for reconstruction, analysis, and event filtering.

For the reconstruction of a quantity, a module of the EDProducer type typically takes some data product from the event is input, as well as some eventual additional parameters like, e.g. calibration constants, and produces a new data product as its output, that is then put back into the event.

External workers

To accommodate for offloading of work from the CPU to GPUs or other accelerators a **modification of the EDProducer for external workers 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 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** of a job and execute the GPU enabled producers possible, falling back to the CPU producers if the job runs on a machine without GPU [4].

GPU-friendly data formats

New data formats for digis and (uncalibrated) RecHits implemented as **Structs-of-Arrays (SoA)** to profit from high memory bandwidth of GPUs and read variables from several objects in one operation. **Conversion modules** produce traditional Array-of-Structs (AoS) data formats to be used by 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. GPU optimized **SoA versions of the various ECAL conditions formats** were implemented. The conditions necessary for the algorithm running on the GPU are packaged in the *acquire()* function and transferred to the device together with the event data.

Performance

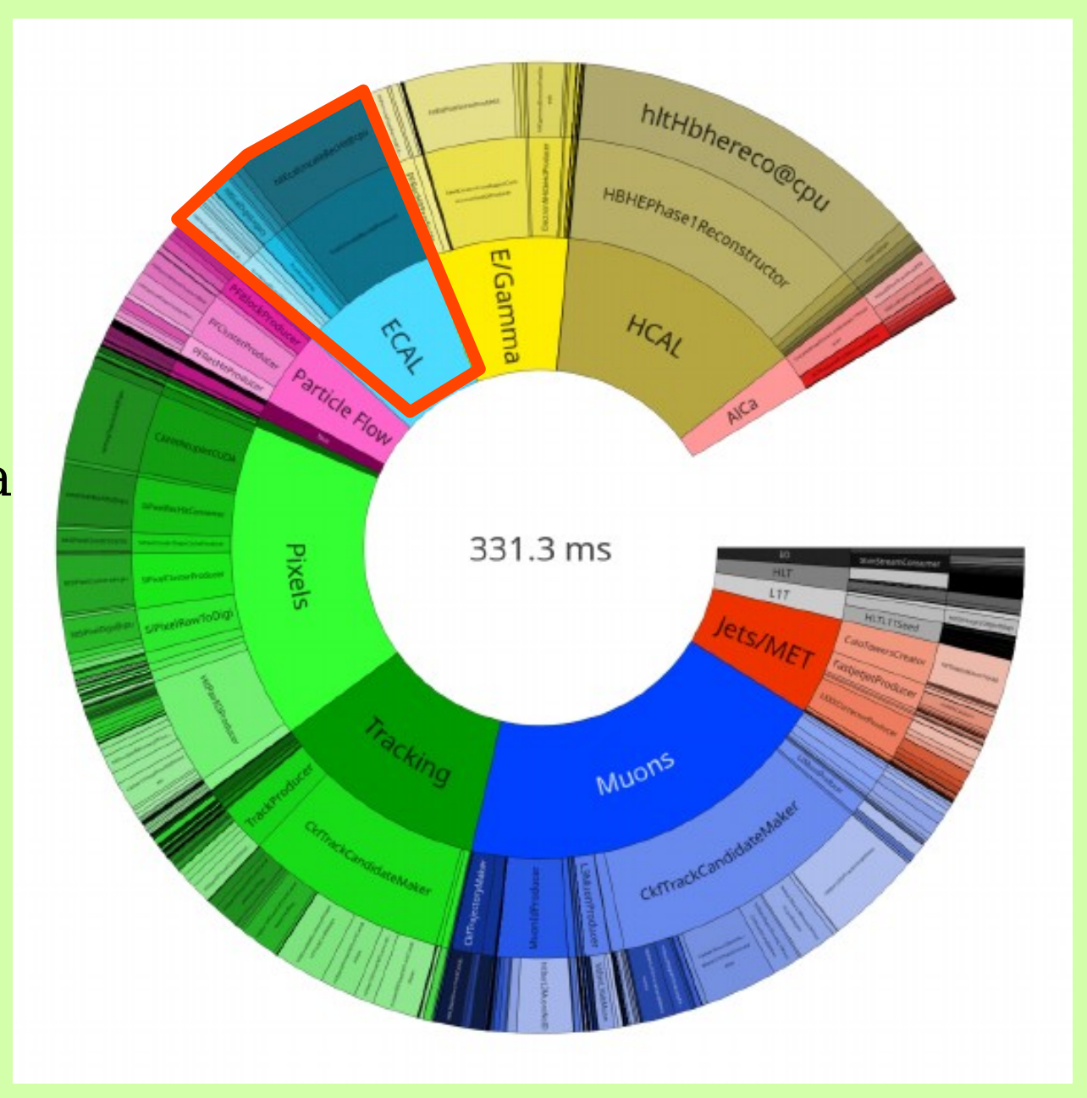
A **comparison between the CPU version and the GPU version** of the local reconstruction was performed on an Intel Xeon Gold 6148 machine (32 CPU cores) with and NVIDIA V100 GPU [6]. A **4x speed-up** was measured, with a throughput of ~2000 events/s for the GPU algorithm processing 8 events in parallel compared to around 500 events/s for the CPU-only algorithm using all 32 cores.

