ALICE HLT TPC Tracking on GPUs

I: Introduction

II: Integration

III: GPU Tracker Performance

IV: CPU / GPU Tracker Comparison

V: Global Tracking

David Rohr for the ALICE Collaboration

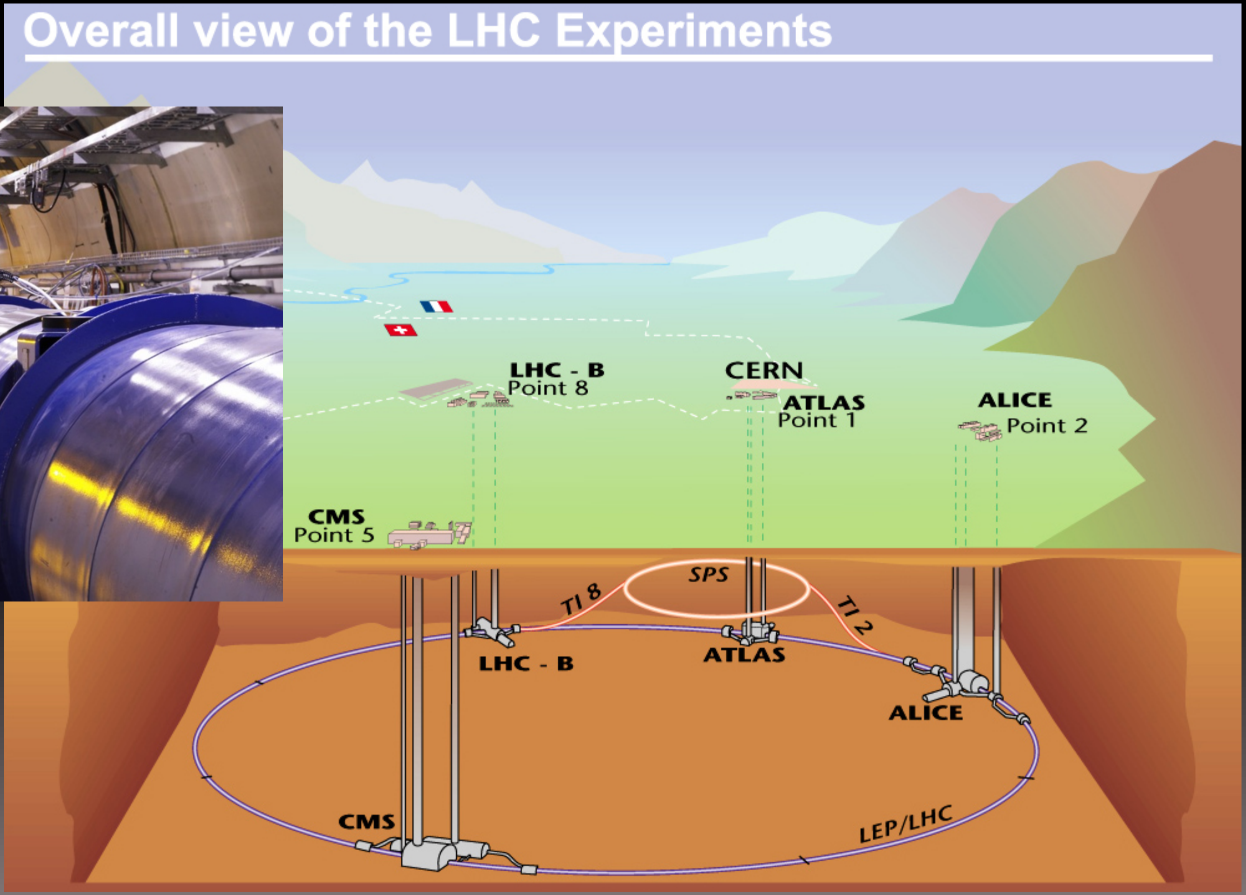
DESY - 15.4.2013

FIAS Frankfurt Institute for Advanced Studies

INTRODUCTION

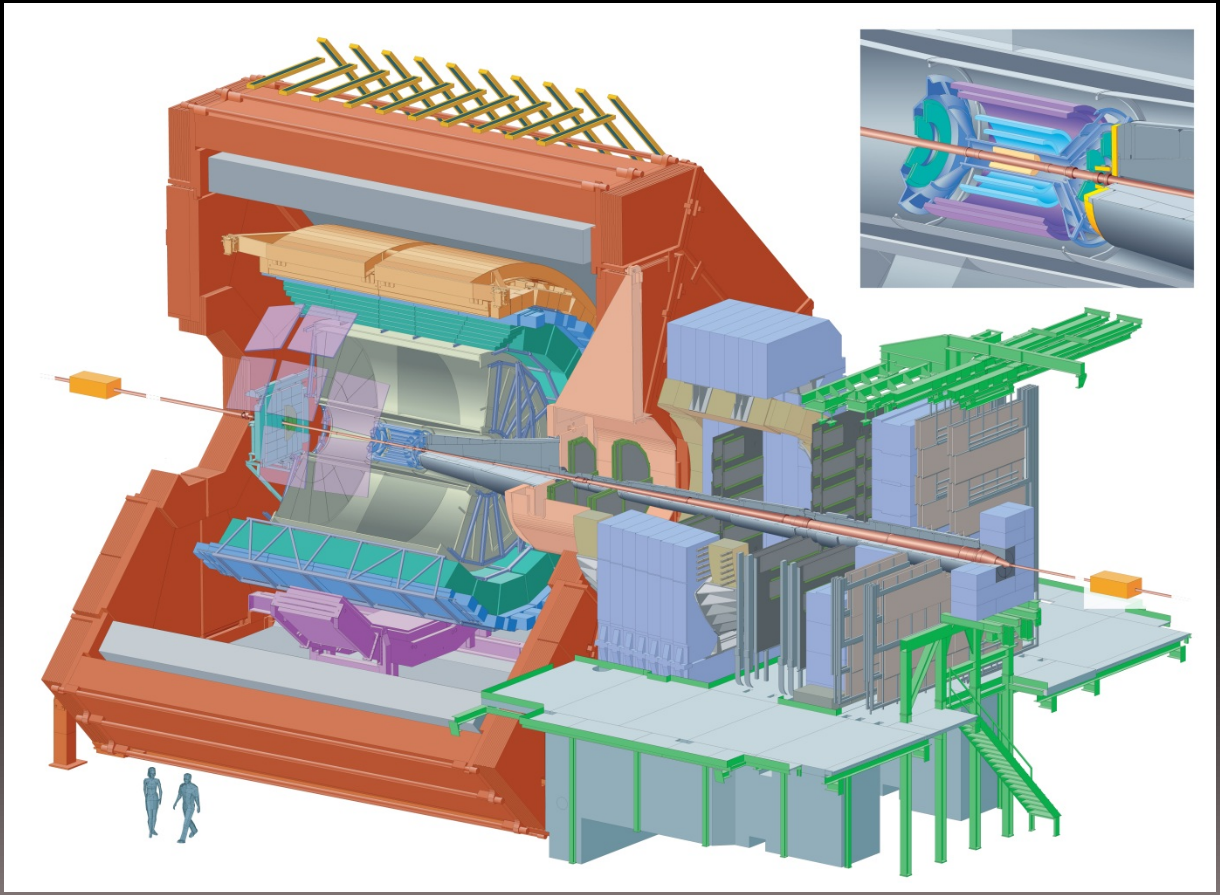
Introduction

- The Large Hadron Collider (LHC) at CERN
  - The Large Hadron Collider is today's largest particle accelerator colliding protons at an energy of up to 14 TeV and ions at more than 1 PeV in ist 27km tunnel.



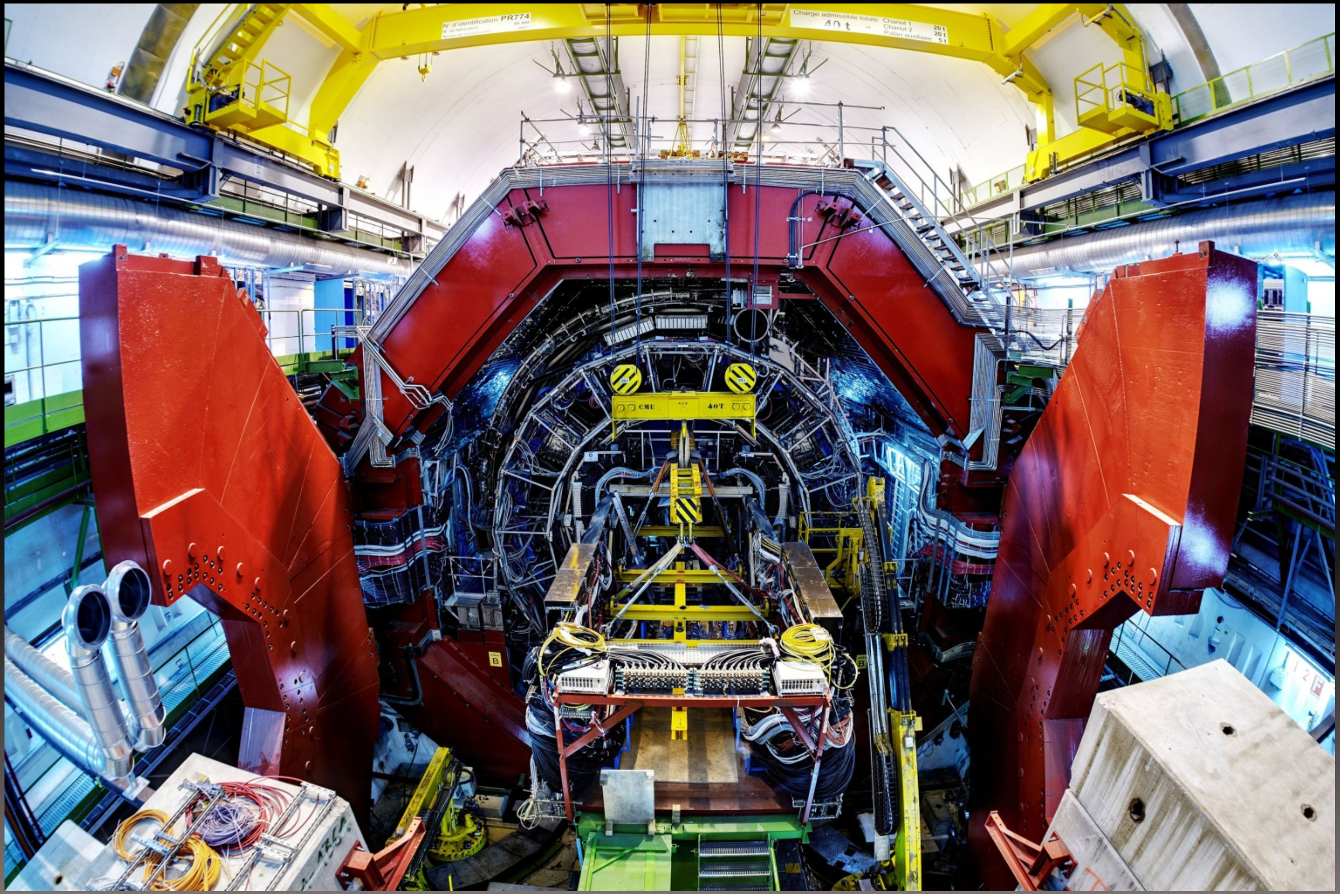
Introduction

- The ALICE detector
  - ALICE is one of the major four experiments of the Large Hadron Collider at CERN. It was specifically designed to study heavy ion collisions.



