A vendor-unlocked ITS GPU reconstruction in ALICE

Matteo Concas[1,*]

[1]CERN

Abstract.
In LHC Run 3, ALICE experiment's upgrades enable recording Pb-Pb collisions at a 50 kHz rate using trigger-less continuous readout, managing a peak data rate of 1 TB/s primarily through GPU utilization. Two main phases of data processing exist: the synchronous phase, focused on TPC reconstruction consuming most computing resources, and the asynchronous phase with more GPU resources available. The Inner Tracking System (ITS) has been successfully ported to GPUs for primary vertex finding and track reconstruction, leveraging automatic code generation to support multiple GPU brands with a single code base. This work not only enhances ALICE's computational capability but also sets a precedent for offloading reconstruction tasks to GPUs in other detectors.

1 Introduction

During the Run 3 at the Large Hadron Collider (LHC), the accelerator will deliver a rate of Pb-Pb collisions up to 50 kHz. The A Large Ion Collider Experiment (ALICE) [1] is adopting a novel data acquisition strategy, the "continuous readout" consisting of a *trigger-less* approach that steadily registers input from every sub-detector. Data are therefore split along the time dimension in so-called *timeframes* of the duration of few ~ ms. The aim is to record an integrated luminosity greater than 10 nb^{-1} of minimum bias events, to enable the study of very rare physics signals.

This translates into a huge sample of collected data, corresponding to a factor ×50 times more minimum bias Pb-Pb collisions. To support and implement such a strategy, a lot of challenges both on the hardware and software front have been faced. The ALICE computing model for Run 3 relies on a completely rewritten framework encompassing both the online and the offline software in a single stack called O^2[2]. As briefly illustrated in Fig. 1, the raw data input from the detectors is reduced from ~3.5 TB/s in the First Level Processing infrastructure, down to less ~900 GB/s. Then the Event Processing Nodes (EPNs) farm performs a data calibration and reduction via reconstruction. Such a reconstruction is split in two phases: the *synchronous phase* performing the calibration and compression of the data simultaneously with the data taking and the *asynchronous phase* that operates the full processing of data previously staged on a temporary buffer. The usage of Graphics Processing Units (GPUs) is envisioned to supercharge the computing capabilities of the EPN. During the synchronous reconstruction the processing time is dominated by the data processing of the Time Projection Chamber (TPC). The efficient usage of remaining GPU cycles is a key point to maximise the efficiency in the usage of the EPNs. During the asynchronous reconstruction,

[*]e-mail: matteo.concas@cern.ch

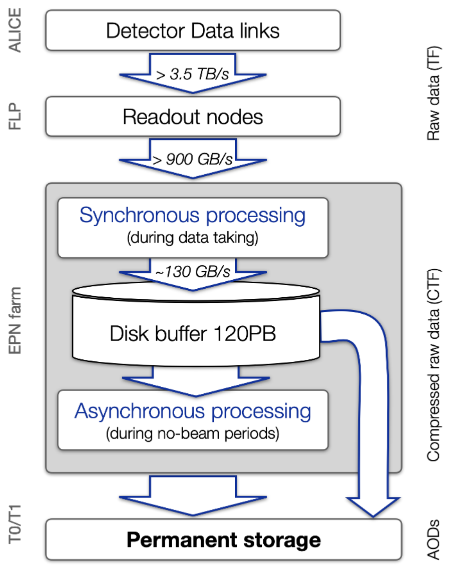


Figure 1. Schema of the data processing operated via synchronous and asynchronous reconstruction on the EPN farm.

the estimated fraction of available GPUs increases. To this extent the ALICE Inner Tracking System (ITS), among with other detectors, is porting its reconstruction on GPUs.

2 ITS reconstruction software for Run 3

The ALICE upgraded ITS is a brand new cylindrical silicon pixel detector counting more than 10 m^2 of sensitive area and a total of ~12.5 billion pixels. Its main goal is to provide spatial measurement of charged particles crossing the detector in the form of *clusters* of fired pixels. It operates within the continuous readout regime and it is responsible for the identification of the interaction region, providing an improved resolution of the primary vertex position, and is capable of tracking the charged particles produced in the collision and possibly their secondary vertices.