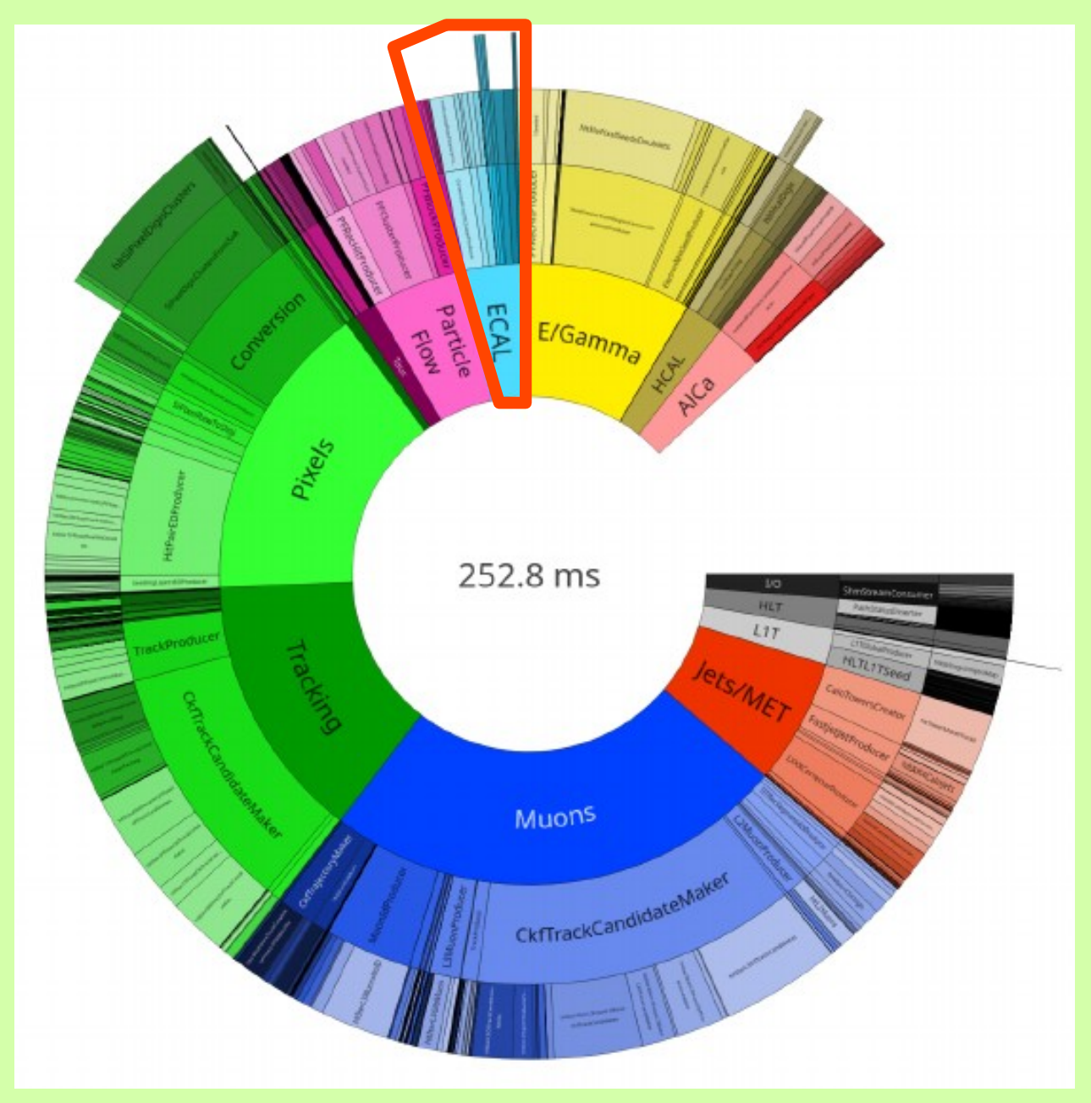
An event-by-event and channel-by-channel comparison of the reconstructed energies show a **relative difference between CPU and GPU values of less than 10^{-4}**. For the disagreements between CPU and GPU reconstructions the input signal typically shows high noise or pedestal issues and the amplitude reconstructed by CPU and GPU algorithms does not correspond well to the pulse amplitude.