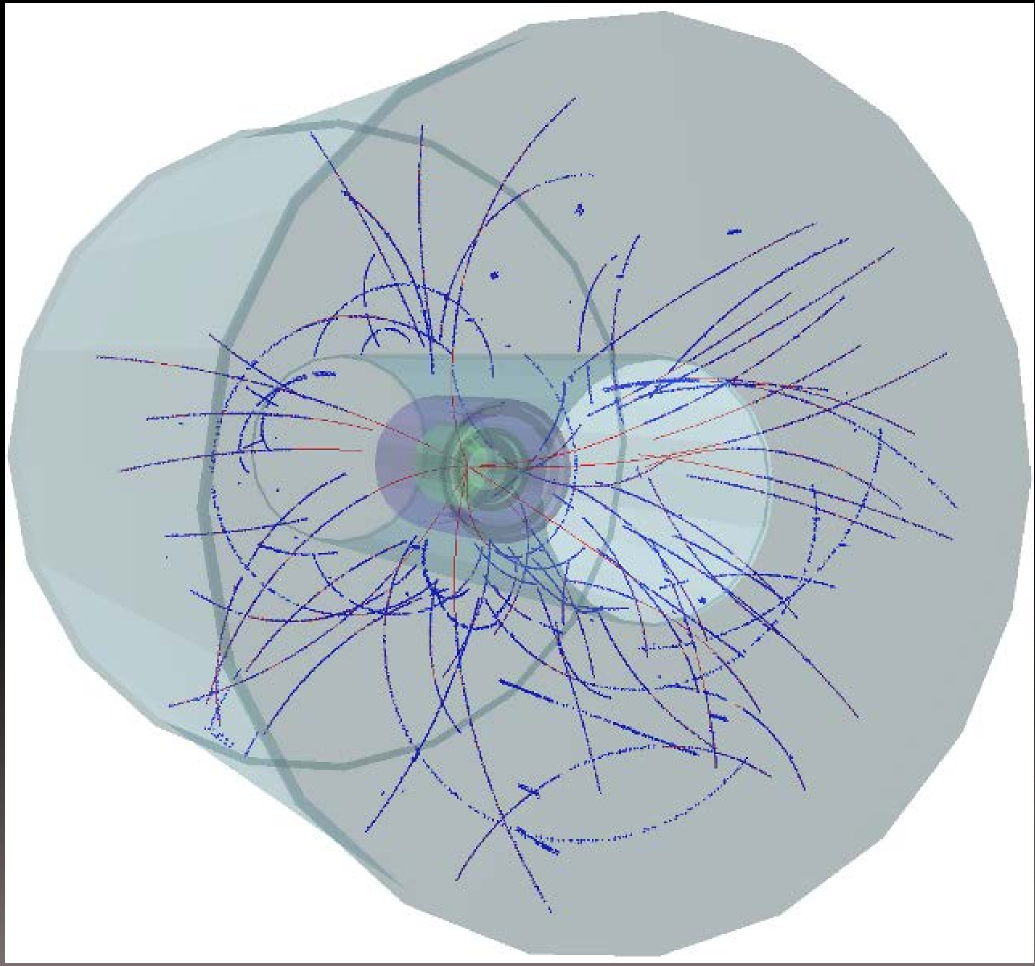
Introduction

- The ALICE detector
  - ALICE is one of the major four experiments of the Large Hadron Collider at CERN. It was specifically designed to study heavy ion collisions.



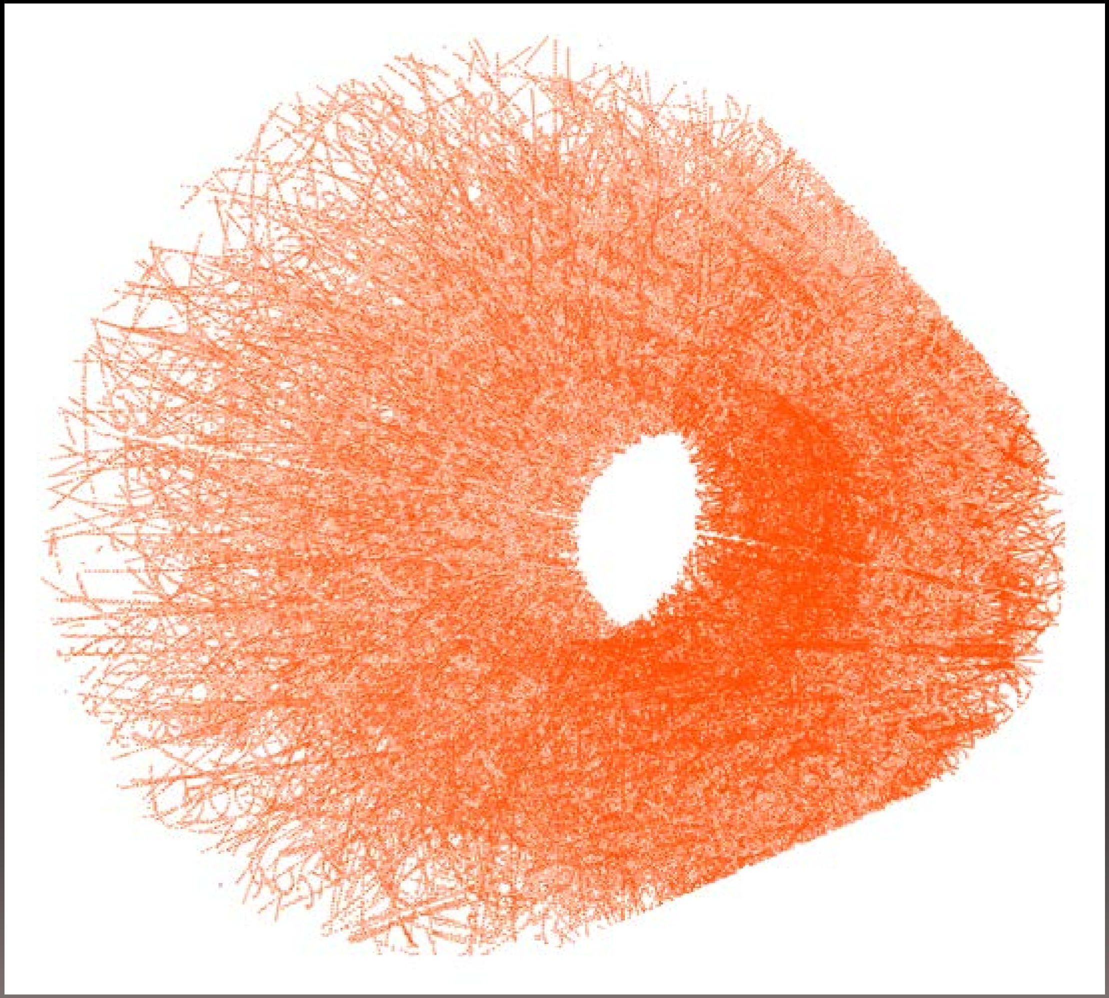
Introduction

- Proton event in TPC



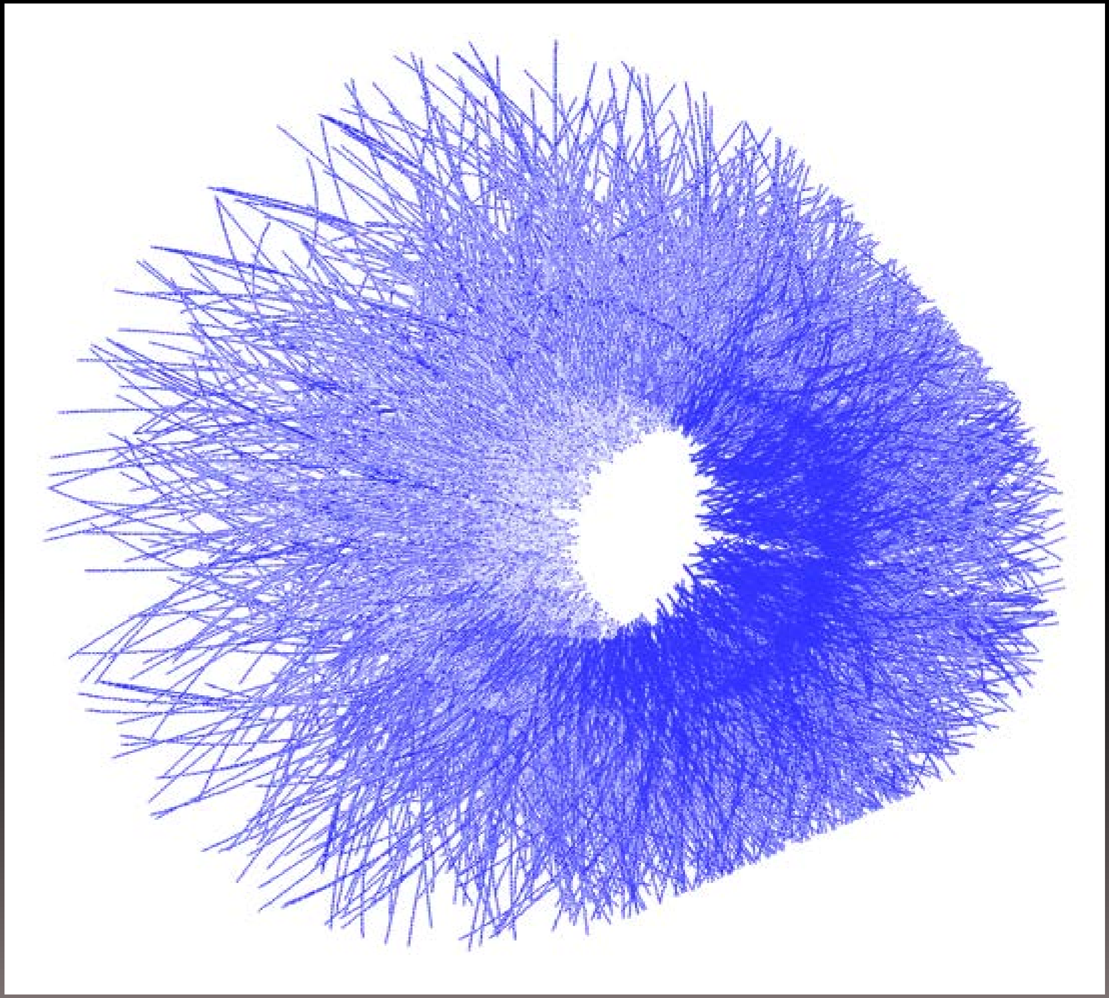
Introduction

- TPC clusters of heavy-ion event.



