ALICE HLT High Speed Tracking on GPU

S. Gorbunov, D. Rohr, K. Aamodt, T. Alt, H. Appelshäuser, A. Arend, M. Bach, B. Becker, S. Böttger, T. Breitner, H. Büsching, S. Chattopadhyay, J. Cleymans, I. Das,Ø. Djuvsland, H. Erdal, R. Fearick, Ø. S. Haaland, P. T. Hille, S. Kalcher, K. Kanaki, U. Kebschull, I. Kisel, M. Kretz, C. Lara, S. Lindal, V. Lindenstruth, A. A. Masoodi, G. Øvrebekk, R. Panse, J. Peschek, M. Ploskon, T. Pocheptsov, T. Rascanu, M. Richter, D. Röhrich, B. Skaali, T. Steinbeck, A. Szostak, J. Thäder, T. Tveter, K. Ullaland, Z. Vilakazi, R. Weis, and P. Zelnicek for the ALICE collaboration

***Abstract*—The on-line event reconstruction in ALICE is performed by the High Level Trigger, which should process up to 2000 events per second in proton-proton collisions and up to 300 central events per second in heavy-ion collisions, corresponding to an input data stream of 30 GB/s.**

In order to fulfill the time requirements, a fast on-line tracker has been developed. The algorithm combines a Cellular Automaton method being used for a fast pattern recognition and the Kalman Filter method for fitting of found trajectories and for the final track selection.

The tracker was adapted to run on Graphics Processing Units (GPU) using the NVIDIA Compute Unified Device Architecture (CUDA) framework. The implementation of the algorithm had to be adjusted at many points to allow for an efficient usage of the graphics cards. In particular, achieving a good overall workload for many processor cores, efficient transfer to and from the GPU, as well as optimized utilization of the different memories the GPU offers turned out to be critical. To cope with these problems a dynamic scheduler was introduced, which redistributes the workload among the processor cores. Additionally a pipeline was implemented so that the tracking on the GPU, the initialization and the output processed by the CPU, as well as the DMA transfer can overlap.

The GPU tracking algorithm significantly outperforms the CPU version for large events while it entirely maintains its efficiency.

M. Richter, Ø. Djuvsland, H. Erdal, K. Kanaki, G. Øvrebekk, Ø. S. Haaland, D. Röhrich, A. Szostak and K. Ullaland are with the Department of Physics and Technology, University of Bergen, Norway e-mail: Matthias.Richter@ift.uib.no

T. Alt, M. Bach, S. Gorbunov, S. Kalcher, V. Lindenstruth, T. M. Steinbeck, and J. Thäder are with the Frankfurt Institut für Informatik IfI, Frankfurt Institut for Advanced Studies FIAS, Ruth-Moufang-Str. 1, 60438 Frankfurt, Germany.

S. Böttger, T. Breitner, U. Kebschull, M. Kretz, C. Lara, R. Panse, J. Peschek, D. Rohr, R. Weis, and P. Zelnicek are with the Kirchhoff Institute of Physics, University of Heidelberg, Germany

K. Aamodt, S. Lindal, T. Pocheptsov, B. Skaali, T. Tveter are with Department of Physics, University of Oslo, Norway

M. Ploskon is with the Institut für Kernphysik, University of Frankfurt, Frankfurt am Main, Germany

H. Appelshäuser, A. Arend, H. Büsching, and T. Rascanu are with the Institut für Kernphysik, Max-von-Laue-Str. 1, Goethe-Universität Frankfurt, 60438 Frankfurt am Main, Germany.

B. Becker, and Z. Vilakazi are with the iThemba LABS, University of Cape Town,PO Box 722, Somerset West 7129, South Africa.

S. Chattopadhyay and I. Das are with the High Energy Physics Division, Saha Institute of Nuclear Physics, 1/AF, Bidhan Nagar, Kolkatta 700 064, India.

J. Cleymans and R. Fearick are with the Department of Physics, University of Cape Town, Private bag X3, Rondebosch 7700, South Africa.

