Preliminary Performance Study of an Alternative Calorimeter Clustering Solution for Allen in LHCb

Núria Valls Canudas[1,*], *Xavier* Vilasis Cardona[1,**], *Míriam* Calvo Gómez[1,***], and *Elisabet* Golobardes Ribé[1,****] *on behalf of the LHCb Real Time Analysis Project*

[1]Smart Society Research Group, Engineering Department, La Salle-Universitat Ramon Llull, Quatre Camins 30, 08022 Barcelona, Spain

Abstract. The LHCb experiment has recently started a new period of data taking after a major upgrade in both software and hardware. One of the biggest challenges has been the migration of the first part of the trigger system into a parallel GPU architecture framework called Allen, which performs a partial reconstruction of most of the LHCb sub-detectors. In Allen, the reconstruction of the Electromagnetic Calorimeter (ECAL) sub-detector is used in many selection algorithms, but its efficiency is currently 10% lower than the full reconstruction performed in the second stage of the trigger. In this work, we present a preliminary performance study of an alternative ECAL reconstruction algorithm implemented in Allen that complements the current algorithm to maximise the reconstruction efficiency and also minimise the impact on the throughput rate.

1 Introduction

The LHCb experiment is one of the main particle detectors located at the LHC [1]. In the recent Upgrade I, LHCb has implemented a unique approach to data taking and analysis called Real-Time Analysis (RTA) [2] able to process events at 30 MHz. This approach is used as a trigger system to reconstruct and select specific physics events for further analysis and storage. This approach is segmented into two stages. The first part, named high level trigger 1 (HLT1), processes the full readout of the detector at 30 MHz. It is build in a GPU CUDA framework called Allen [3]. After HLT1, there is a buffer that allows to compute in quasi real-time the best alignment and calibration parameters of the detector, in order to achieve a full offline quality reconstruction in the second stage of the trigger (HLT2).

To make an accurate selection of the events to further process, HLT1 makes a preliminary reconstruction of the events, which include the reconstruction of tracks, associating them to primary and secondary vertices and performing a muon particle identification. All the algorithms in HLT1 are designed to run efficiently across parallel architectures [4, 5, 6]. Although the reconstruction of clusters from the Electromagnetic calorimeter [7] was not included in the initial stages of HLT1, other experiments within the collaboration have demonstrated the feasibility of efficient calorimeter clustering algorithms for parallel architectures [8, 9, 10].

[*]e-mail: nuria.valls@salle.url.edu
[**]e-mail: xavier.vilasis@salle.url.edu
[***]e-mail: miriam.calvo@salle.url.edu
[****]e-mail: elisabet.golobardes@salle.url.edu

https://doi.org/10.1051/epjconf/202429502023

However, the calorimeter clustering currently implemented in HLT1 is very preliminary and has a lower efficiency when compared to the reconstruction done in HLT2.

This work aims to improve the current HLT1 calorimeter clustering algorithm by adapting the same logic processes of the Graph Clustering algorithm used in the HLT2 reconstruction [11], into the CUDA algorithm. Three different approaches are proposed in order to implement the cluster overlap separation, leading to three different versions of the algorithm. The insights of those proposals are addressed in this article, together with results regarding efficiency, performance and throughput impact inside the LHCb framework.

2 ECAL reconstruction in HLT1

The calorimeter clustering implemented in HLT1 consists in searching for the cluster seeds and building clusters with the 3×3 neighbors that are sitting around the seed. Comparing this approach to the current HLT2 reconstruction, the most relevant difference is that the overlap cases are not taken into account. This means that if there is an overlap cell in between of two clusters, the energy of that cell is accounted twice, once for every cluster. Therefore, the total energy of the resulting clusters may be higher than the true energy of the particles.

To identify the cluster seeds, the current algorithm retrieves the distance one neighbors of each cell and checks if it has the maximum value among them. If it is the case, and the cell value is greater than a threshold of 10 ADC counts[1], that cell is accounted as a cluster seed. The accounting must be atomic in order to give a unique identifier to each cluster seed. This operation is parallelized for each energy digit in an event. Then, to build the reconstructed clusters, a parallel loop iterates for each cluster seed and retrieves again its distance one neighbors. If a neighbor's energy is higher than a certain threshold, that cell is accounted as part of the cluster.

Depending on the energy value set as the threshold for the neighbors of a seed, the performance of the reconstruction will be considerably affected. To make an estimation of this effect, Figure 1 shows the evolution of the algorithms efficiency as a function of the neighbors threshold value. It can be seen how lowering the threshold value considerably increases the efficiency. However, the maximum efficiency is achieved when the threshold has 10 ADC value, not zero. This suggests that filtering some of the low energy digits actually increases the number of reconstructed clusters rather than accounting for all of the neighbors. In other words, this could mean that accounting for all of the digits implies reconstructing a higher energy than the truth. This gives a hint that resolving the overlap cases may improve the inefficiency seen when the threshold is lowered to zero.