The minimum frame of data it produces is the Readout Frame (ROF), which corresponds to the data recorded in a $\sim 4\mu$s time interval. A single timeframe may then include from hundreds to up to few thousands of ROFs, depending on the collision system and the experimental settings.

2.1 Seeding vertexer and tracker

The ITS reconstruction is designed to operate on a single timeframe at a time, performing the preliminary interaction point identification (vertexing) and the track reconstruction (tracking). The same standalone vertexing and tracking algorithm is run during the synchronous and asynchronous processing. During the synchronous phase the goal is to reconstruct 1% of tracks, which are needed for TPC online reconstruction and calibration. The asynchronous phase targets the reconstruction of all charged particle down to low transverse momentum

(p_T) and it differs just in terms of algorithm configuration, quality selection criteria and subsequent iterations on the data. The reconstruction is divided in two main stages each constituted by multiple consecutive steps that produce a variety of intermediate objects as depicted in Fig. 2.

1. **Primary vertex seeding**: it performs a combinatorial matching of the clusters in the first three innermost layers, then a combinatorial validation followed by linear extrapolations of the found *tracklets* towards the beam line. Finally, it uses an unsupervised clustering to find the collision point(s) that are sent to the next stage.
2. **Track finding and track fitting**: it uses a combinatorial *Cellular Automaton* algorithm that exploits the vertex position to reduce the volume of combinatorics in matching the clusters of pixels. It subsequently connects segments of tracks, the *cells*, into a tree of candidates: the *roads*. Ultimately, it applies a Kalman filter to fit tracks from candidate roads and performs selection based on the quality of the fit.

Each of the intermediate sub-steps is characterised by the combinatorial nature of the problems to solve, where each object requires to be matched with others independently from its neighbours. The whole algorithm is decomposable into multiple parallel steps where

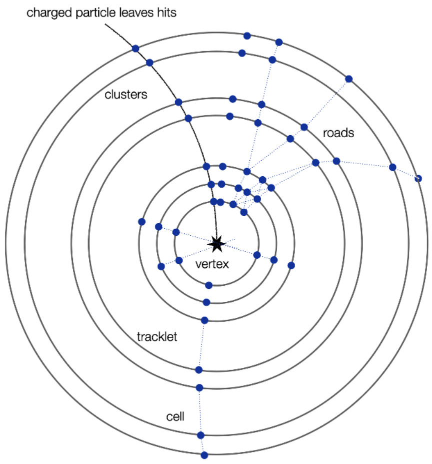
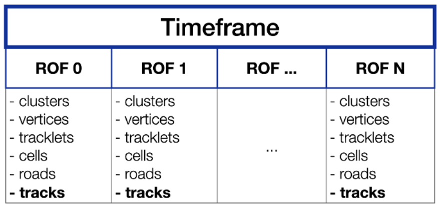


Figure 2. Visualisation of the ITS tracking intermediate objects.



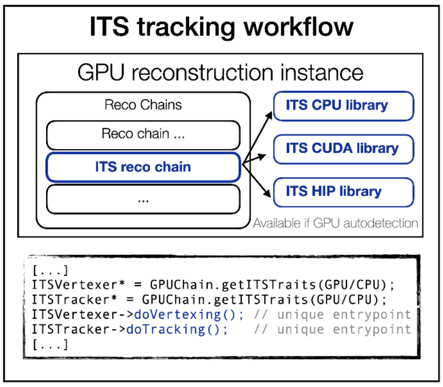
Timeframe			
ROF 0	ROF 1	ROF ...	ROF N
- clusters - vertices - tracklets - cells - roads **- tracks**	- clusters - vertices - tracklets - cells - roads **- tracks**	...	- clusters - vertices - tracklets - cells - roads **- tracks**

Figure 3. Inherent parallelism inside the processing of ITS artefacts. It applies to to the intra-frame combinatorics process and to each frame independently.

intra-frame combinatorics can be processed simultaneously and each ROF can be processed independently from the others following grid in Fig. 3. This structure makes it suitable for a scalable parallel GPU implementation.