Tests with a Run 3 HLT menu and Run2018 data on a 2 x AMD EPYQ 7502 machine equipped with an NVIDIA Tesla T4 GPU show an overall CPU usage reduction of 24% for all GPU enabled algorithms. **The ECAL algorithm runtime is reduced from 24.5 ms for the CPU-only version to 7.4 ms when utilising the GPU**.

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

The a



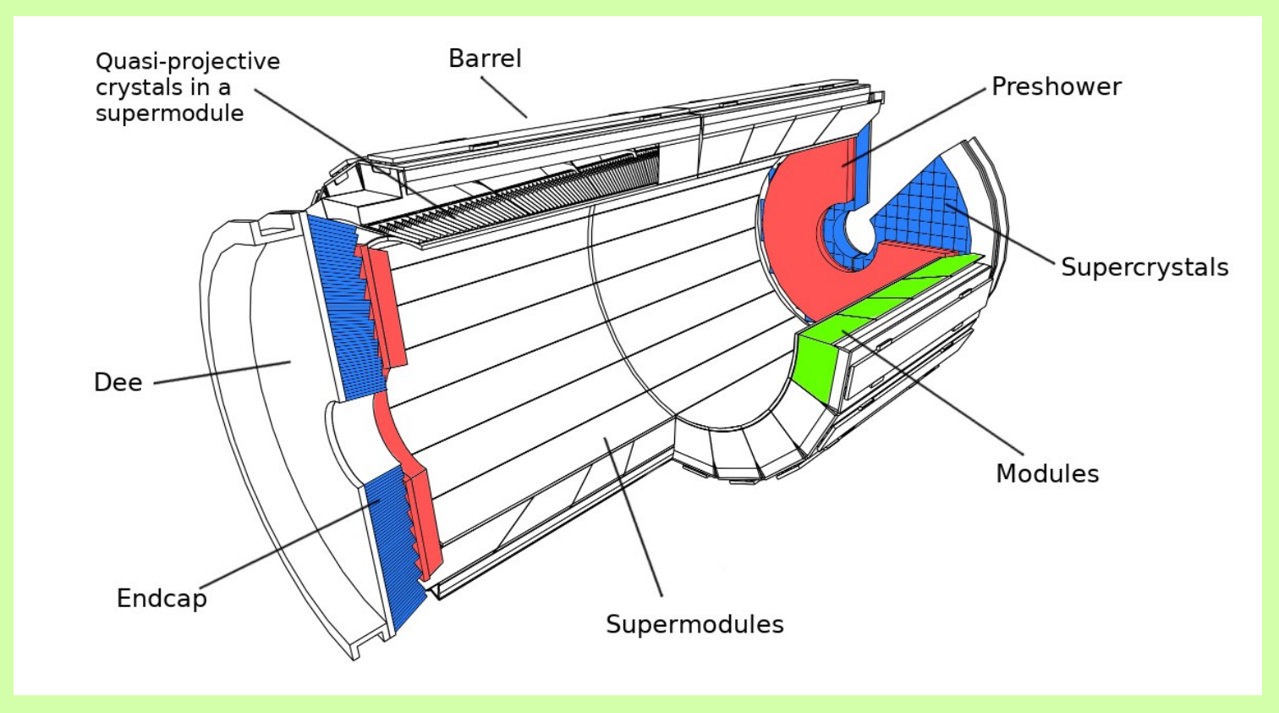


Runtime fractions of different object reconstructions in a Run 3 HLT menu for CPU-only (left) and GPU enabled menu (right). The ECAL local reconstruction contribution is highlighted by the orange frame.

The CMS Electromagnetic Calorimeter local Reconstruction

The **lead tungstate (PbWO4) electromagnetic calorimeter** of the CMS experiment consists of 61200 crystals in the barrel and 7324 crystals in each of the two endcaps, for a total of **75848 channels**.

The analogue pulse from a crystal is **sampled every 25 ns** and a fixed number of samples is read out for energy reconstruction when a trigger signal arrives.