M. Ploskon is with the Physics Division, Lawrence Berkeley National Laboratory,1 Cyclotron Road, 50R4049, Berkeley, CA 94720-8153, United States.

A. A. Masoodi is with the Department of Physics, Aligarh Muslim University, Aligarh 202001, U.P., India.

P. T. Hille is with the Department of Physics, Yale University, New Haven, CT 06520, United States.

I. Kisel is with the GSI Helmholtzzentrum für Schwerionenforschung GmbH, Planckstr. 1, 64291 Darmstadt, Germany.

I. Introduction

THE ALICE High-Level Trigger [1], [2] currently processes proton-proton collisions at 1kHz. It is scheduled that towards the end of 2010 the first heavy ion collisions will be processed. The increased complexity of heavy ion collisions makes the tracking computationally more expensive (figures 1 and 2 show complexity of both types of events to be reconstructed in real-time).

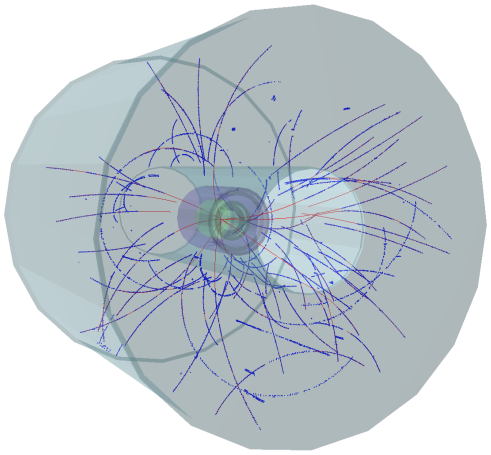

Fig. 1. Proton-proton event in the ALICE TPC detector. Real data reconstructed by HLT, run 00010480 (2009).

In recent years, the increase in processor clock speed has stagnated. Instead a new trend to multi- and many-core chips has developed. It is evident that, for raw computation power, the best approach is a big set of small and simple cores as it has been realized within graphics cards for many years now. While at first they could only be used for very special problems using algorithms that had to be developed with a particular architecture in mind, there are frameworks available today to run general purpose code written in high level languages on GPUs with little changes.

The Cellular Automaton tracking algorithm used in the ALICE HLT for online reconstruction has been developed with multi-core support in mind. All steps of the algorithm can

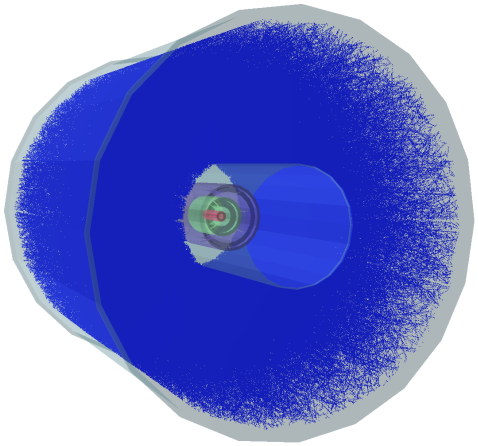

Fig. 2. Simulated heavy ion event in the ALICE TPC detector.

easily be spread over many independent processors. Proton-Proton collisions resulting in up to several hundreds of clusters can already be handled by the HLT compute farm. Primarily targeted at processing upcoming Pb-Pb events, with more than 20.000 tracks and several million clusters in the worst case, the tracker was adjusted to run on GPUs. A framework being able to run the same source code on a CPU as well as a GPU was developed, where the same source files are included in wrappers for both processor types. This assures that code maintainability does not suffer.

II. HLT Reconstruction Scheme

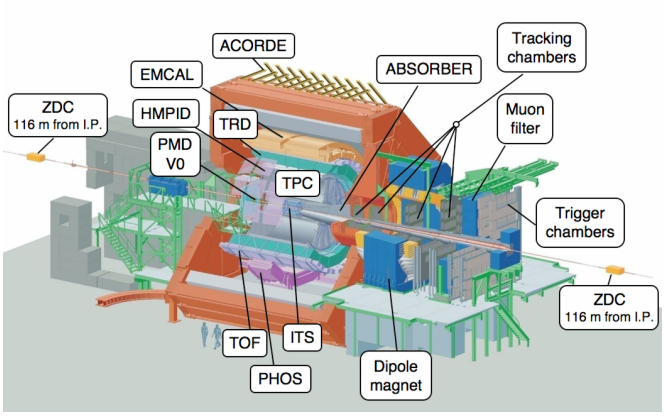

Fig. 3. The ALICE spectrometer at LHC.