Figure 1 also plots the benchmark HLT2 ECAL reconstruction efficiency provided by the Graph Clustering algorithm, which does not have any threshold value for the neighbor digits of a seed. Compared to the maximum efficiency currently achieved with the HLT1 algorithm, there are still opportunities for improvement. Therefore, the approaches presented in the following sections implement the shower overlap mechanic as an additional process in the current HLT1 ECAL clustering algorithm.

3 The method

In order to improve the reconstruction efficiency of the CUDA algorithm, the shower overlap process is implemented following the same logic used in the Graph Clustering and previous HLT2 algorithms: cells that are overlapping between two or more clusters need to be identified. Then, its energy is split between all the overlapping clusters in fractions according to the

[1]One ADC count equals 2.5 MeV according to the gain of the ECAL PMTs.

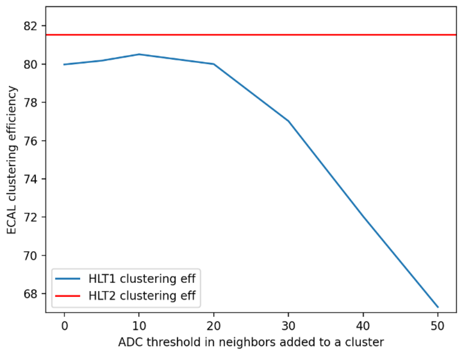

Figure 1. Efficiency of the ECAL reconstruction as a function of the threshold value for the neighbor digits of a cluster seed. The efficiency is evaluated usig $10,000$ $B \rightarrow K^{*}\gamma$ Monte Carlo events in Upgrade conditions.

energy of each involved cluster. In order to optimize the process as much as possible without the need to filter digits according to its energy, three different implementations of the shower overlap are proposed. Each one is then evaluated in terms of efficiency and throughput.

3.1 First approach

In the first approach, the seed finder kernel is maintained. As a second step, in order to identify the overlap cases, another kernel iterates again through all the digits of an event. Then, the eight neighbors of the digit cell are retrieved and the number of seeds among them is accounted. If there is more than one seed, that digit cell is identified as an overlap case. An additional vector containing atomic instances of the overlap cells is then filled. Each overlap instance contains the digit identifier, its energy value and the unique seed identifiers of the two overlapping clusters. This particular implementation only takes into account two overlapping clusters at a time to keep the size of the overlap cell variable static. Algorithm 1 summarizes the behavior of the overlap kernel for this approach in pseudo code.

Algorithm 1 Overlap kernel first approach

```
 1: overlapClusters = []
 2: for each digit ∈ eventDigits do
 3:     seedCounter = 0
 4:     overlapSeedIDs = []
 5:     for each neighbor ∈ digit neighbors do
 6:         if neighbor is a seed then
 7:             increment seedCounter
 8:             add neighbor to overlapSeedIDs
 9:         end if
10:     end for
11:     if seedCounter > 1 then
12:         atomic add of ($digit, overlapSeedIDs$) to overlapClusters
13:     end if
14: end for
```

Before the next step, a prefix sum kernel is needed to account for the number of overlap instances per event created in the new variable. As a third step of the algorithm, the original kernel that builds the clusters with the neighbors of the seeds is maintained, but with the neighbor threshold value set to zero. Finally, a fourth kernel is added in order to make a correction of the clusters energies according to the overlap cases. In this function, the kernel iterates through all the overlap cells and computes the energy correction for each involved cluster as defined in Equation 1. Since the energy of the clusters E_{cluster1} and E_{cluster2} is accounted including the energy of the overlap cell on both cases, half of E_{overlap} is subtracted from each cluster to avoid accumulating twice the energy of the overlapping cell.

$$\text{correction}_{\text{cluster1}} = E_{\text{overlap}} \frac{E_{\text{cluster2}} - \frac{E_{\text{overlap}}}{2}}{E_{\text{cluster1}} + E_{\text{cluster2}} - E_{\text{overlap}}} \tag{1}$$

Finally, the correction is subtracted from the total accumulated energy of the corresponding cluster. At the end of this step, the reconstructed clusters for an event are obtained with the total energy accumulated taking into account the overlap corrections.