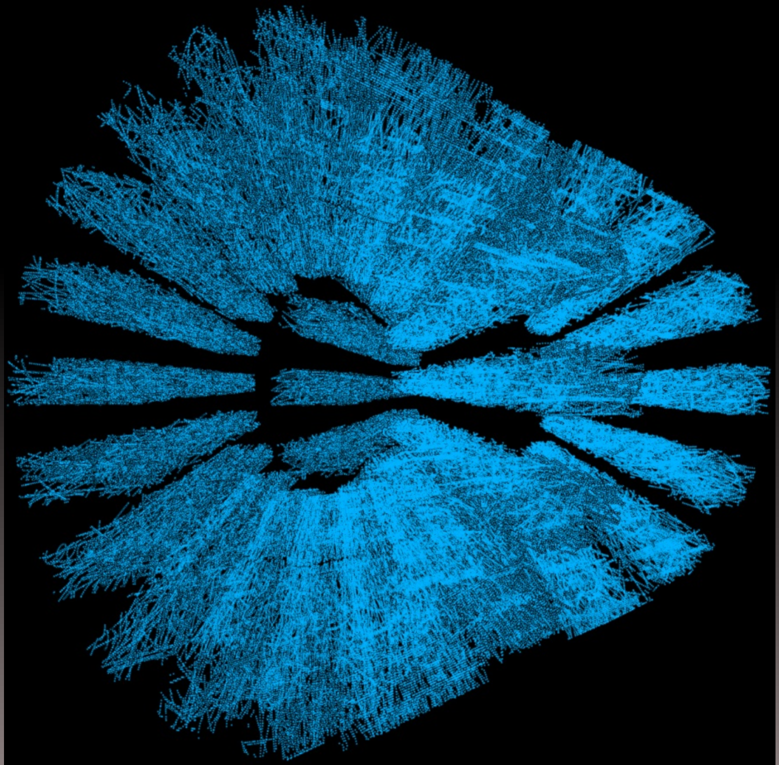
Introduction

- Tracks reconstructed from the clusters.



Introduction

- ALICE HLT tracker divides the TPC in slices and processes the slices individually.
- Track segments from all slices are merged later.



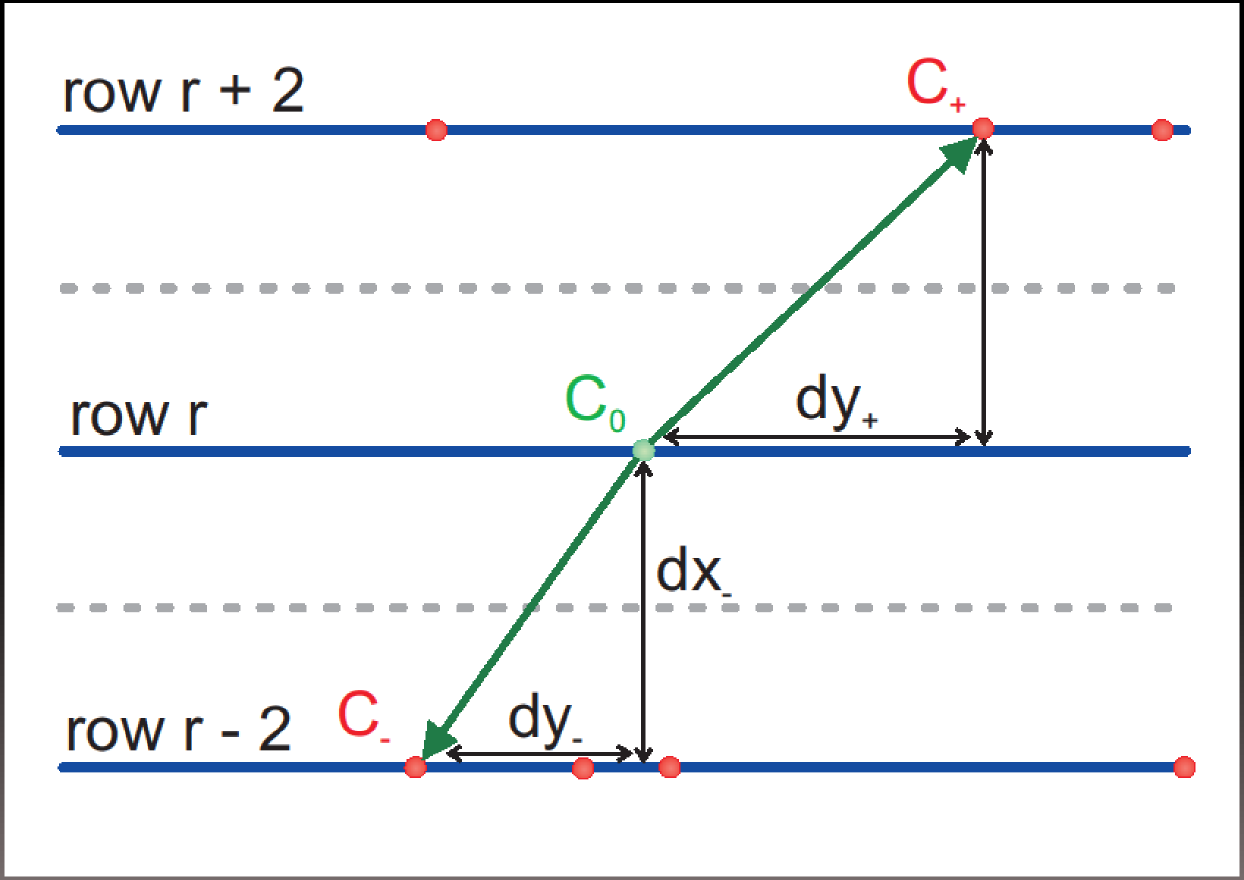
Introduction

Tracking algorithm

Category of task	Name of task	Description on task
	(Initialization)	
Combinatorial part (Cellular automation)	I: Neighbors finding	Construct seeds (Track candidates)
	II: Evolution	
Kalman filter part	III: Tracklet construction	Fit seed, extrapolate tracklet, find new clusters
	IV: Tracklet selection	Select good tracklets, sssign clusters to tracks
	(Tracklet output)	