The Time Projection Chamber (TPC) detector, which is shown in figures 1, 2, 3, 4, is the main tracking detector of the ALICE experiment and computing the TPC tracks is a major part of event reconstruction. The TPC detector consist of two cylindrical ($\pm$Z) volumes placed along the beam; either Z-volume is split into 18 readout sectors. The detector measures track positions on 159 rows as it is shown in figure 4.

The overall on-line reconstruction scheme is presented in figure 5. It starts with the TPC cluster finder, which finds the hits by identifying localized clusters and computing their centre of gravity. These reconstructed hits are sent to the sector tracker which reconstructs the tracks in each TPC sector individually. Then the sector tracks are merged by the track merger algorithm, and later updated with the measurements from the ITS detector. The reconstruction of the event's vertex and the physical triggers are run at the end of the reconstruction tree structure. Typically every processing stage reduces the size of the event data. This scheme processes data as early as possible avoiding any unnecessary copy steps and uses all available data locality and parallelisation.

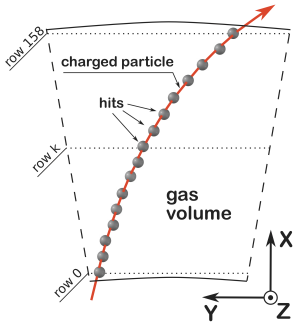

Fig. 4. Geometry of a TPC sector.

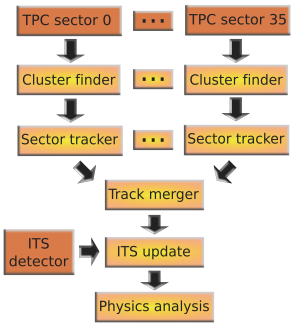

Fig. 5. HLT reconstruction scheme.

The core of the event reconstruction happens in the TPC sector tracker, which creates the tracks from the measurements. It is the only component which processes the TPC hits, the higher level components operate on the reconstructed sector tracks.

III. HLT Tracker Algorithm

An event coming from the detector only contains information about the spatial position of the hits, but no information about particles which caused the hits. The task of the track finder is to group the hits in such a way that they form the original particle trajectories.

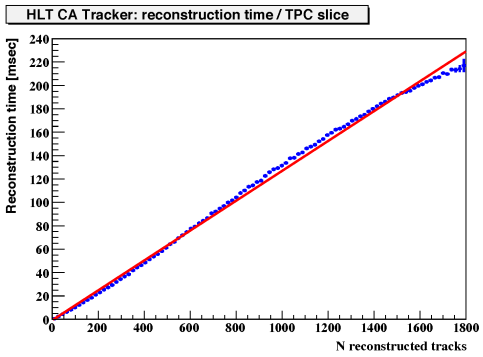

Fig. 6. TPC reconstruction time on CPU.

This is a combinatorial pattern recognition problem. Since the potential number of hit combinations is enormous, [1] there is no exact solution of the problem. Therefore heuristic methods are applied. Due to the rapid growth of the number of combinations with increase of the input size the key issue

[1]For example, given n tracks producing hits in each of 159 TPC rows, the number of possible hit combinations to create a single track is equal to n^{159}.

is dependence of the reconstruction time on the number of tracks to be reconstructed. Figure 6 shows that the presented algorithm requires 130 us per track independent of the detector occupancy, thus the combinatorial part of the algorithm is built optimally.