3 Parallel implementation using GPUs

In this work, the state of the art about the porting of vertexing and tracking on GPU is presented. The current version operates as a plug-in component to the central GPU reconstruction framework. The latter provides a convenient centralised service for loading external GPU brand-dependent libraries responsible for steering the execution as shown in Fig. 3. The ITS code supports both CUDA [5] and HIP [6] via the ROCm [7] stack with a single code base thanks to the automatic on-the-fly translation as described in [4].



```
[...]
ITSVertexer* = GPUChain.getITSTraits(GPU/CPU);
ITSTracker* = GPUChain.getITSTraits(GPU/CPU);
ITSVertexer->doVertexing(); // unique entrypoint
ITSTracker->doTracking();   // unique entrypoint
[...]
```

Figure 4. Integration of the ITS GPU as a plug-in component.

3.1 Cornerstones of the GPU implementation

In order to support the natural variations of input size of the time frame in different collision systems or different experimental settings, focus has been put on having a configurable way of sub-sampling it into resizeable groups of ROFs to be processed in parallel.

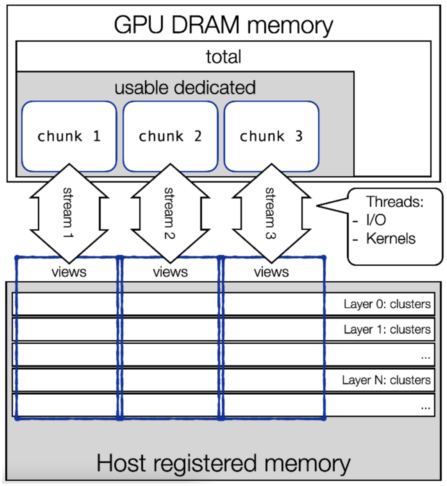


Figure 5. Dynamic subsampling in "views" of the data on multiple GPU streams.

The amount of usable memory is a parameter that is passed to the algorithm and the sizes of data "chunks" are set as a fraction of the total available memory. Multiple POSIX threads manage the computations on different streams, where the tracking instances are independent from each other. This approach not only allows for limiting the usable GPU memory in cases where the device is shared across different detectors, but also the scheduling of the computations alongside the memory transfers increases the efficiency in using the device. Ultimately, such a flexibility opens up to a generic solution via a configurable N-layer implementation of the tracking algorithm as the related scaling can be absorbed by narrower views of ROFs to be processed.

Elapsed Time [ms]	**AMD EPYC™ 7452**	**AMD Ryzen™ 9 7950X**	**AMD Instinct™ MI50**	**Nvidia™ TITAN Xp**
Vertexer	2913±376	1416±183	291±38	478±64
Tracker (Neigh. Finder)	550±71	287±37	211±27	779±105
Tracker (Full)	13756±1780	6917±893	W.I.P.	W.I.P.

Table 1. Preliminary results of the GPU implementation of the current state of the ITS tracking.

3.2 State of the development and testing

Primary vertexing is fully operational on its GPU implementation. The porting of the tracking is currently being finalised: the Cellular Automation is implemented and tested up to the "road finding" step. Track fitting is currently being updated to match the latest state of the art of the CPU code. All the code base supports and has been tested both on CUDA and HIP.

4 Preliminary results

In this section the preliminary results achieved so far by the algorithm are reported. Elapsed time is measured repeating 10 times the reconstruction of 5 timeframes from 500 kHz pp data with asynchronous reconstruction settings, corresponding to the most resource demanding configuration and a realistic scenario. Mean and the RMS of the timing distributions are measured and reported for both vertexer and tracker. For the GPU benchmark the measurement is done up to the latest step of the tracking algorithm that is currently available called "neighbour finder", responsible for identifying cells that can be merged to build road segments. The CPU reference is run in single thread configuration. For a better comparison the multi-threaded CPU version should be fully benchmarked. Results are reported in Tab. 4. The timing achieved by the GPU implementation is faster than the CPU one for the considered architectures. In the perspective of trading GPUs for CPUs this is already acceptable, as using GPUs allows for offloading this part of the reconstruction for similar or better performance.