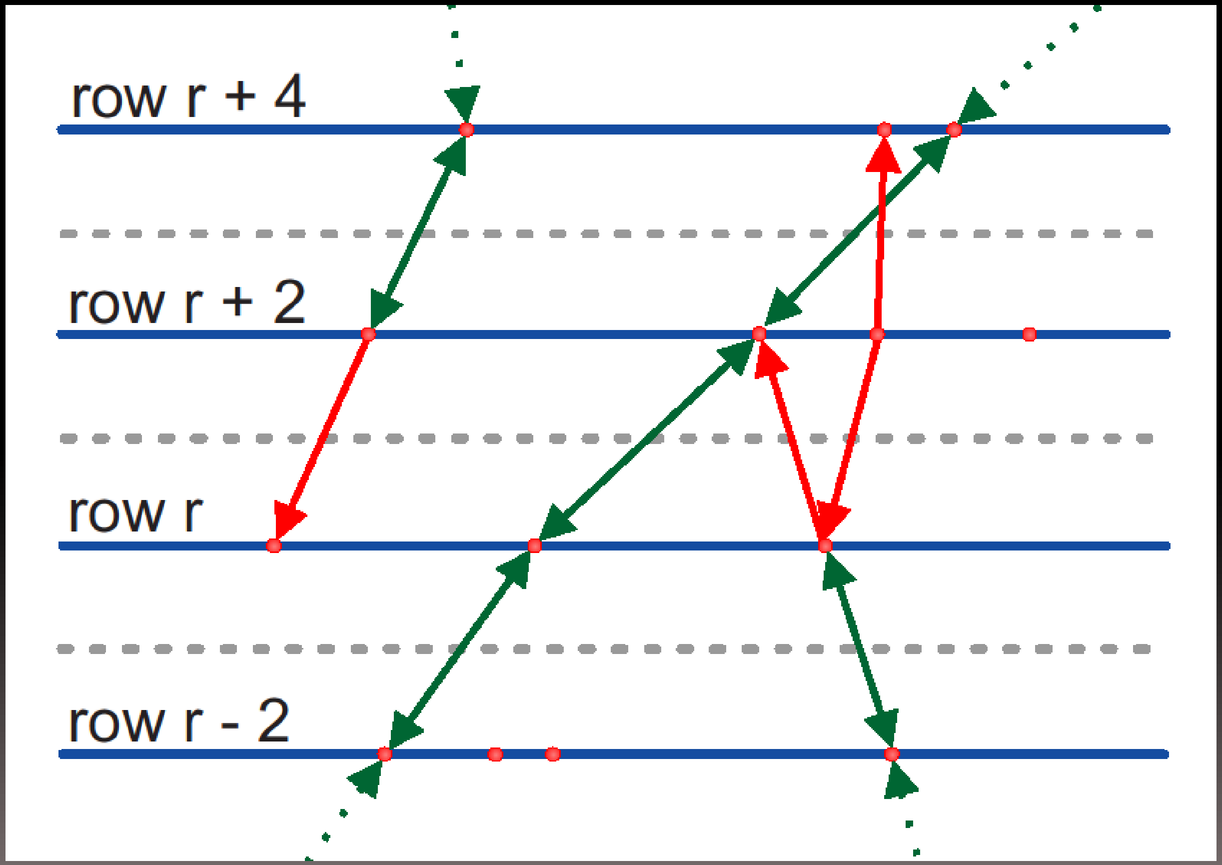
Introduction

Illustration of neighbors finding



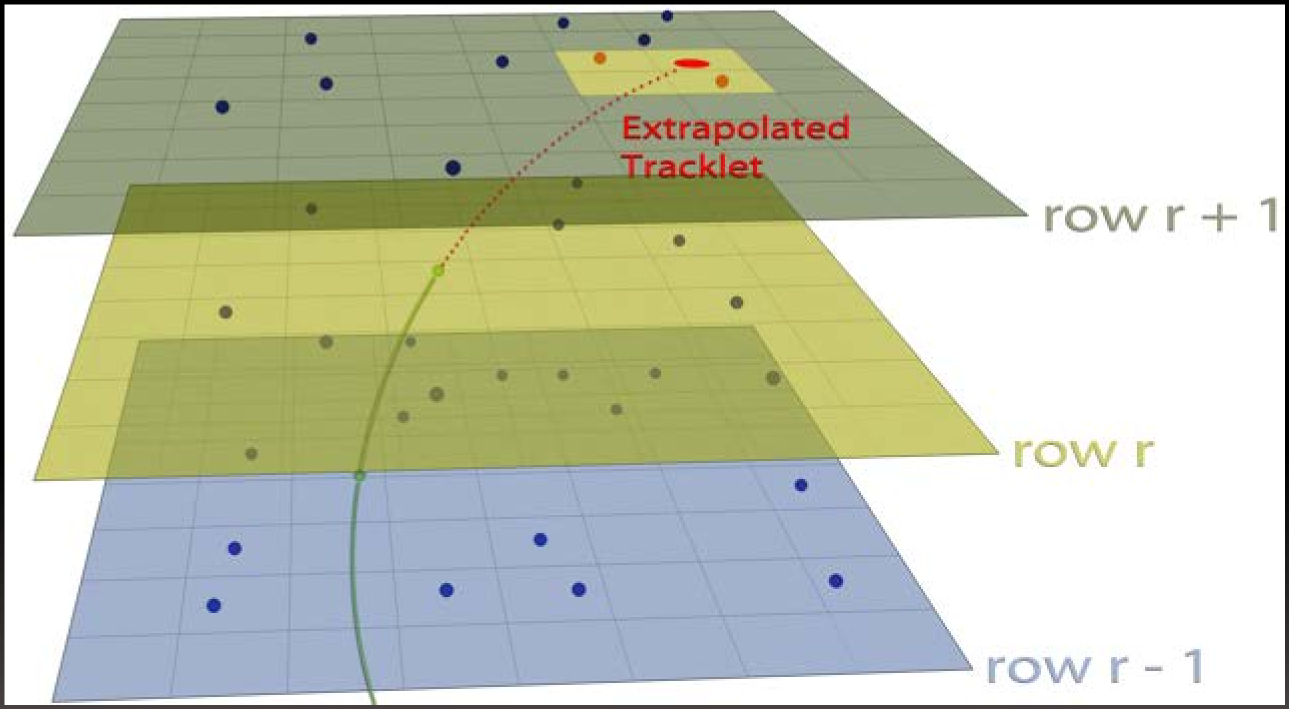
Introduction

Illustration of evolution step



Introduction

Illustration of tracklet construction

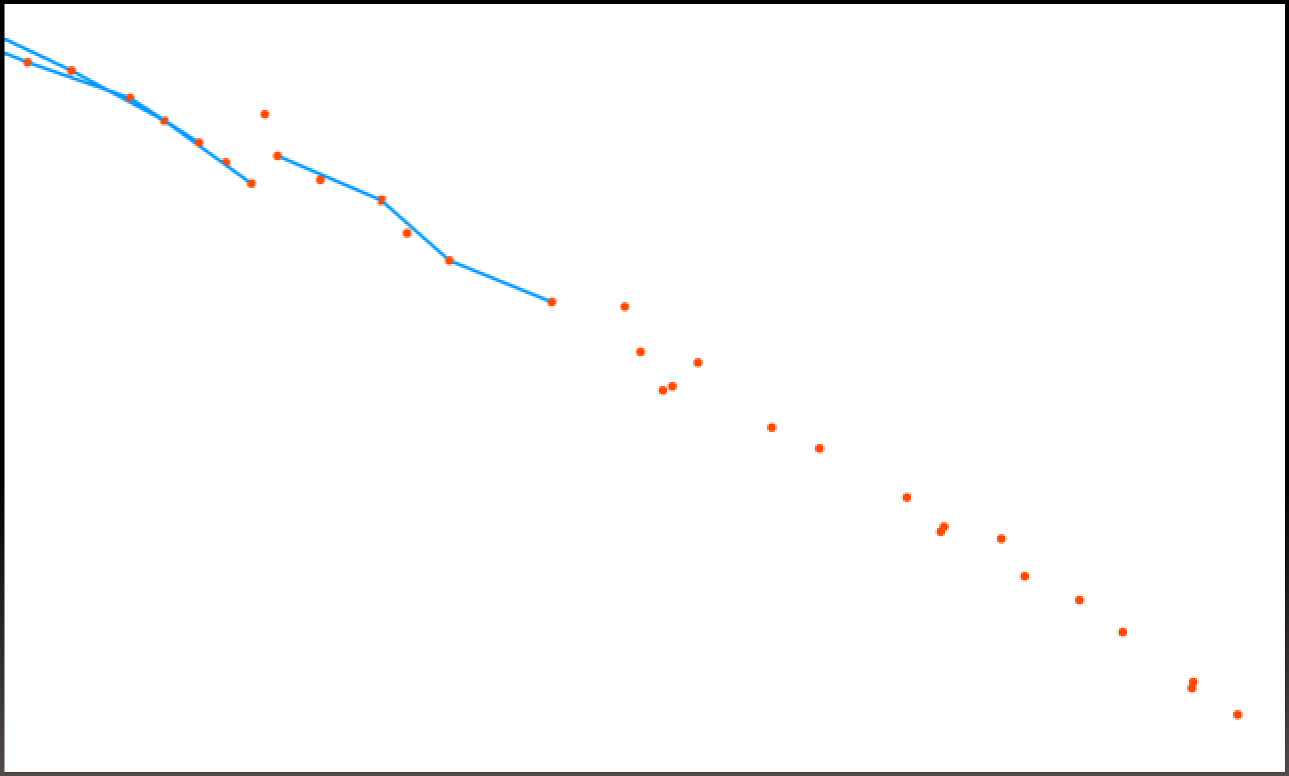


Green: Seed Red: Extrapolation

Clusters close to the extraplation point are searched

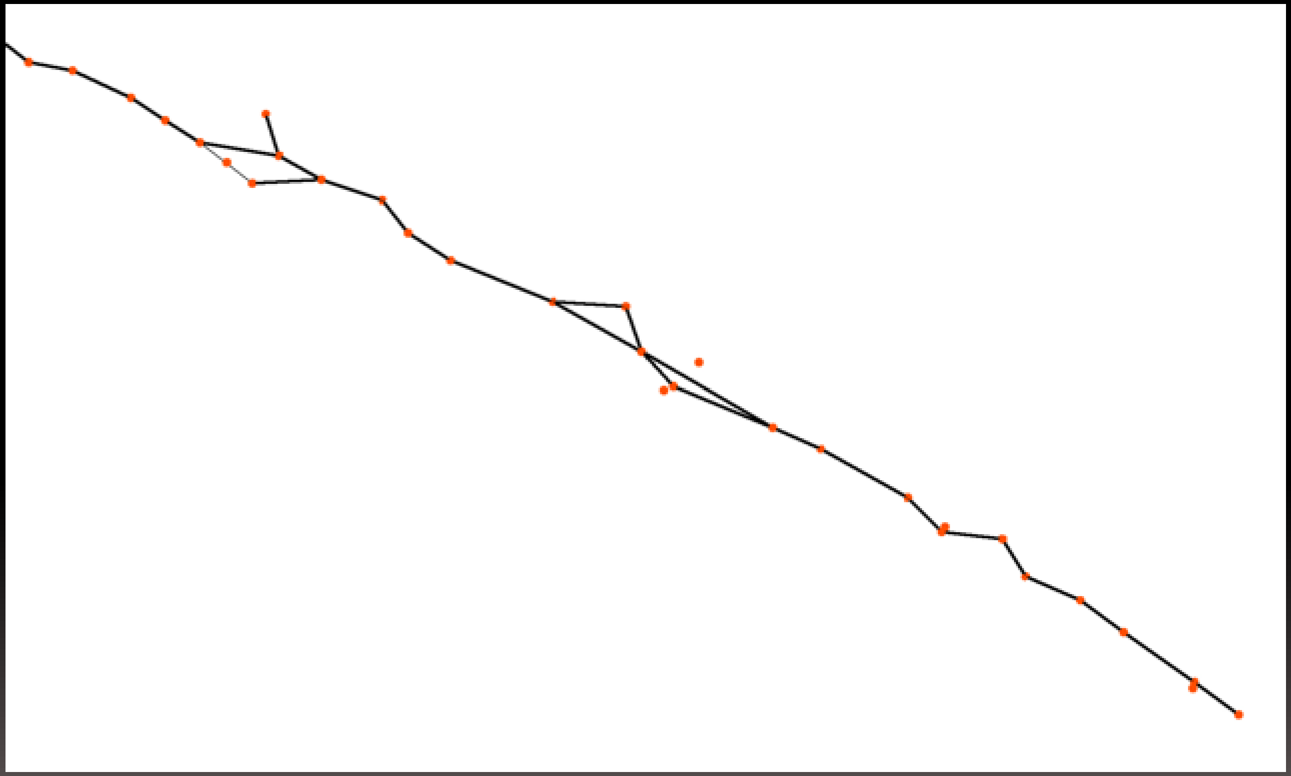
Introduction

Illustration of evolution step