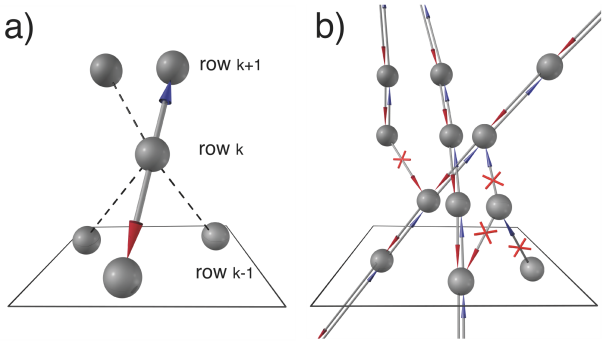

Fig. 7. a) Neighbours finder. b) Evolution step of the Cellular Automaton.

The tracking algorithm starts with a combinatorial search for track candidates (tracklets), which is based on the Cellular Automaton method [3]. Local parts of trajectories are created from geometrically nearby hits, thus eliminating unphysical hit combinations at the local level. The combinatorial processing composes the following two steps:

- 1. Neighbour finder: For each hit at a row k the best pair of neighbouring hits from rows k+1 and k-1 is found, as it is shown in fig. 7a. The neighbour selection criteria requires the hit and its two best neighbours to form a straight line. The links to the best two neighbours are stored. Once the best pair of neighbours is found for each hit, the step is completed.
- 2. Evolution step: Reciprocal links are determined and saved, all the other links are removed (see fig. 7b).

Every saved one-to-one link defines a part of the trajectory between the two neighbouring hits. Chains of consecutive one-to-one links define the tracklets. One can see from fig. 7b that each hit can belong to only one tracklet because of the strong evolution criteria. This uncommon approach is possible due to the abundance of hits on every TPC track. Such a strong selection of tracklets results in a linear dependence of the processing time on the number of track candidates. When the tracklets are created, the sequential part of the reconstruction starts, implementing the following two steps:

- 3. Tracklet construction: The tracklets are created by following the hit-to-hit links as it is described above. The geometrical trajectories are fit using a Kalman Filter, with a χ^2 quality check. Each tracklet is extended in order to collect hits being close to its trajectory.
- 4. Tracklet selection: Some of the track candidates can have intersected parts. In this case the longest track is saved, the shortest removed. A final quality check is applied to the reconstructed tracks, including a cut on the minimal number of hits and a cut for low momentum.

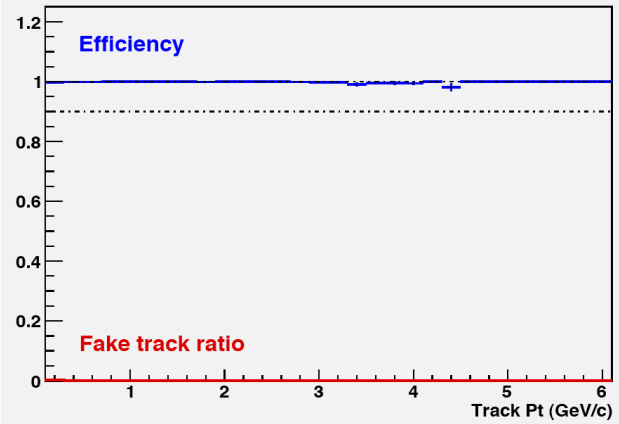

Fig. 8. Reconstruction performance for proton-proton collisions at 14TeV.

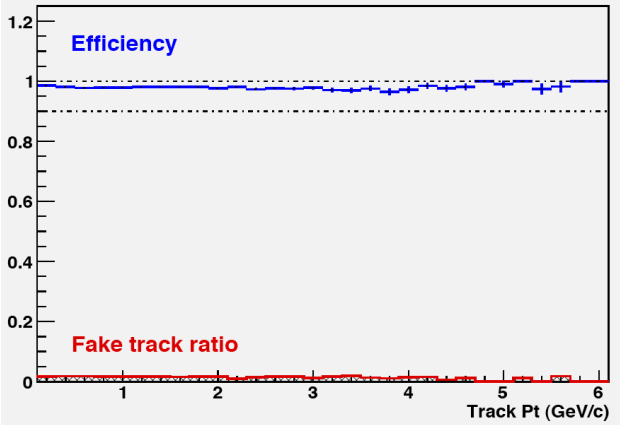

Fig. 9. Reconstruction performance for central heavy ion collisions at 5TeV.

