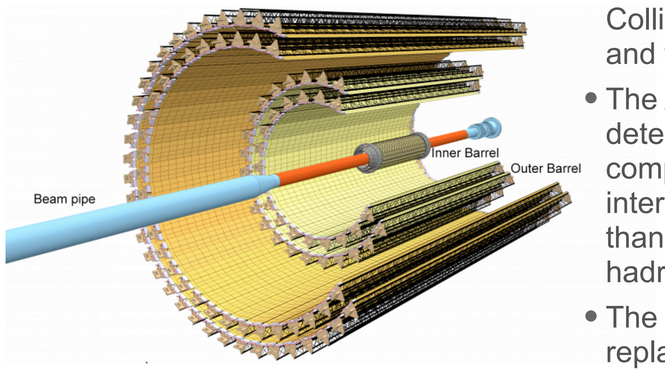
Online reconstruction of tracks and vertices using the Upgraded Inner Tracking System of ALICE at the LHC

Matteo Concas[1,2] [mconcas@cern.ch], **Maximiliano Puccio**[1,3] [mpuccio@cern.ch], **Iacopo Colonnelli**[2], **Franco Dessena**[1,2], **Stefania Bufalino**[1,2], **Massimo Masera**[1,3] on behalf of the **ALICE Collaboration**

[1]Istituto Nazionale di Fisica Nucleare, [2]Politecnico di Torino, [3]Università degli Studi di Torino

CHEP 2018 Sofia, Bulgaria

Upgrade of the ALICE Inner Tracking System



New design of the Inner Tracking System [1]

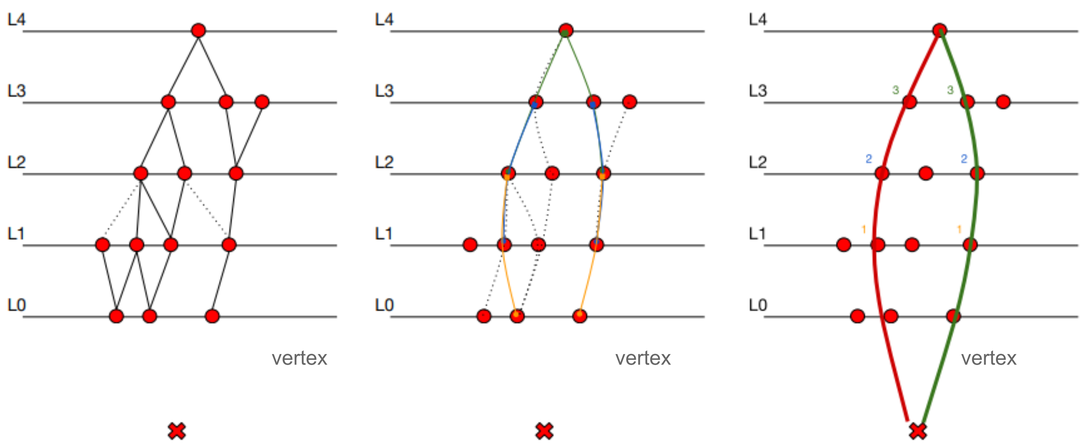
- **Run 3**: In 2020 the upgraded Large Hadron Collider (LHC) will increase Pb-Pb luminosity and frequency up to $6x10^{27}cm^{-2}s^{-1}$ at 50kHz
- The A Large Ion Collider Experiment (**ALICE**) detector will be upgraded in many of its components, to enable the read-out of all interactions and integrate a luminosity higher than 10 nb^{-1}, corresponding to about 10^{11} hadronic interactions.
- The Inner Tracking System (**ITS**) is being replaced with a new setup constituted by seven layers of Monolithic Active Pixel Sensors (MAPS)
- With respect to current ITS:
  - The innermost layer is closer to the interaction point: 39 mm → 22 mm
  - Better pointing resolution by a factor of 3(5) in rφ (z) at p_T =500 MeV/*c*
  - Better tracking efficiency and p_T resolution at low p_T

The Online-Offline (O^2) project

- The ALICE Computing Model for Runs 3-4 is driven by the need to reduce the data volume to the maximum possible extent, to minimise the computing resources requirements needed for data processing with minimum impact on physics performance
- It is therefore very relevant to perform preliminary online reconstruction to decide whether to keep or throw incoming data
- The new software stack will need to include both Online and Offline processing in order to better cope with such goals. Hence, it is necessary to design and develop new and more performant reconstruction algorithms and to implement them in a way to make it cope with the high readout rates and the large I/O throughput
- The O^2 software framework takes care to implement a distributed, parallel and staged data processing model capable to address these requirements. Its main goal is to perform detector calibration and data reconstruction concurrently with data taking
- It can be tailored to different configurations by the adaptation or addition of detector specific algorithms and to specific computing systems [2]
- The O^2 framework will also support the use of dedicated hardware accelerators (e.g. Graphic Processing Units (GPUs)), the new ITS reconstruction will benefit from this