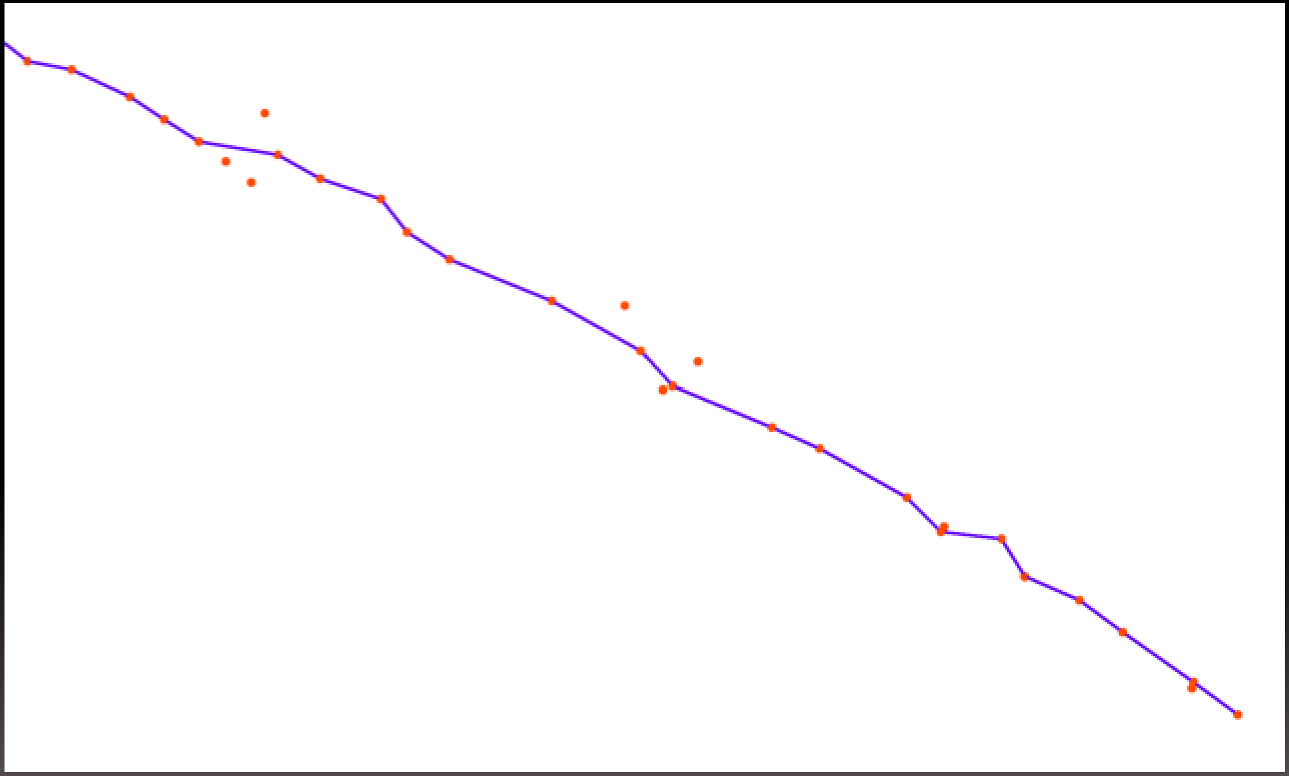
Introduction

Illustration of tracklet construction



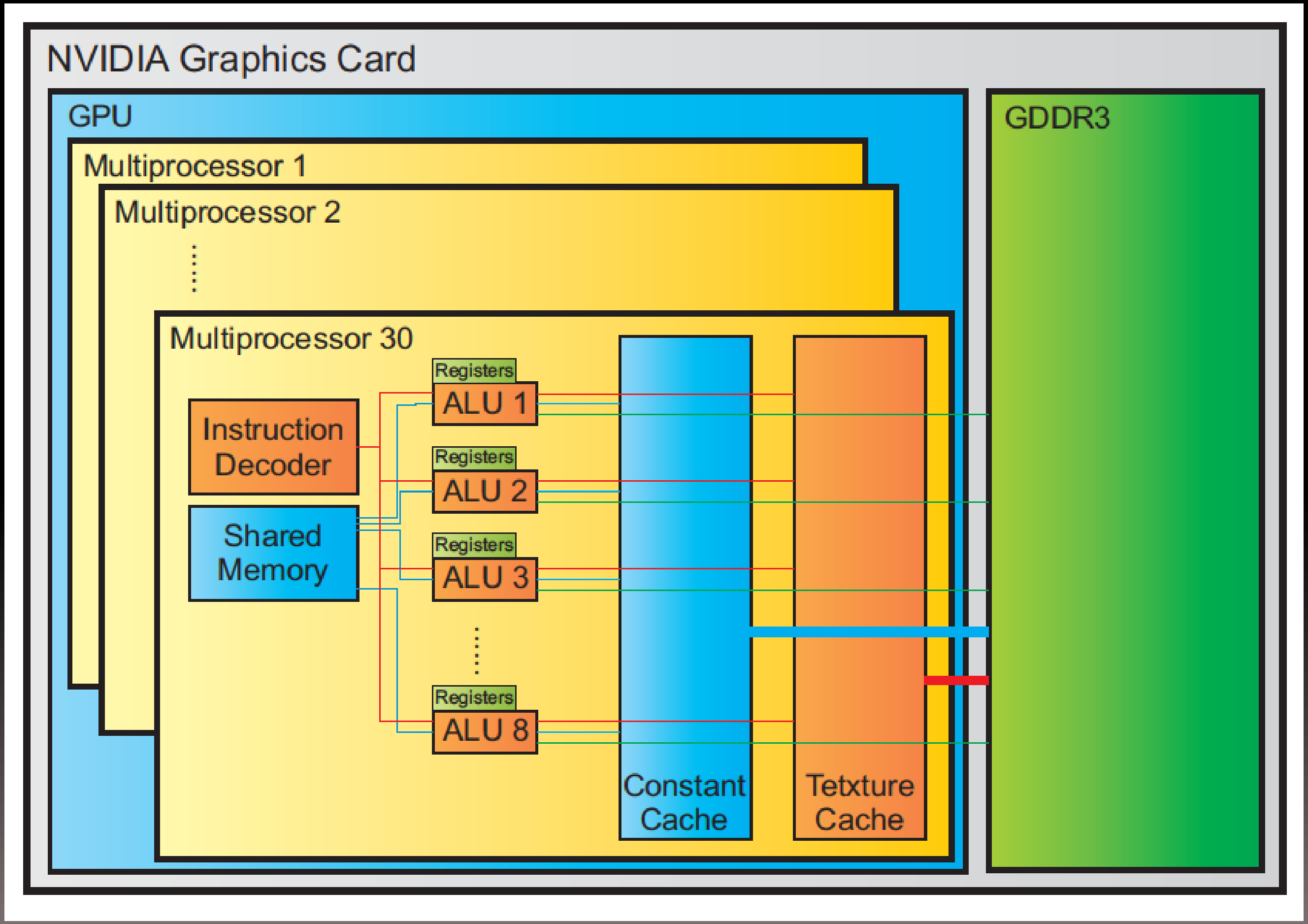
Introduction

Illustration of tracklet selection



Introduction

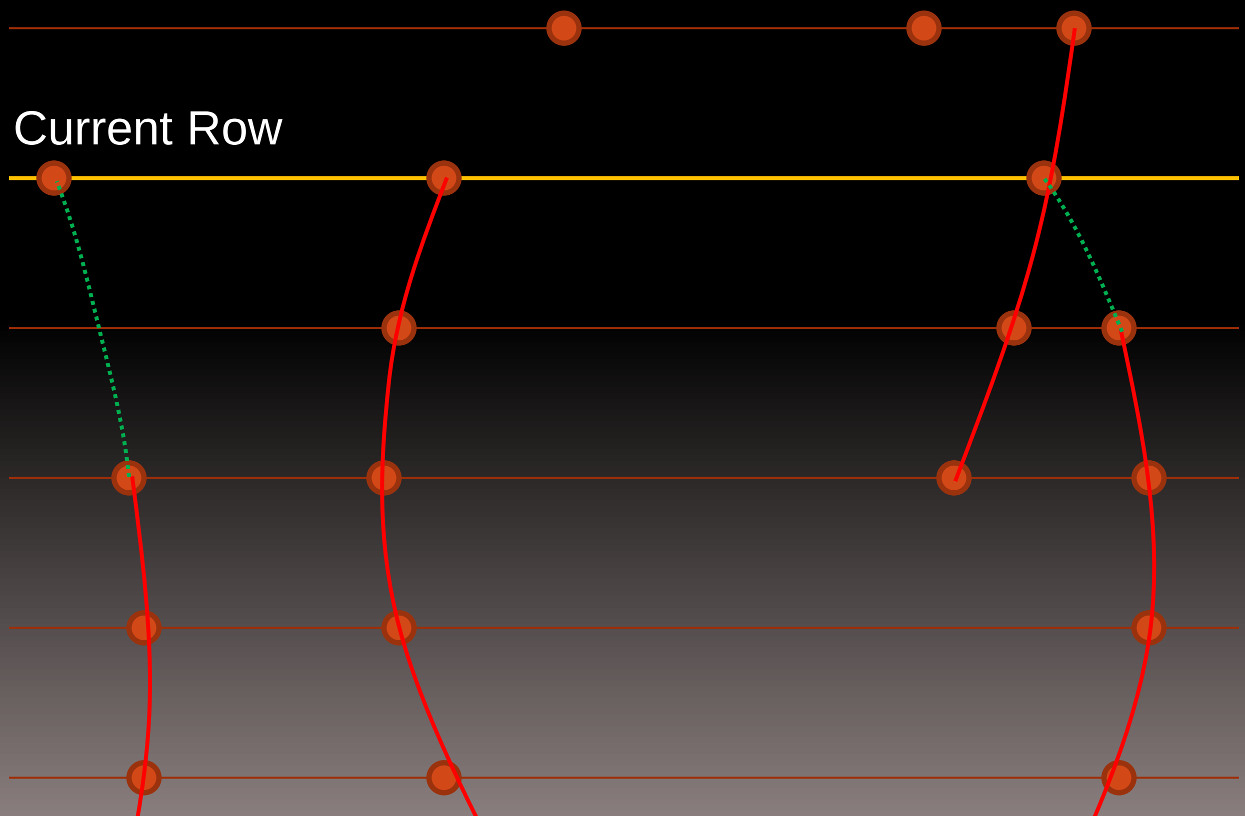
NVIDIA CUDA GPU



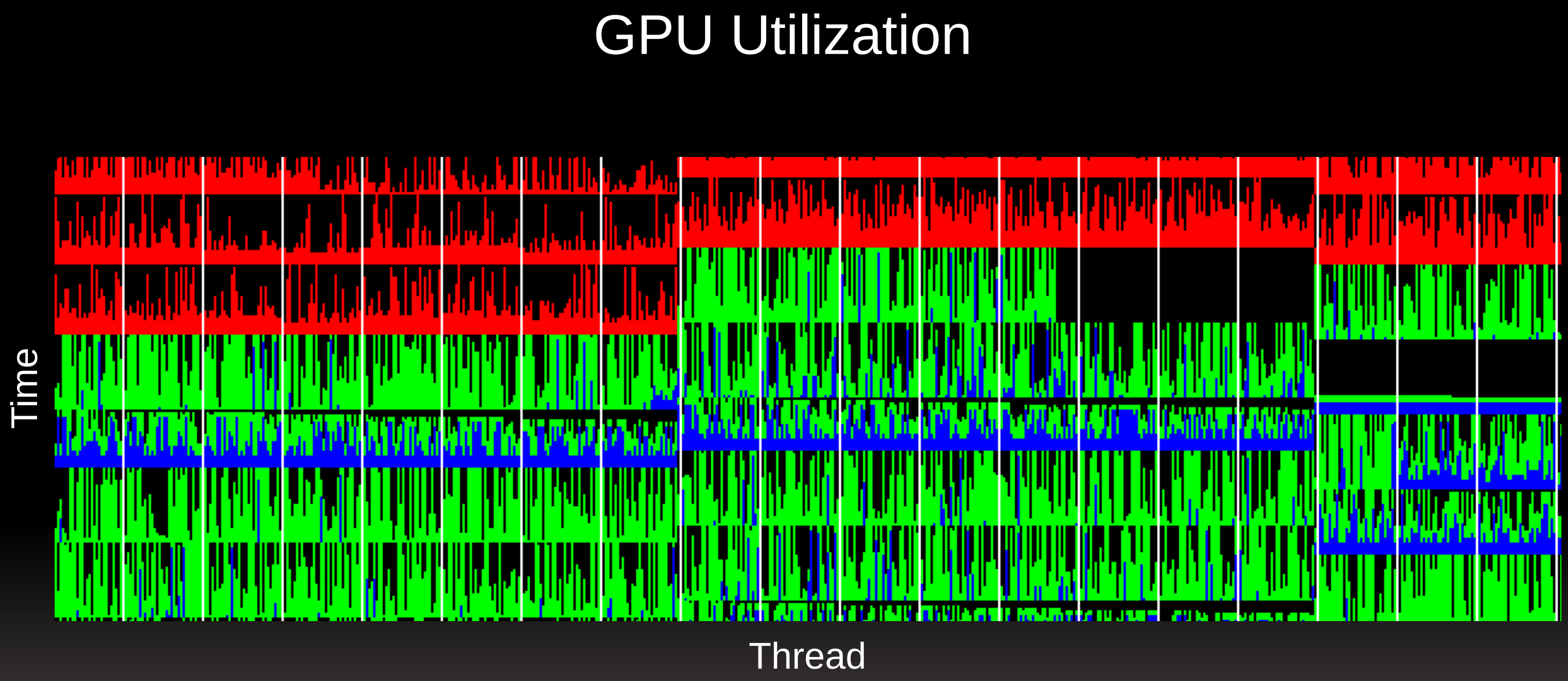
Introduction

Parallel Tracklet Construction

Tracklets are independent and can be processed simultaneously
Because of Data Locality the Tracklets are processed for a common Row