IV. Tracker Efficiency

The performance of the HLT track finder of 99.9 % for proton-proton events and 98.5 % for central Pb-Pb collisions has been verified on simulated events. Corresponding efficiency plots are shown on figures 8 and 9. In addition to the high efficiency, the real-time reconstruction is an order of magnitude faster than the off-line algorithm used as reference. The described algorithm has the advantage of a high degree of locality and allows for massively parallel implementation as outlined in the following section.

There are many parts of the event reconstruction which are running after the tracker, in particular the primary vertex finder and the V0 finder. As previously noted, the HLT reconstruction was not only tested on simulated data, but it is running on the real data since 2009 [4], [5], [6]. A snapshot of one of the first ALICE proton-proton events obtained by the HLT is shown in the figure 10.

V. Tracking on GPU Hardware

During the tracking there are 5 steps with a non-negligible requirement of computation time: Initialization, Neighbour Finding, Tracklet Construction, Tracklet Selection and Tracklet Output.

Of these, the Tracklet Construction contains all the mathematics and most non trivial calculations, while consuming

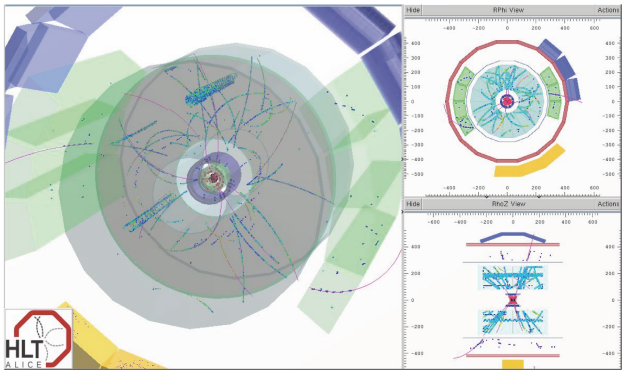

Fig. 10. The first proton-proton event, obtained by the ALICE High Level Trigger.

50% of the time. It is therefore both the part best suited for running on a GPU and the part with the best opportunity for optimisation. Currently the hardware deployed in the HLT are GT200 chips from NVIDIA.

All above-mentioned steps have been ported to CUDA with the most effort put into the Tracklet Construction. The GPU tracker is implemented in a way, that the main tracking algorithm is contained in common source code for the CPU and GPU versions. Only two specialised wrappers are used for each architecture respectively. The common source file is included in both wrappers and processed by the CPU and GPU compiler. This way changes to the algorithm have to be applied only once.

Since current GPU chips show good performance only for single precision calculations, the whole tracker code uses single precision only. An adaptation of the Kalman filter assures numerical stability to the algorithm in single precision [7], [8].

To efficiently run the Tracklet Construction on the GPU, a basic understanding of the GPU's architecture is needed. The GT200 chip consists of 30 independent multiprocessors with 8 Arithmetic-Logic Units (ALU) each. Each multiprocessor can handle a vast number of threads in parallel. Depending on the algorithm, one should have about 256 concurrent threads running on each multiprocessor for fully exploit the GPU. The threads running on a multiprocessor are organized in warps of 32 threads each. All threads in one warp can only execute one particular common instruction. If different threads are to execute different instructions, for example due to branching in the code, these operations have to be serialized.

VI. Optimizations for GPU Tracking

The GPU implementation of the Tracklet Construction has each tracklet processed by a different thread. The problem arising here is caused by different lengths of the tracklets. As a matter of fact all threads within one warp have to wait for the one thread processing the longest tracklet, even if their current task is already finished. This resulted in the GPU Utilization staying below 20% for the first implementation (See Fig. 11).

