ALICE TPC Online Tracking on GPU

D. Rohr[1], S. Gorbunov[1,2], and M. Kretz[1]
[1]Kirchhoff Institute for Physics, University of Heidelberg, Germany; [2]FIAS, Frankfurt, Germany

ALICE Online Tracker

For the ALICE High Level Trigger a fast tracking algorithm was developed by Sergey Gorbunov based on the Cellular Automaton method and the Kalman filter [1], that is currently installed in the HLT. For an efficient handling of upcoming lead-lead collisions in 2010 with a tremendous increase of clusters and tracks, possibilities for a better usage of parallelism and many core hardware were analyzed. The tracker itself was designed with parallel approaches in mind. A SIMD approach was realized by Matthias Kretz [2] in his diploma thesis and further the tracker was adopted to run on the NVIDIA CUDA framework [3].

Modern Many-Core Processors

With the race for higher and higher clock frequencies having reached an end, state of the art CPUs improve in efficiency and multi-core capabilities. Graphics cards have been designed using such approaches for many years now, and just recently started to offer support for execution of general purpose code in high level languages. The GT200b chip used consists of 30 cores, providing 8 ALUs and 16kb of fast shared memory per core. For reasons such as memory latencies there should be about 256 threads running on each core, to reach a good GPU utilization.

GPU Tracker

For fully exploiting the GPU's processing power many changes hat to be applied to the tracker. Major challenges hereby have been an efficient usage of the several heterogenous memory types on the GPU as well as a good overall utilization of all cores. In the later case different tracks lengths turned out to be a problem, that was solved by introducing a scheduler that dynamically distributes the workload among cores and threads. Utilization raised from 19% (Fig. 1) to 62% (Fig. 2). The implementation of simultaneous processing of multiple sectors allows the use of a processing pipeline where data processing and data moving to- and from the GPU overlap.

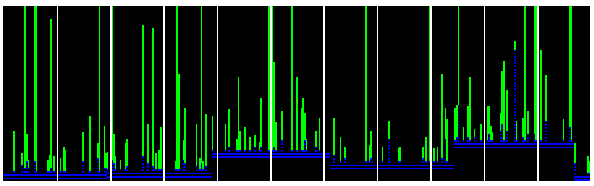
Figure 1: Tracklet Constructor without scheduling [1]

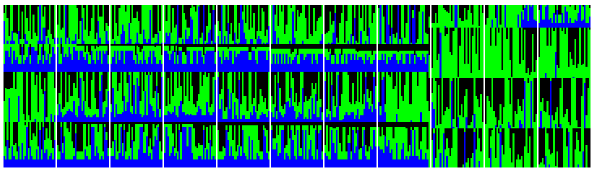
Figure 2: Tracklet Constructor with dynamic scheduling [1]

Performance

Out of the several steps involved in the tracking algorithm three have a non negligible runtime. Fig. 3 shows a performance comparison of the GPU tracker with the CPU version running on an Intel Nehalem 3.2 GHz. It demonstrates that performance for two of these three steps improved significantly. Although there is some drawback during initialization and output of the tracks, caused by GPU-Host synchronization and PCIe transfer, a complete tracking run using simulated data of a central lead-lead event was accelerated by a factor of 3.3.

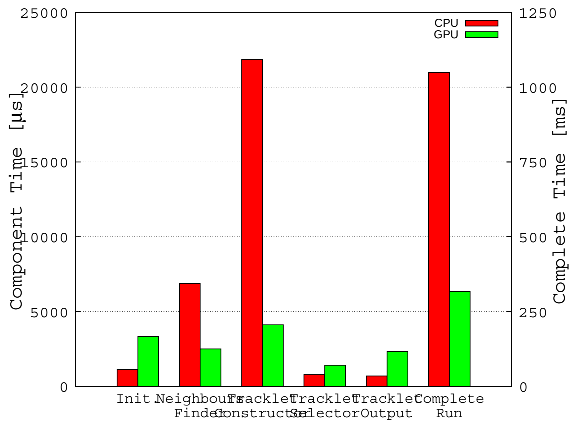

Figure 3: GPU Tracker Performance

References

[1] S. Gorbunov, Matthias Kretz, David Rohr "Fast Cellular Automaton tracker for the ALICE High Level Trigger", GSI Scientific Report 2009

[2] M. Kretz, "Efficient Use of Multi- and Many-Core Systems With Vectorization and Multithreading"

[3] NVIDIA CUDA Reference Manual, http://developer.download.nvidia.com/compute/cuda/2_3/toolkit/docs/CUDA_Reference_Manual_2.3.pdf

[1]Colors stand for: black: idling, color: working., y-axis: time, x-axis: thread