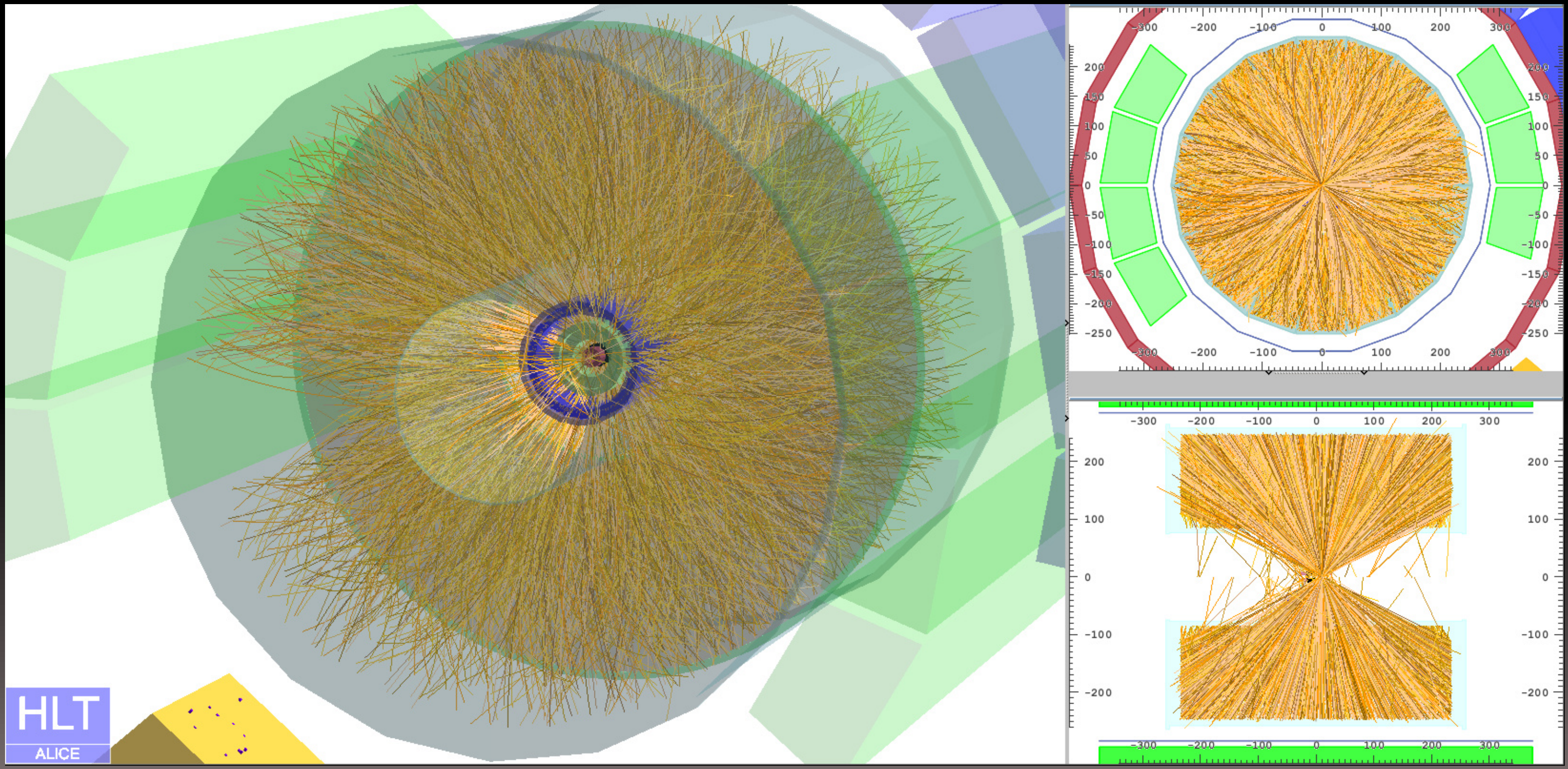


Introduction



Introduction

Screenshot of ALICE Online-Event-Display
during first physics-fill with active GPU Tracker



INTEGRATION

Integration

- GPU and CPU tracker share a common source files.
- Specialist wrappers for CPU and GPU exist, that include these common files.

common.cpp:

```
__DECL FitTrack(int n) {
….
}
```

cpu_wrapper.cpp:

```
#define __DECL void
#include ``common.cpp``

void FitTracks() {
  for (int i = 0;i < nTr;i++) {
    FitTrack(n);
  }
}
```

gpu_wrapper.cpp:

```
#define __DECL __device void
#include ``common.cpp``

__kernel void FitTracksGPU() {
  FitTrack(threadIdx.x);
}

void FitTracks() {
  FitTracksGPU<<<nTr>>>();
}
```

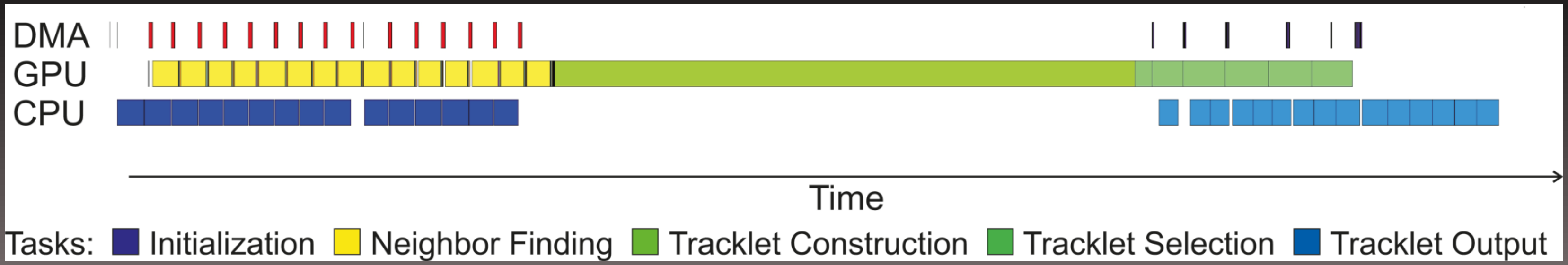
Integration

- The GPU Tracker is accessed via a virtual interface. The actual implementation is contained in a dedicated library (cagpu), which links against the CUDA runtime.
- AliRoot opens cagpu with dlopen, this creates a clear separation between AliRoot and CUDA.
- The same AliRoot binaries can be used on compute nodes with GPU and without GPU.
- This scheme is easily adoptable to other programming APIs, such as OpenCL.

CPU/GPU Performance

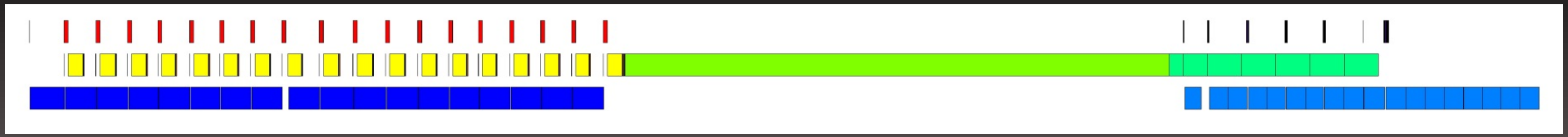
GPU Tracker Performance

- For good performance the GPU tracker pipelines the slices such that initialization on CPU, GPU tracking, and DMA transfer can overlap.
- Pipeline on old hardware works well, initialization on CPU and first GPU step require similar time.



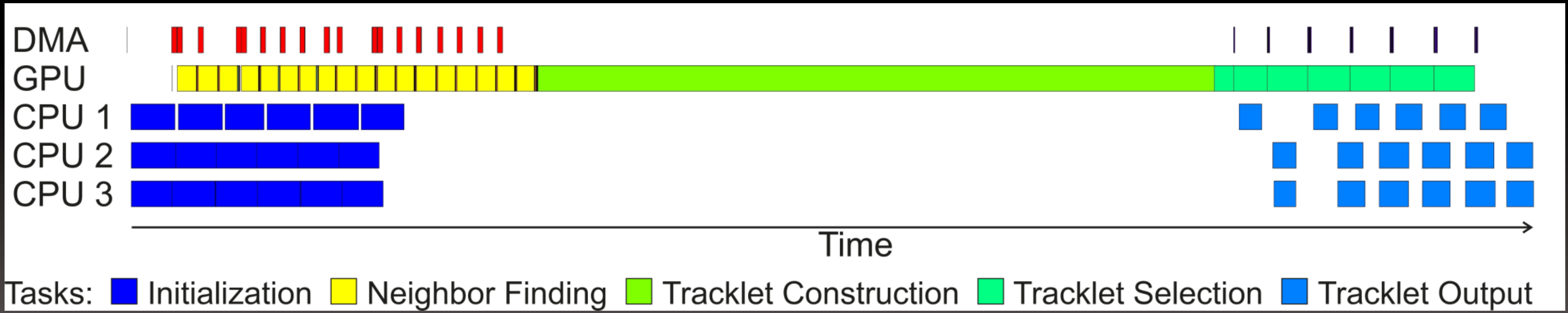
GPU Tracker Performance

- On new hardware, Fermi GPU and Magny-Cours CPU, simple pipeline does not work
  - Per-core performance of Magny-Cours is lower than for Nehalem (even though total peak is better), but the GPU tracker was single-threaded
  - New Fermi GPU accelerates GPU tracking