Track reconstruction in ITS using Cellular Automata

- The reconstruction in ITS is responsible to find and classify the tracks generated by charged particles and the position of the interaction vertex
- After a preliminary vertex position estimation, needed as a seed for current tracking algorithm implementation, the tracking phase is constituted by three macro steps:



Tracklet finding

A combinatorial routine to find pairs of clusters on adjacent layers, filtering them using some cut criteria

Cell finding

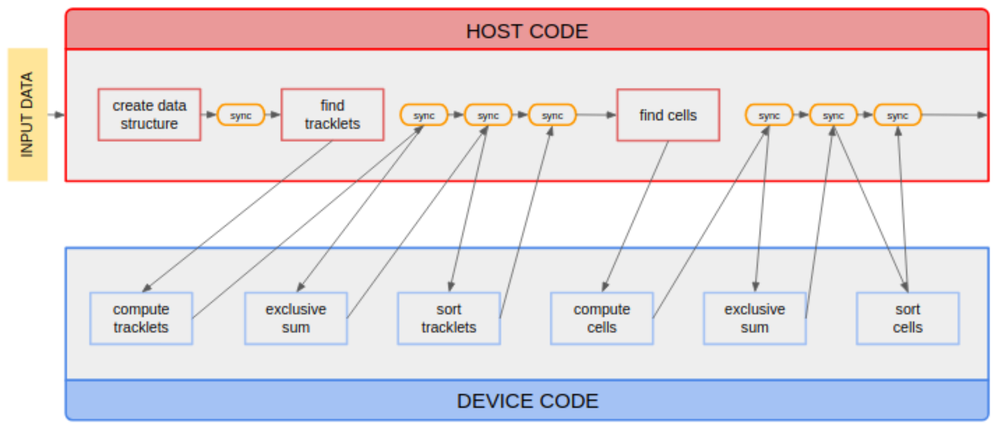
Subsequent tracklets that satisfy some filtering criteria are merged into *cells*

Track fitting

Neighbour cells are combined into track candidates a fit is later performed using a Kalman Filter

CUDA™ Implementation

- First implementation on GPU was developed using Computing Unified Device Architecture (CUDA™) [3] and the CUB library
- Tracklet and Cell finding are the most computing time demanding operations
  - Try to port these two steps on the GPU!



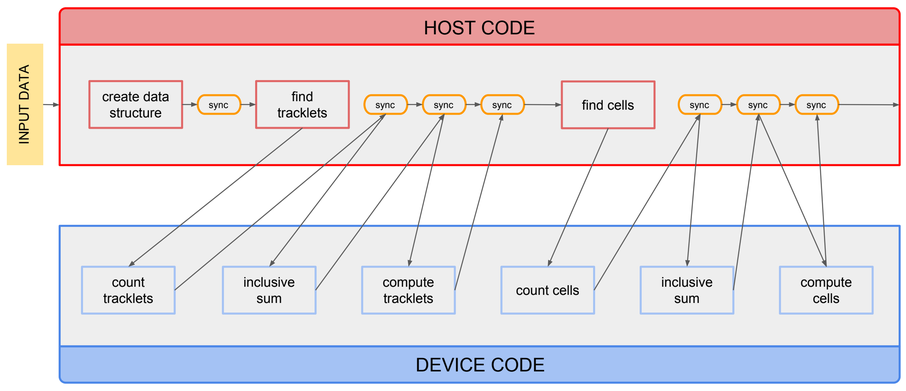
Host-device paradigm: host execution, red filled, and device execution, blue filled, must be properly synchronized

# Vertices		1	2	4	5
Context init	Min	5.3	9.1	19.3	24.9
	Mean	6.7	13.0	25.1	32.5
	Max	11.2	22.2	63.1	94.3
Tracklet finding	Min	1.8 (18.2)	6.3 (76.4)	27.0 (329.1)	44.2 (517.3)
	Mean	3.1 (39.4)	9.0 (139.0)	35.2 (549.6)	55.1 (810.0)
	Max	5.3 (70.6)	13.3 (269.3)	47.2 (1094.0)	70.5 (1641.0)
Cell finding	Min	2.0 (10.2)	6.5 (50.9)	35.4 (350.3)	60.6 (646.9)
	Mean	5.0 (28.5)	15.0 (134.5)	72.5 (862.4)	123.1 (1520.3)
	Max	7.9 (44.4)	20.9 (188.7)	104.4 (1213.6)	166.0 (2140.7)
Total	Min	9.6 (33.4)	23.4 (135.0)	84.2 (696.2)	133.3 (1186.2)
	Mean	16.7 (75.0)	40.2 (285.4)	139.5 (1437.7)	219.9 (2362.8)
	Max	27.0 (110.3)	59.9 (446.5)	220.7 (2326.1)	317.9 (3810.5)