Cut away view of the CMS ECAL detector.

The ECAL local reconstruction algorithm has three steps

- **Unpacking of the raw data** from the detector into so called digis. Digis are collection of consecutive samples from a single ECAL channel.
- **Amplitude reconstruction**. Uncalibrated reconstruction hits (RecHits) for the triggered event are calculated for each channel read out from the digis. Eventual overlapping signals from previous or later events (out-of-time pileup) are subtracted from the central event amplitude with the multifit algorithm [2].
- **Calculation of energies** from the uncalibrated RecHits. Various scale factors and calibrations are applied and RecHits in GeV units are obtained

The **multifit algorithm** is the central part of the ECAL local reconstruction. It involves the **iterative minimisation of a χ2 function**

Sample vector S

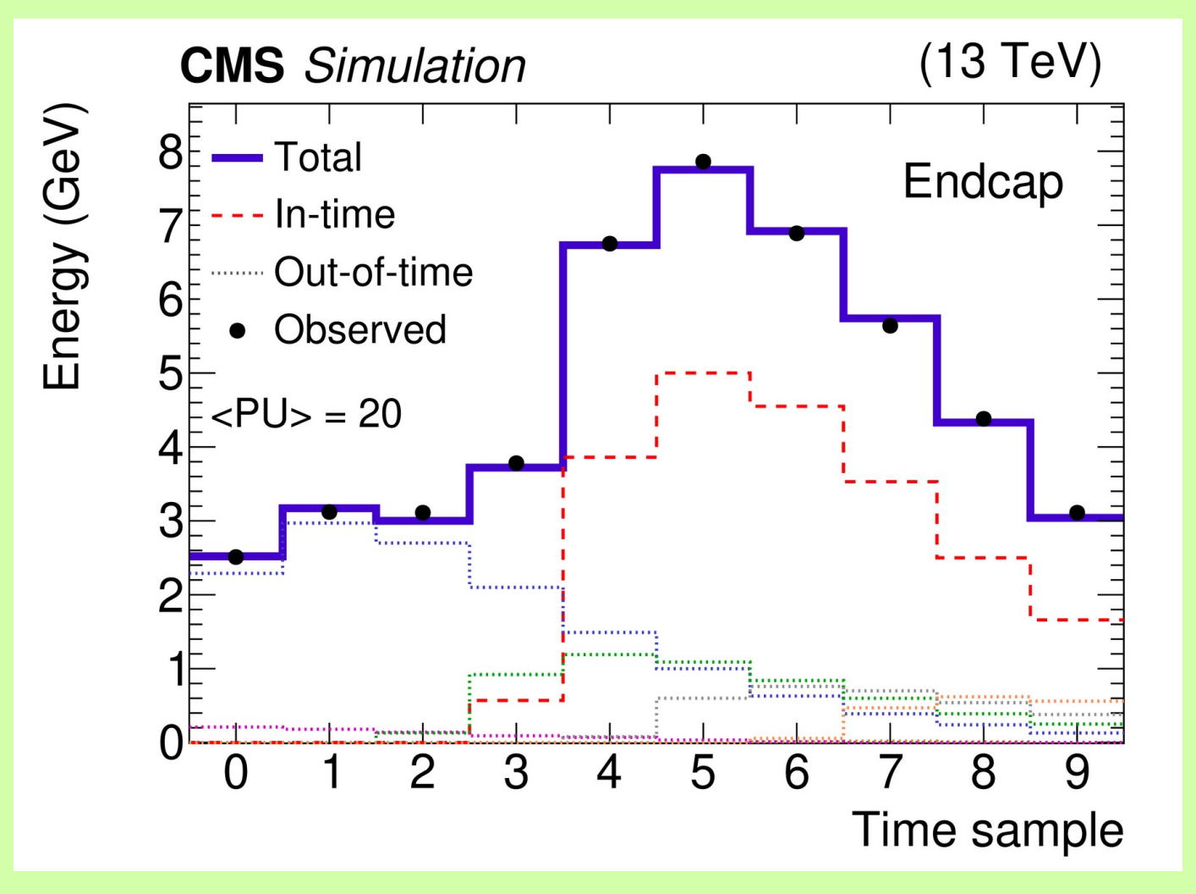
Amplitudes A_j
(1 in-time and 9 out-of-time)