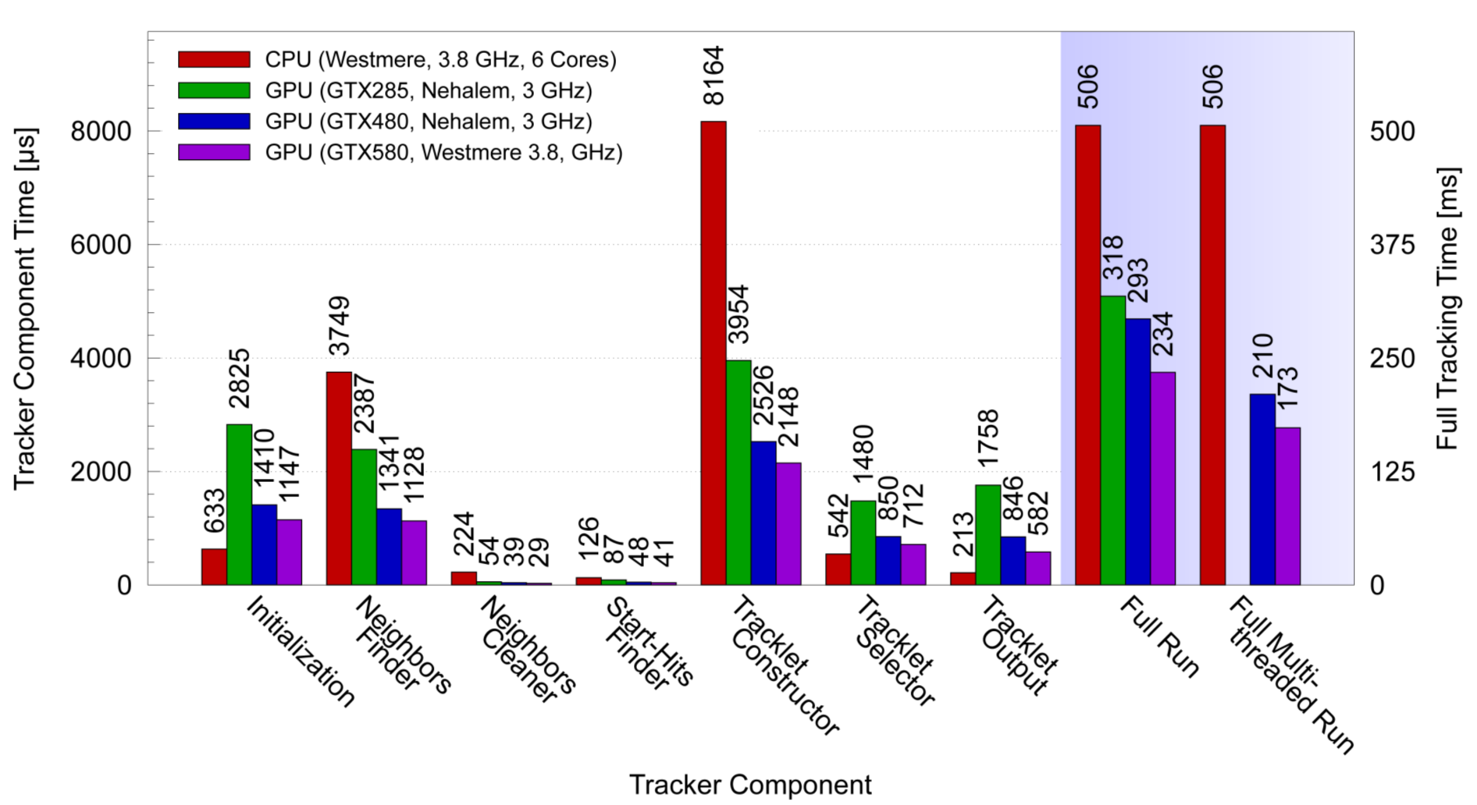
GPU Tracker Performance

- Solution:
  - Multiple GPU cores used in the pipeline.
  - The CPU threads process the slices in a round-robin fashion



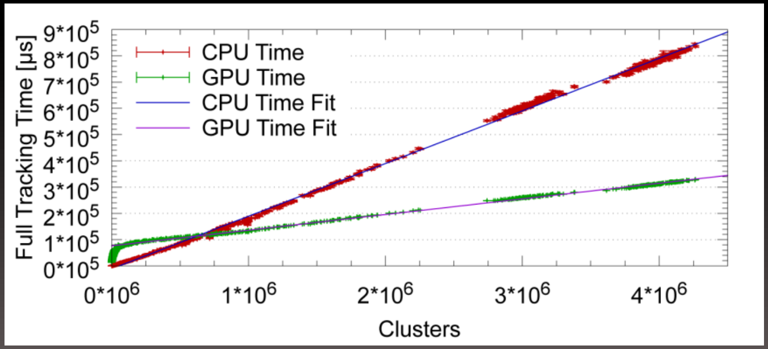
CPU / GPU Tracker Comparison

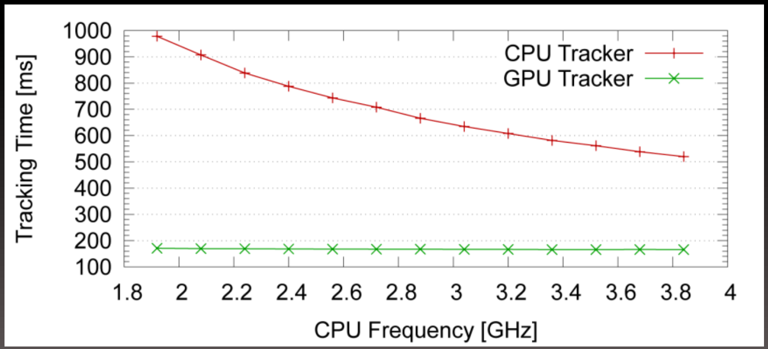
- Performance: GTX580 GPU almost three times as fast as 6-core processor.



GPU Tracker Performance

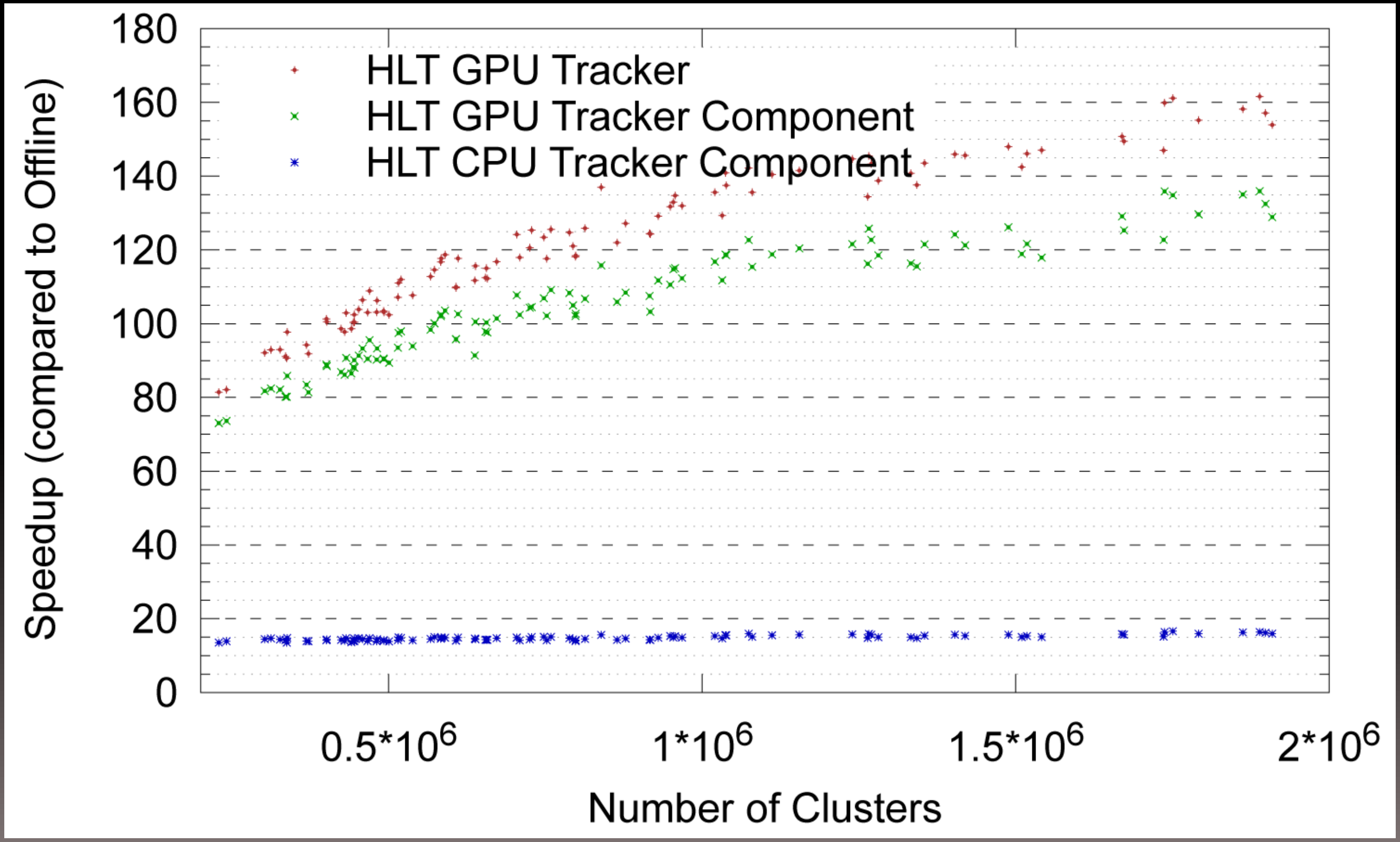
- Tracking time depends linearly on input data size.
- GPU tracking time independent from CPU performance (if initialization is fast enough).