Two limiting factors were found in this implementation. The first one concerns that accounting for the overlap cells is an atomic function that stores data to a new variable and is only needed in the last step of the reconstruction. The second one is that in order to modify the energy of the clusters with the overlap corrections, an entire copy of the cluster vector needs to be done, which is an overhead cost added to the algorithm.

3.2 Second approach

The second approach tries to overcome the previous limitations by making a pre-calculation of the overlap energy correction before building the clusters. Initially, the first local maxima kernel is maintained. As a second step, instead of iterating through the digits of an event, we take advantage of the fact that the number of clusters of the event is already known. Moreover, it is certain that any overlap cell will be in the neighborhood of a seed. Therefore, the kernel iterates through all the seeds of the event and retrieves its neighbors, which are potential overlap candidates. For each neighbor, its own distance one neighbors are retrieved, and we account for the number of other seeds among them and accumulate its energy value in a separate variable. If the counter is higher than one, an overlap cell is identified. However, instead of an atomic accounting of the overlap cell, this approach directly computes the overlap correction that needs to be applied to the cluster from the initial seed. In this function, the total energy of the clusters around the seeds has not been computed yet. Therefore, the overlap correction is approximated using the energy of the overlapping seeds as an estimation of the cluster's energy, following Equation 2.

$$\text{correction}_{\text{cluster1}} = E_{\text{overlap}} \frac{E_{\text{seed2}}}{E_{\text{seed1}} + E_{\text{seed2}}} \tag{2}$$

Since the energy of all the overlapping seeds found in the neighbourhood of a cell is accumulated, this approach is not limited to two overlapping clusters. Algorithm 2 summarizes in pseudo code the mentioned behavior.

Although the computed correction is not as accurate as taking into account the total energy of the clusters, this allows to simplify the overlap identification process and the number of intermediate variables needed.

As a third step, the original cluster building kernel is modified to directly subtract the energy correction at the time of accumulating the energy of all of the neighbors of a seed.

3.3 Third approach

The third approach consists of adding a small modification to the second approach such that the energy corrections is computed using the total energy of the clusters instead of only using

Algorithm 2 Overlap kernel second approach

```
clusterCorrections = []
for each seed ∈ eventSeeds do
    for each $n1$ ∈ seed neighbors do
        seedCounter = 0
        seedsEnergy = 0
        for each $n2$ ∈ $n1$neighbors do
            if $n2$ is a seed and $n2$ =! seed then
                increment seedCounter
                add $n2.energy$ to $seedsEnergy$
            end if
        end for
    end for
    if seedCounter > 0 then
        correction = $n1.energy \frac{seedsEnergy}{seed.energy+seedsEnergy}$
        $clusterCorrections[seed]+ = correction$
    end if
end for
```

the energy of the seed cells. To do so, the first kernel is updated to accumulate the energy of the neighbour cells or each event digit at the time of checking the local maxima. If the digit is a local maxima, the accumulated energy is also stored in the seed object, together with its unique identifier.

Then, in the second kernel, once an overlap cell is identified, the energy correction associated to a cluster can be computed as in the first approach, given that the total energy of the clusters is known. Therefore, the overlap kernel for this approach follows Algorithm 2 except for the correction computation in line 14 that follows Equation 1. Finally, the third kernel is maintained as in the second approach.

4 Results

The three presented approaches have been implemented inside the LHCb Allen framework and evaluated in terms of efficiency, energy and position resolution and throughput impact. The efficiency measurement is evaluated as the fraction of reconstructed particles over the number of reconstructible particles in a set of events, where a reconstructible particle is a photon that has deposited at least 90% of its energy in the ECAL and it is considered reconstructed if there is a cluster matching at least 90% of its energy. For this purpose, a total of $50,000$ events from $B \to K^{*}\gamma$ Monte Carlo simulations in Run 3 conditions have been used. The measurement of the throughput as events processed per second has been done in a GPU shared environment using an NVIDIA GeForce RTX 2080 Ti. The throughput is obtained as the maximum value among 20 independent executions of the HLT1 default sequence evaluating 1000 events, including the respective calorimeter reconstruction algorithms. Table 1 shows the efficiency and throughput values of the three proposed approaches as well as the metrics for the current HLT1 ECAL clustering. Additionally, the efficiency of the Graph Clustering algorithm used in the HLT2 reconstruction is evaluated using the same simulation samples for comparison purposes.