$$\chi^2=\left(\sum_{j=0}^{N_{BX}} A_j \vec{p}_j-\vec{S}\right)^T \boldsymbol{C}^{-1}\left(\sum_{j=0}^{N_{BX}} A_j \vec{p}_j-\vec{S}\right)$$

Pulse templates p_j

$$\boldsymbol{C}=\boldsymbol{C}_{noise}\oplus\sum_{j=0}^{N_{BX}} A_j^2\boldsymbol{C}_{pulse}^{j}$$

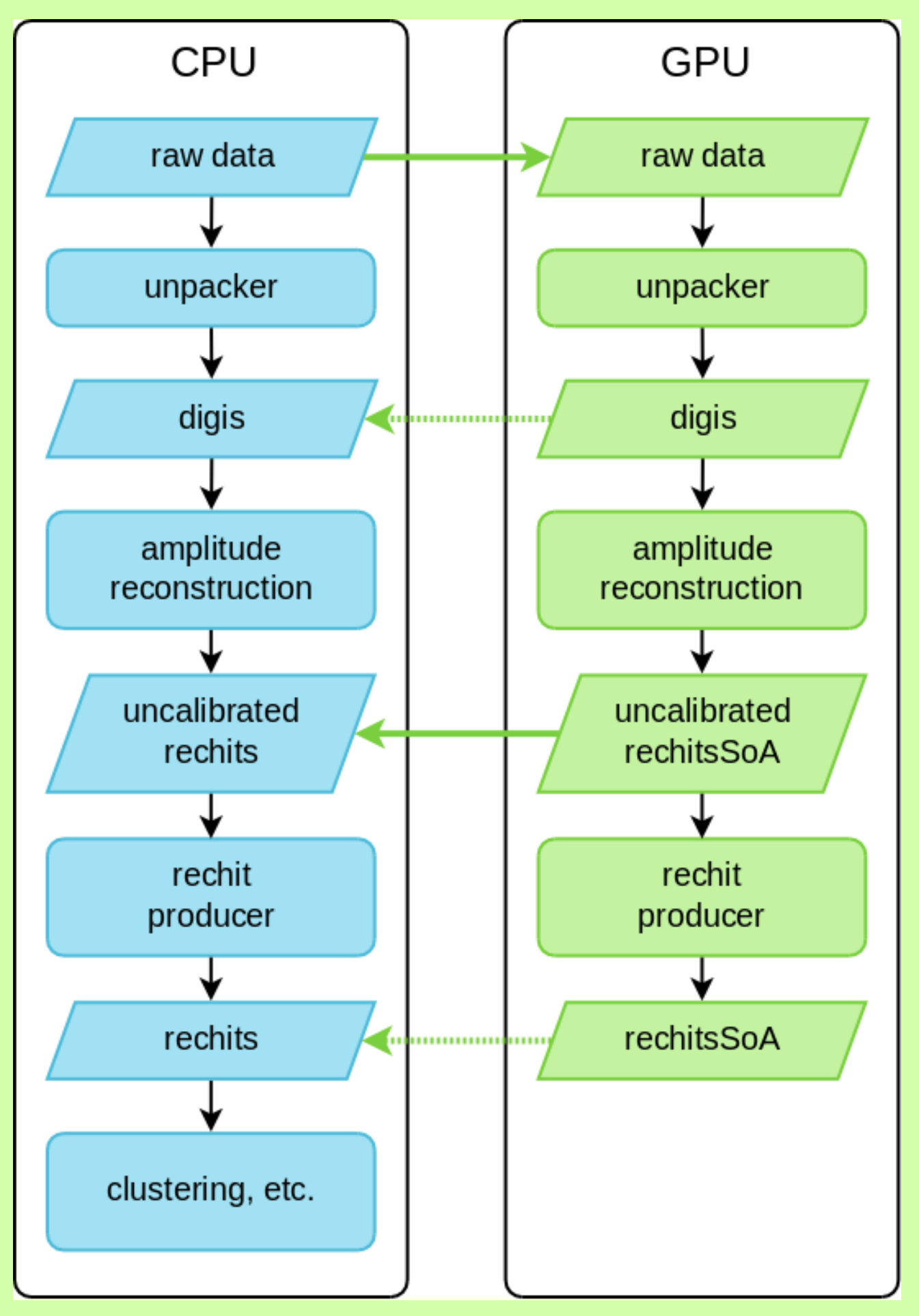
Correlation matrix



Observed samples and fitted sum of in-time and out-of-time pulses.

Implementation of the ECAL local reconstruction

All three steps of the ECAL local reconstruction are **implemented in a consecutive way as GPU algorithms** using the **CUDA toolkit** for NVIDIA GPUs [5]. This **avoids costly data transfers between host and device** and data formatting between the SoA data format and the legacy AoS formats on CPU



Data products and CMSSW framework producers of the ECAL local reconstruction on CPU and GPU. The dotted and solid arrows between the CPU and GPU data formats represent possible and actual conversions between data formats, respectively.

The 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.

The amplitude reconstruction module

The minimisation and amplitude reconstruction is **split into several kernels**. Kernels for the reconstruction of the pulse time exist but are not executed at the HLT to save time. The **kernels are chosen to maximise the parallelism at different stages of the minimisation** and avoid recomputation of common variables.

The rechits producer module

The various calibration constants and scale factors are provided by the ES and applied to the uncalibrated rechits to obtain the correct energies deposited in the channels. Some features of the CPU algorithm still need to be implemented in the GPU algorithm

References

[1] CMS Collaboration, The CMS electromagnetic calorimeter project: Technical Design Report, CERN-LHCC-97-033 (1997)
[2] CMS Collaboration, Reconstruction of signal amplitudes in the CMS electromagnetic calorimeter in the presence of overlapping proton-proton interactions, JINST 15 (2020) 10, P10002
[3] https://github.com/cms-sw/cmssw/ (accessed 2021-11-23)
[4] A. Bocci et al., Bringing heterogeneity to the CMS software framework, EPJ Web Conf. 245 (November, 2020) 05009
[5] NVIDIA, CUDA C++ Programming Guide, https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html (accessed 2021-11-23)
[6] CMS Collaboration, The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger Technical Design Report, CERN-LHCC-2021-007