- Host code is still serial: a CPU parallelisation of such portion of the algorithm could lead to an additional speedup
- Creation of CPU threads and context switch between them are costly operations: possible parallelisation strategies must be carefully analysed

OpenCL Implementation

- Another Implementation has been developed using the OpenCL [5]
- The algorithm has been modified to avoid the sorting of the tracklet and atomic operations
  - A “dry run” of the tracklet finding algorithm is performed in order to count the total number of tracklet reconstructed per cluster
  - A second iteration of the algorithm is used to instantiate the object (tracklet or cells) in memory already sorted for the following step

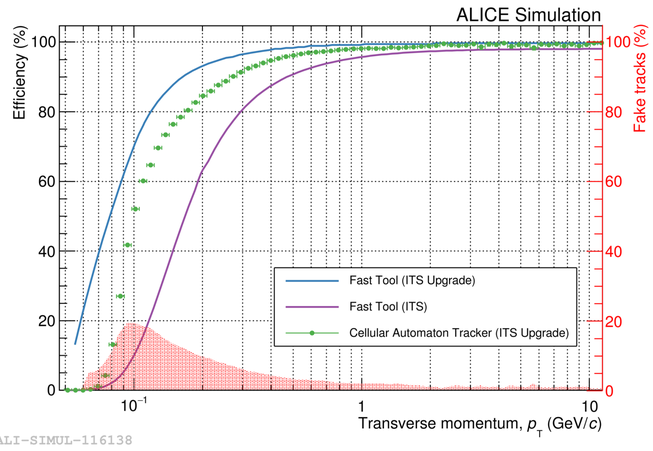


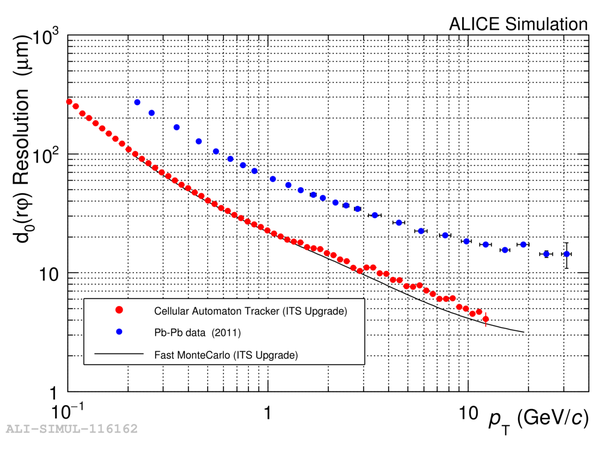
Version		Native	Sort	Boost	Serial	CUDA
Context init	min	7.0 (9.0)	4.0 (4.0)	4.0 (4.0)	3.5	5.3
	mean	10.8 (14.9)	5.02 (5.2)	4.8 (5.2)	4.7	6.7
	max	18.0 (22.0)	8.0 (6.0)	7.0 (7.0)	8.1	11.2
Tracklet find.	min	5.0 (3.0)	9.0 (19.0)	1.0 (1.0)	21.4	1.8
	mean	6.0 (7.1)	12.27 (26.7)	1.0 (3.16)	49.2	3.1
	max	11.0 (14.0)	17.0 (33.0)	1.0 (11.0)	86.1	5.3
Cell find.	min	3.0 (5.0)	8.0 (10.0)	0.0 (8.0)	9.82	2.0
	mean	8.6 (14.5)	10.09 (15.2)	2.0 (17.0)	36.96	5.0
	max	13.0 (24.0)	12.0 (17.0)	3.0 (26.0)	41.22	7.9
Total	min	15.0 (17.0)	21.0 (33.0)	5.0 (13.0)	35.1	9.6
	mean	26.4 (37.5)	28.37 (48.0)	8.8 (26.4)	82.5	16.7
	max	42.0 (53.0)	35.0 (57.0)	12.0 (40.0)	136.01	27.0

- Boost.Compute was used to reduce the OpenCL boilerplate code
- VexCL scan algorithm has been used as it is faster than the Boost.Compute version

Conclusions

- Two different algorithms and implementation strategies have been tried for the ITS tracker
- The tracking efficiency obtained in both implementation are consistent with each other and with the baseline CPU tracking algorithm
- Both implementation show a great improvement in terms of computing time with respect to the serial CPU implementation
  - A new parallel CPU version is in development to have a fair comparison of the obtained results
- Initialisation and track fitting became the new bottleneck for the reconstruction algorithm and will be further optimised





We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

[1] https://cds.cern.ch/record/1625842/files/0954-3899_41_8_087002.pdf
[2] https://cds.cern.ch/record/2011297/files/ALICE-TDR-019.pdf
[3] http://www.infn.it/thesis/thesis_dettaglio.php?tid=12030
[4] https://developer.nvidia.com/about-cuda
[5] https://www.khronos.org/opencl
[6] TBD
[7]
[8]