The results in terms of efficiency show that there is an improvement in the number of reconstructed clusters in all of the presented approaches that implement the shower overlap resolution. From the three of them, the second approach has the least improvement in efficiency. This is expected since the overlap corrections are computed using only the energy of

Algorithm	**Efficiency (%)**	**Throughput (events/second)**
HLT1 original	80.51 ± 0.29	$99,911.03$
HLT2 Graph Clustering	81.54 ± 0.28	-
First approach	81.17 ± 0.29	$96,515.43$
Second approach	81.04 ± 0.29	$97,383.57$
Third approach	81.32 ± 0.29	$97,141.25$

Table 1. Performance in terms of efficiency and throughput of the HLT1 clustering algorithm, the three proposed approaches and the Graph Clustering algorithm in HLT2.

the seeds. Although the correction is computed with the same formula in the first and third approaches, the last one has a higher efficiency. This can be explained given that the first approach only takes into account a maximum of two overlapping clusters on each overlapping cell, whereas in the second and third approaches, any number of overlapping clusters in the neighbourhood of an overlapping cell will be accounted. Therefore it is expected that the third approach gives the highest accuracy among the three. The results achieved show an improvement in the efficiency which is compatible with the Graph Clustering within errors for the three proposed approaches.

In terms of throughput, all of the proposed approaches entail a reduction of the throughput. Which is expected since the implementation of the shower overlap is adding complexity to the reconstruction process and adding more kernel functions in the trigger chain. Giving a little more of detail, the first approach is the one with a highest impact lowering the original throughput in 3.4%. The second approach has the smallest impact representing only a 2.5% of throughput reduction. The third proposal stays in between of the two with a throughput reduction of 2.8%. Overall, the decrease in throughput is not large and could be accepted in the current HLT1 sequence.

Regarding the resolution of the reconstructed clusters, $80,000$ simulation samples from $B \to K^{*}\gamma$ in upgrade conditions have been used to evaluate the difference in position on the X and Y axis and the difference in energy as a percentage. The study accounts for all the clusters with a match fraction higher than 0.9 since it is the standard match threshold for a cluster to be considered reconstructed in terms of efficiency. Figure 2 shows the energy resolution for the three presented approaches as well as the current ECAL HLT1 reconstruction and the Graph Clustering algorithm from HLT2 without any cluster corrections. The mean of the Graph Clustering distribution has been taken as reference zero value.

Upon analyzing the mean energy resolutions, a negative bias is observed in the original HLT1 clustering when compared to the Graph Clustering approach. This discrepancy can be attributed to the original implementation's lack of overlap separation and the exclusion of digits with less than 10 ADCs from the clusters. As a result, the total energy of the clusters is reduced, impacting the energy resolution more than the overlap itself. In contrast, the presented approaches, which incorporate overlap separation and do not impose any energy cut on the digits, show energy resolutions that are much closer to the performance of the Graph Clustering method. The differences between them are minimal, but the third approach ($A2\_1$) has the closest resolution to the Graph Clustering performance.

While the standard deviation of the distributions appears nearly identical, it alone cannot serve as an indicator of improvement. The efficiency, on the other hand, plays a crucial role, particularly as the evaluated samples already have a match fraction of at least 0.9. In this context, a higher efficiency indicates that a greater number of samples are above the 0.9 match fraction threshold. However, this does not necessarily imply that the new samples

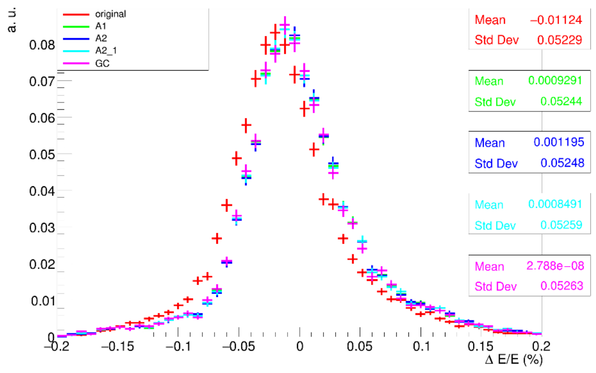

Figure 2. Normalized histograms of the energy resolution with no corrections for clusters with a match fraction over 0.9 using γ samples.

have better resolution; rather, they are now successfully reconstructed, which was not the case before.

Regarding position resolution, the plots from Figure 3 do not show any significant difference between the studied methods. However, they are all close to the Graph Clustering performance, taken as a reference.