Hereafter the performance as a function of the available memory for both vertexer and tracker is reported. The same reconstruction as used for the tests described above has been run by changing the available memory from 1GB up to the maximum available memory of each of the two cards (respectively 16GB for Nvidia and 32GB for AMD). The hardware is the same as in Tab. 4.

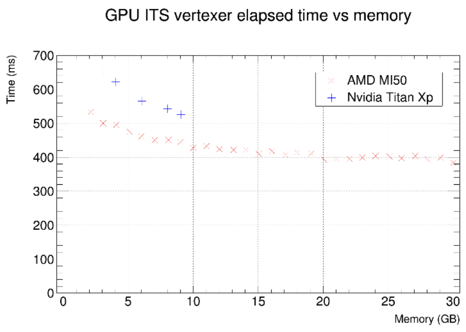


Figure 6. Scaling of the vertexer performance vs available memory.

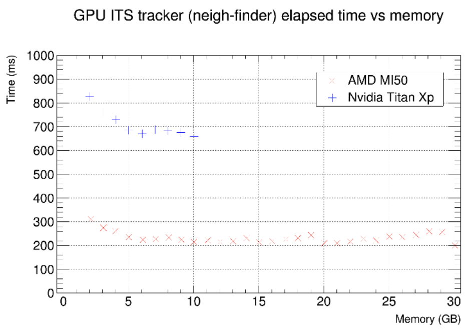


Figure 7. Scaling of the tracker performance vs available memory.

In both cases timing improves with increasing available memory, then it reaches a plateau. This is due to the fact that the overall footprint of the algorithm is limited with respect to the available memory. This important detail opens up to the possibility of running multiple steps of the barrel reconstruction on the same GPU, creating a pipeline that processes all the data on the device minimising the data transfers.

5 Conclusions

ALICE plans to extend the coverage of GPU usage in the asynchronous reconstruction. The primary goal is to increase the efficiency in using the resources when TPC does not use all the GPU resources. To achieve that the plan is to build a GPU reconstruction chain that includes all the detectors in the ALICE barrel that operate with continuous readout. Such a reconstruction framework will centrally manage the GPU resources and kernel scheduling so to make it easier to integrate additional steps (e.g.: detector matching). The ITS is finalising porting of the seeding vertexer and tracking on GPU architectures, targeting the asynchronous reconstruction. Road finding and track fitting, the last missing components, are under active development. The performance measured with pp collision data is not yet the final optimal one but shows some promising margin and a good scaling with the available device memory. GPU adoption in the ITS software chain can be further extended both in the simulation and in the data reconstruction: digitisation of the simulated signal and cluster identification from pixel data are the good candidates that are being tackled.

References

[1] ALICE Collaboration, *"The ALICE experiment at the CERN LHC", J. Inst. 3 S08002 (2008)*

[2] P. Buncic, M. Krzewicki, P. Vande Vyvre, *Technical Design Report for the Upgrade of the Online-Offline Computing System (2015)*

[3] ALICE Collaboration, *"Technical Design Report for the Upgrade of the ALICE Inner Tracking System", CERN-LHCC-2013-024 (2013)*

[4] M Concas, *"A vendor-agnostic, single code-based GPU tracking for the Inner Tracking System of the ALICE experiment", Journal of Physics: Conference Series (2023)*

[5] John Nickolls, Ian Buck, Michael Garland, Kevin Skadron, "Scalable Parallel Programming with CUDA" (2008)

[6] HIP, https://rocm-developer-tools.github.io/HIP/

[7] AMD, ROCm, https://rocmdocs.amd.com/en/latest/Current_Release_Notes/Current-Release-Notes.html