The problem was solved by introducing a custom scheduler. One thread only extrapolates a tracklet for a constant amount

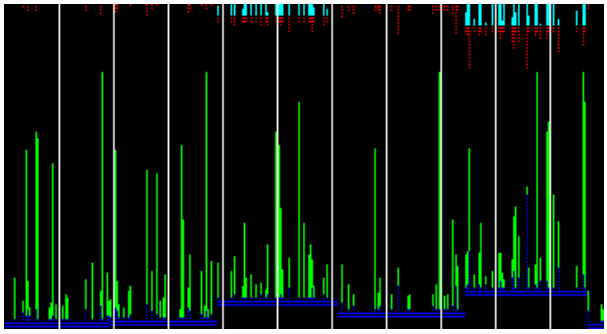

Fig. 11. GPU utilization without scheduling during Tracklet Construction [1]

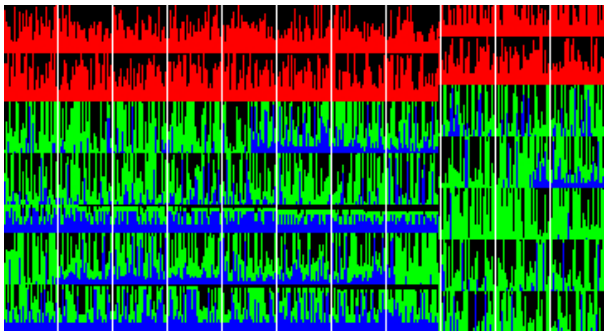

Fig. 12. GPU utilization with scheduling during Tracklet Construction [1] [2]

of rows. Afterwards all unfinished tracklets are redistributed among threads and even multiprocessors. Further, some prefiltering was introduced to remove very short tracklets from the queue before even starting the extrapolation. The scheduler works efficiently when the tracker processes multiple slices in parallel. This ensures that there are always enough threads available for scheduling. By applying these changes the GPU utilization raised to almost 70% (Fig. 12).

Additionally the memory layout was changed in such a way, that threads which are executed in parallel access consecutive memory addresses. This is done by interleaving the data structures for different threads. The GPU memory controller can coalesce accesses from different threads into one single memory transaction.

Apart from the Tracklet Construction also the Neighbour Finding and Tracklet Selection was ported onto the GPU. The performance of the Neighbours Finder could be significantly improved by caching intermediate data in the fast shared memory of the NVIDIA GPU. However, the current memory is insufficient for all intermediate data. Fortunately the memory size will be increased with the next GPU generation possibly resulting in a further speedup.

Running the Tracklet Selection on the GPU is necessary since even though the Tracklet Selection is slower as compared to the CPU version, it greatly decreases the output of the pipeline and thus the amount of data that is transferred back

[1] White borders seperate threads of one warp. Colors stand for: black: idling, colored: different states during Tracklet Construction.

[2] The three rightmost threads belong to different multiprocessors and are scheduled separately.

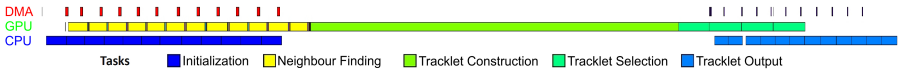

Fig. 13. Asynchronous event processing

to the host. Contrary to all these tasks, the Initialization and Output steps do not involve computation, but instead have lots of random memory reads requiring most data only once. This is not well suited for a GPU, especially considering the additional data transfer required, but can benefit from big and advanced caches of state of the art CPUs and therefore should stay on the CPU.

Keeping the GPU cores operating at full capacity is the main objective. Since multiple slices are handled simultaneously anyway, to allow efficient scheduling the steps are pipelined asynchronously using both, CPU and GPU, while data is transferred via DMA. This way, after having initialized the tracker data structures for the first slice, the CPU can immediately process the next slice while the GPU starts tracking the first one, as can be seen in Fig. 13.

VII. Integration in the HLT Framework

The general Tracker interface that was developed allows for using both, the CPU and GPU tracker, within different frameworks. The first and most important one to mention is the HLT framework. In addition it is possible to run the GPU tracker from AliRoot which is the ALICE offline framework for data analysis, event reconstruction, and simulation. Both build systems needed to be altered in order to invoke the CUDA compilation toolkit. In addition a standalone version was developed and is maintained for debugging and profiling reasons.