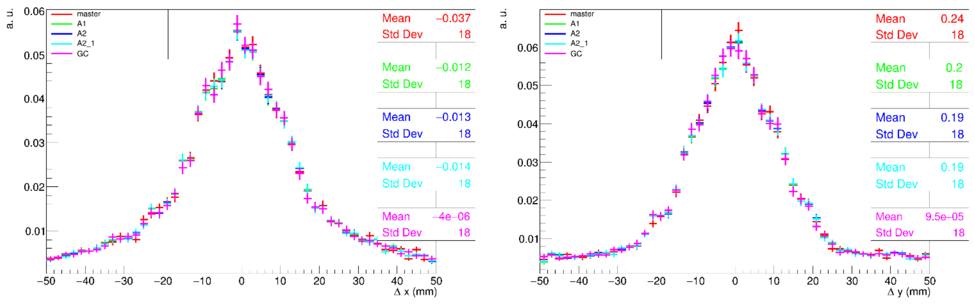

Figure 3. Normalized histograms of the X axis resolution at the left and the Y axis resolution at the right. Both using γ samples and clusters with a match fraction over 0.9 with no corrections.

5 Conclusions

It this work, three alternative clustering algorithms that implement the calorimeter shower overlap have been designed, developed and tested inside the LHCb Allen framework. The results demonstrate that the addition of the shower overlap in the reconstruction improves the efficiency of the clusters with a minor throughput decrease. Among the three approaches presented, the third one has the best trade-off between efficiency and throughput, making it a compelling improvement for the current HLT1 reconstruction.

However, the presented approaches still have a notable difference with the HLT2 reconstruction, which is the capacity to expand the shape of the clusters to more than 3×3 cells.

This characteristic benefits the reconstruction of the merged π^0 particles that arrive at the calorimeter as two closely-spaced photons. Therefore, further studies should explicitly evaluate the reconstruction performance of π^0s using the proposed algorithms and investigate the impact of expanding the cluster shapes on the reconstruction of such cases.

As the prospect of future upgrades draws near, the need to enhance and optimize data reconstruction algorithms becomes imperative. In this context, parallel architectures offer a substantial advantage in accelerating these algorithms. However, the translation from the C++ CPU-based Graph Clustering to CUDA is not a straightforward task.

The purpose of this work is to establish a starting point for exploring alternative reconstruction algorithms for calorimeter clustering that can achieve the performance levels of the current HLT2 algorithms while efficiently running on parallel architectures. By addressing this challenge, we aim to pave the way for improved data reconstruction methods in the LHCb experiment and the HEP community.

References

[1] I. Bediaga et al. *Framework TDR for the LHCb upgrade: technical design report*. Tech. rep. LHCb-TDR-012, 2012.

[2] R. Aaij et al. “A comprehensive real-time analysis model at the LHCb experiment”. In: *Journal of Instrumentation* 14.04 (2019), P04006. DOI: 10.1088/1748-0221/14/04/P04006.

[3] LHCb collaboration. “LHCb Upgrade GPU High Level Trigger Technical Design Report”. In: CERN-LHCC-2020-006 (2020).

[4] D. H. Cámpora Pérez, N. Neufeld, and A. Riscos Núñez. “Search by triplet: An efficient local track reconstruction algorithm for parallel architectures”. In: *Journal of Computational Science* 54 (2021), p. 101422. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2021.101422.

[5] P. Fernandez Declara et al. “A Parallel-Computing Algorithm for High-Energy Physics Particle Tracking and Decoding Using GPU Architectures”. In: *IEEE Access* 7 (2019), pp. 91612–91626. DOI: 10.1109/ACCESS.2019.2927261.

[6] R. Aaij et al. “Allen: A High-Level Trigger on GPUs for LHCb”. In: *Computing and Software for Big Science* 4.1 (2020). DOI: 10.1007/s41781-020-00039-7.

[7] LHCb collaboration. “LHCb PID Upgrade Technical Design Report”. In: CERN-LHCC-2013-022 (2013).

[8] ATLAS Collaboration. “Topological cell clustering in the ATLAS calorimeters and its performance in LHC Run 1”. In: *The European Physical Journal C* 77.7 (July 2017). DOI: 10.1140/epjc/s10052-017-5004-5.

[9] T. Reis for the CMS Collaboration. “Developing GPU-compliant algorithms for CMS ECAL local reconstruction during LHC Run 3 and Phase 2”. In: *Journal of Physics: Conference Series* 2438.1 (2023), p. 012027. DOI: 10.1088/1742-6596/2438/1/012027.

[10] Z. Chen et al. “GPU-based Clustering Algorithm for the CMS High Granularity Calorimeter”. In: *EPJ Web Conf.* 245 (2020). Ed. by C. Doglioni et al., p. 05005. DOI: 10.1051/epjconf/202024505005.

[11] N. Valls Canudas et al. “Graph Clustering: a graph-based clustering algorithm for the electromagnetic calorimeter in LHCb”. In: *The European Physical Journal C* 83.2 (2023), p. 179. DOI: 10.1140/epjc/s10052-023-11332-1.