The HLT uses an AFS file system to store the libraries for all nodes on a common storage. The CUDA runtime library does not support late binding and therefore libraries referencing the CUDA runtime can only be used on nodes with the CUDA framework installed. This is not the case on all cluster nodes. To account for this all GPU related code is separated from the rest of the framework in a separate library. This library offers a plain C interface for the creation and destruction of GPU tracker objects that can be used from the framework through the abstract interface.

Another problem that arose is that the CUDA context is thread local, while the HLT framework may uphold multiple threads, all allowed to access the GPU tracker object. Additionally the ROOT C-interpreter (CINT) is unable to process CUDA files. However, this is required in order for the logging system to work because it is based on ROOT macros. This is solved by making CINT interprete a fake file stripped of all explicit GPU content.

VIII. Efficiency and Performance of the GPU Tracker

In the GPU Tracking algorithm the order in which tracks are reconstructed is not deterministic. Clusters are assigned to tracks according to several criteria. However, if none of them apply, clusters are assigned according to the first-come-first-serve principle, therefore the GPU Tracking algorithm is not completely deterministic. To test the GPU tracker data from Monte-Carlo simulations is used. A comparison of the simulated data shows that the output of the GPU tracker is equal to the output of the CPU tracker. It can be concluded that the GPU tracker is in no way inferior to the CPU version.

While porting the tracker, the memory model was slightly changed. This resulted in more locality for optimal usage of available GPU memory bandwidth and also had positive cache effects on the CPU. Because of this and other optimizations, that were applied the new tracker code performs better by a factor of two on modern CPUs (benchmarked using 3.2 GHz Intel Nehalem, 8 threads and data from monte-carlo-simulation) while the GPU version surpasses the processor by another factor of 3.3 for central lead lead collisions (Fig. 14).

Fig. 15 shows the performance of the CPU and GPU tracker on monte-carlo events of different size. Both trackers show a linear dependency on input size while the GPU tracker has a rather large offset. Obviously the performance benefit of the GPU tracker is the larger the larger the event size.

As a proof of concept an analysis of a different GPU tracker variant was created (pp-mode GPU tracker). However, the CPU tracker still performs better for pp events and the former GPU tracker performs better for heavy ion events.

IX. Summary

A fast on-line tracker has been developed for the ALICE High Level Trigger. The algorithm combines a Cellular Automaton method being used for a fast pattern recognition and the Kalman Filter method for fitting of found trajectories and for the final track selection.

The algorithm has proved its high performance on Monte-Carlo events (99.9% track finding efficiency for 14 TeV proton-proton events and 98.5% efficiency for central Pb-Pb events). The HLT tracker performs the on-line event reconstruction since the first collision runs in 2009.

An important feature of the developed algorithm is the ability to use GPU hardware accelerators. The algorithm was successfully ported to NVIDIA CUDA. For a better GPU utilisation several code optimisations were made, resulting in improved performance of both GPU and CPU trackers.

The GPU tracker significantly outperforms the CPU implementation for heavy ion events while exactly maintaining its efficiency. Both versions share a common source code greatly improving the maintainability.

The GPU tracker is incorporated in the ALICE High Level Trigger framework and will run online in 2010.

References

[1] ALICE collaboration. ALICE - Technical Proposal for A Large Ion Collider Experiment at the CERN LHC. *CERN/LHCC 1995-71*, 1995.

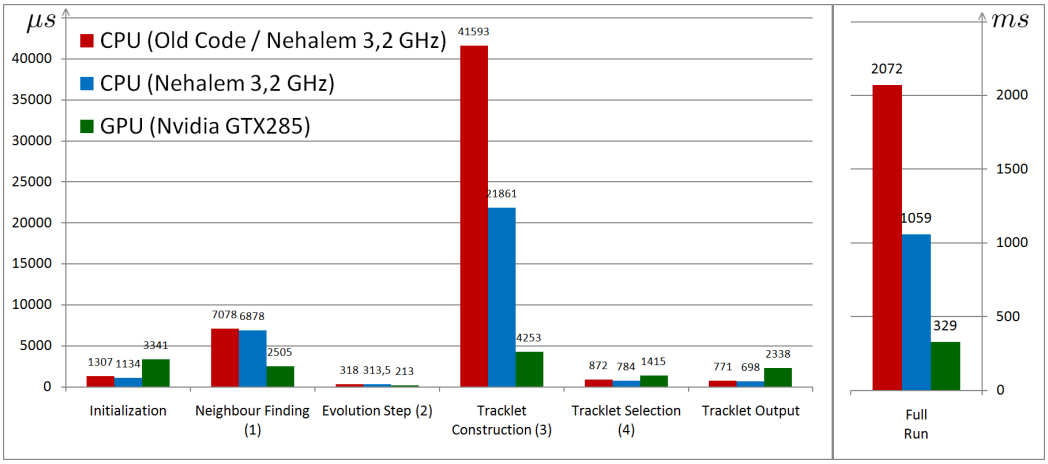

Fig. 14. GPU Tracker performance (central Pb-Pb event)

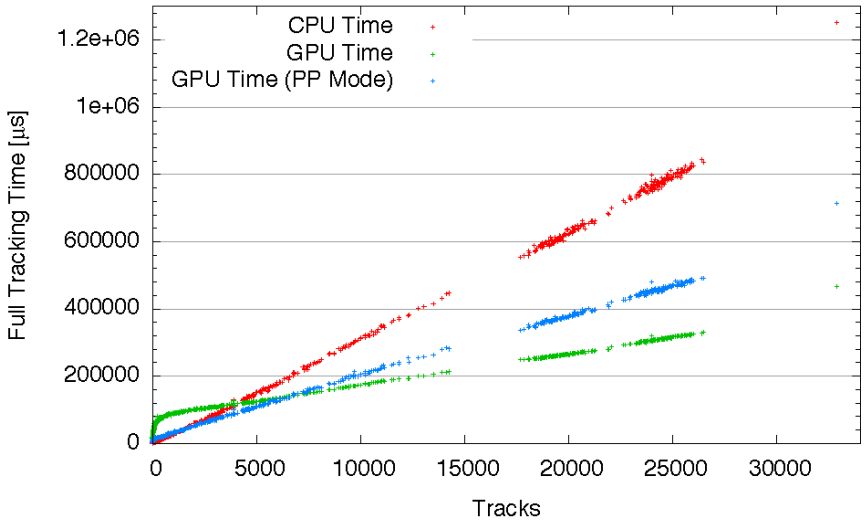

Fig. 15. CPU / GPU performance for different event sizes

[2] K. Aamodt et al. The ALICE Collaboration. The ALICE Experiment at the CERN LHC. *JINST 3 S08002*, 2008. doi: 10.1088/1748-0221/3/08/S08002.

[3] I. Kisel, Event reconstruction in the CBM experiment, Nucl. Instr. and Meth A 566 (2006), pp. 8588. Event reconstruction in the CBM experiment. *Nucl. Instr. and Meth A 566*, 2006, pp.85-88.

[4] K. Aamodt et al. (ALICE collaboration). First proton–proton collisions at the LHC as observed with the ALICE detector: measurement of the charged particle pseudorapidity density at sqrt(s) = 900 GeV. *Eur.Phys.J. C*, 65:111–125, 2010. arXiv: 1004.3034.

[5] J. Thäder et al. First Proton-Proton Collisions in the ALICE High-Level Trigger. *Proc.17th Real Time Conference, Lisbon 2010*

[6] M. Richter et al. Event Reconstruction Performance of the ALICE High Level Trigger for p+p collisions. *Proc.17th Real Time Conference, Lisbon 2010*

[7] S. Gorbunov, U. Kebschull, I. Kisel, V. Lindenstruth, and W.F.J. Mller. Fast SIMDized Kalman Filter based track Fit. Computer Physics Communications, 178:374 - 383, 2008

[8] S. Gorbunov. On-line reconstruction algorithms for the CBM and ALICE experiments. Dissertation Thesis, Frankfurth Institute for Advanced Studies, in preparation