GPU Tracker Performance

- Speedup of HLT GPU tracker v.s.offline and CPU Tracker (four CPU cores used each)



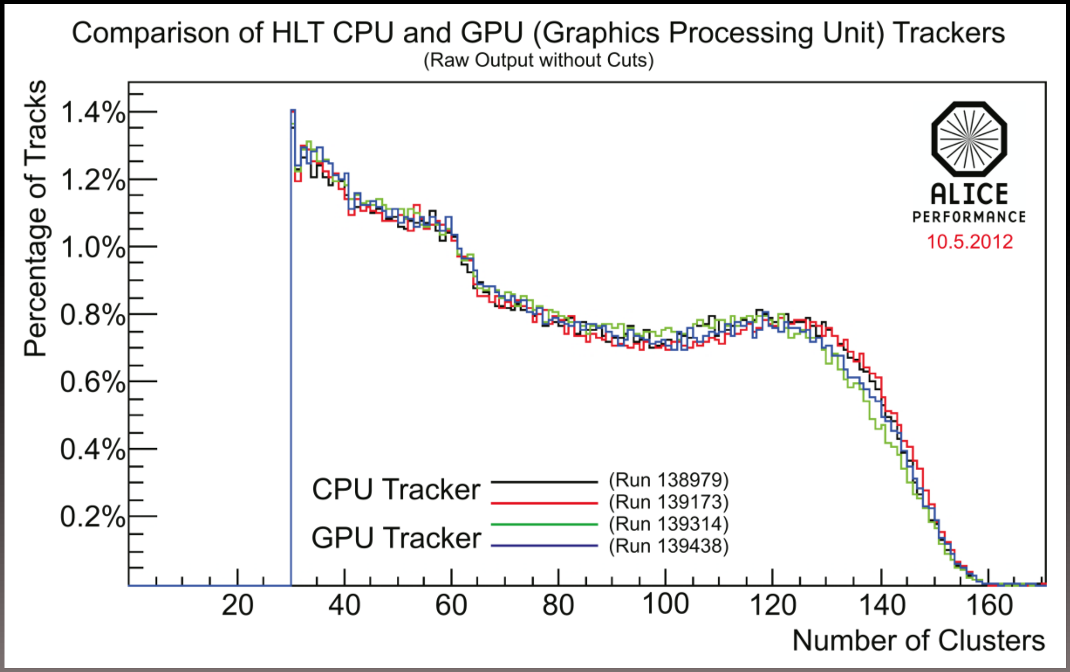
CPU / GPU Tracker Comparison

CPU / GPU Tracker Comparison

- Comparison of GPU and CPU Tracker during 2010 run
  - No significant variations in physically observables.
  - Only the number of clusters per track statistics shows a variation.

CPU / GPU Tracker Consistency

- Inconsistencies during November 2010 run
  - Cluster to track assignment
  - Track Merger
  - Non-associative floating point arithmetics



CPU / GPU Tracker Consistency

- Cluster to track assignment
  - Problem: Cluster to track assignment was depending on the order of the tracks.
    - Each cluster was assigned to the longest possible track. Out of two tracks of the same length, the first one was chosen.
    - Concurrent GPU tracking processes the tracks in an undefined order.
  - Solution: Both the chi^2 and the track lenth are used as criteria. It is extremely unlikely that two tracks coincide in both values.

CPU / GPU Tracker Consistency

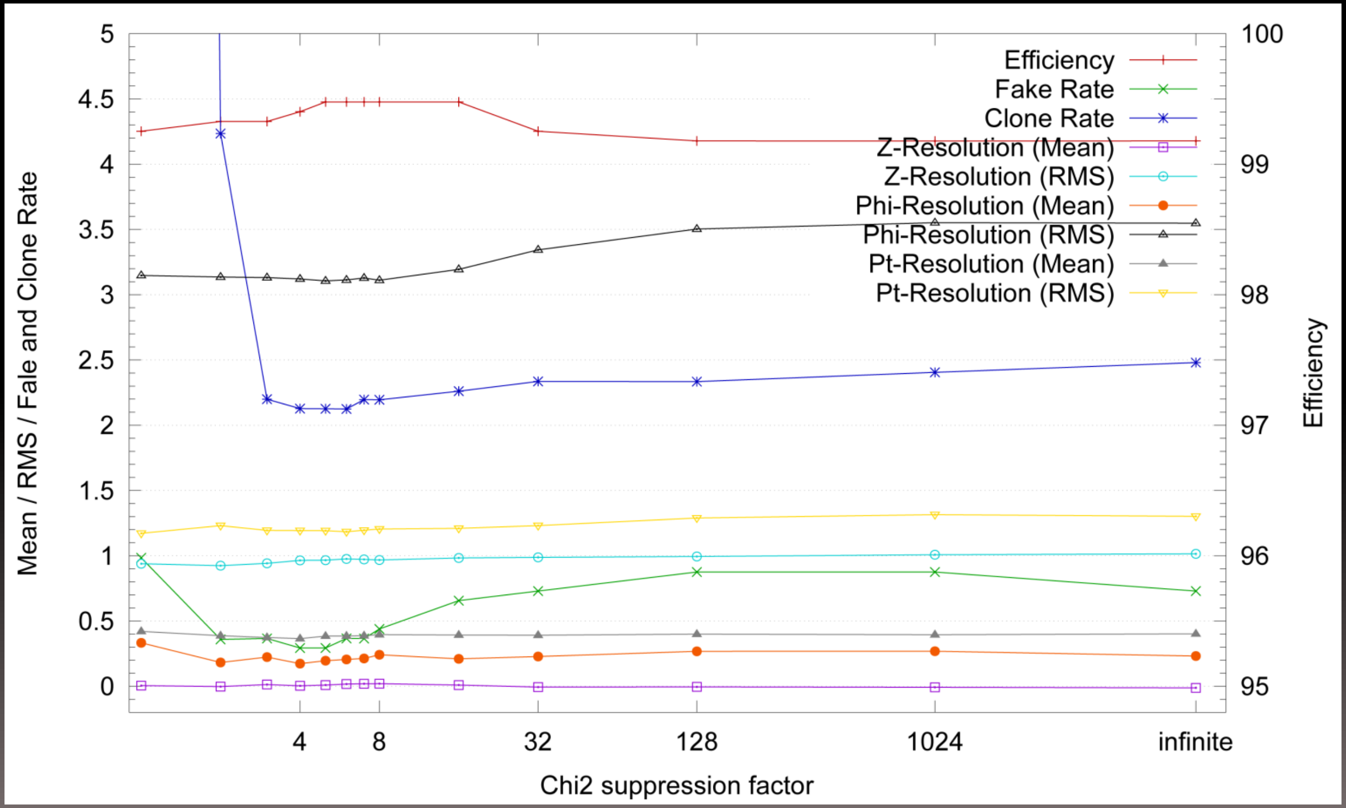
- How to combine chi² and track length?
  - Regarding the deviation between the track and the cluster for each cluster individually leads to many clones.
  - Hence, the total deviation of the track is used.
  - Small tracks have a higher probability for having a small chi², the right weight for both parameters must be determined.
  - Therefore, a chi² suppression factor is introduced, that weigths chi² less than the tracklet length.

CPU / GPU Tracker Consistency

- Determinining best suppression factor
  - A factor of infinite equals the old method were only the track length is decisive.
  - Incorporating chi^2 improves efficiency and resolution.
  - At low suppression factor only the chi^2 is decisve and the tracking becomes unstable.
  - Currently, a factor of 6 is used.

CPU / GPU Tracker Consistency

- Determinining best suppression factor



CPU / GPU Tracker Consistency

- Track merger
  - Problem: Result of the track merger depended on the order of input tracks.

  - Solution: Merger input is sorted.
    - Sorting is performed during a reformatting step.

    → No additional data copy.

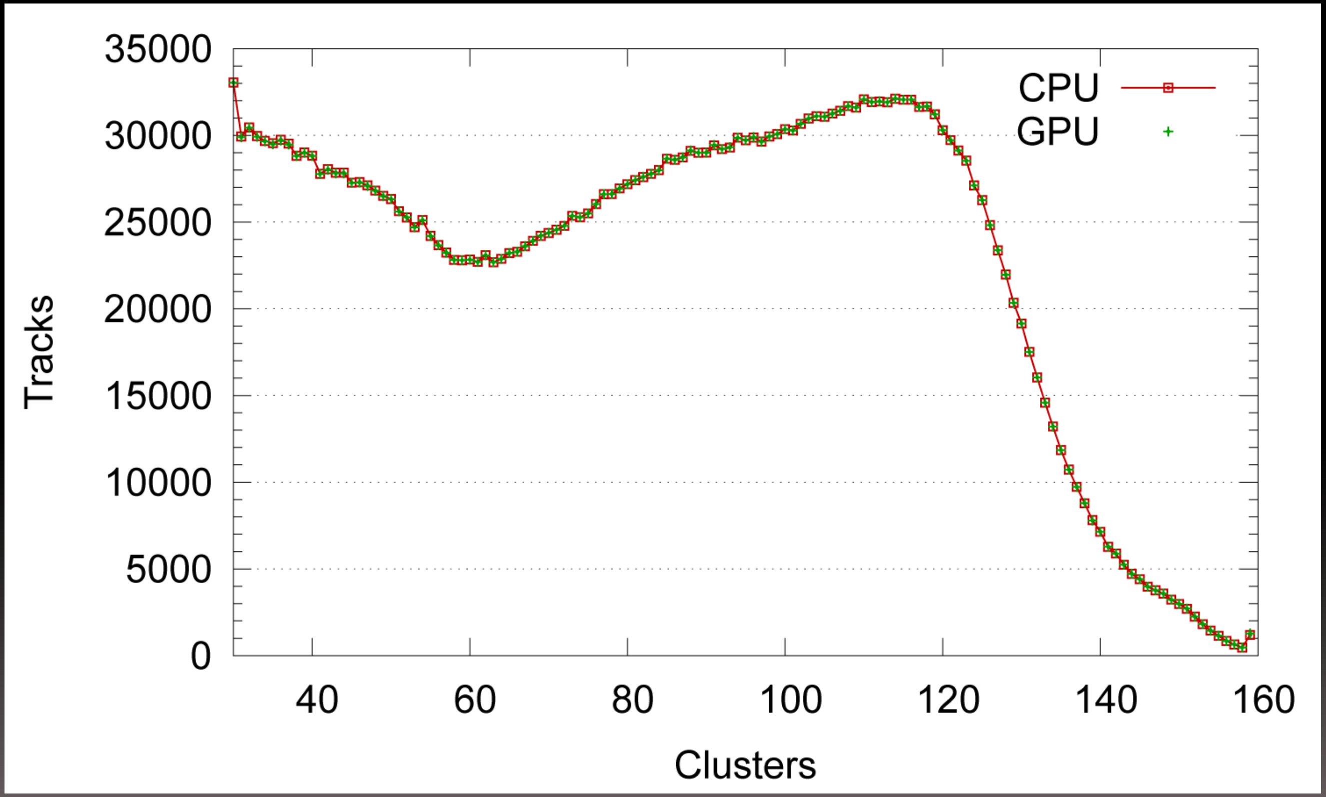
    → No performance penalty.

CPU / GPU Tracker Consistency

- Non associative floating point arithmetics
  - Problem: Different compilers perform the arithmetics in different order (also on the CPU).
  - Solution: Cannot be fixed, but...
    - Slight variations during the extrapolations do not matter as long as the clusters stay the same.
    - Inconsistent clusters: 0,00024%

CPU / GPU Tracker Consistency

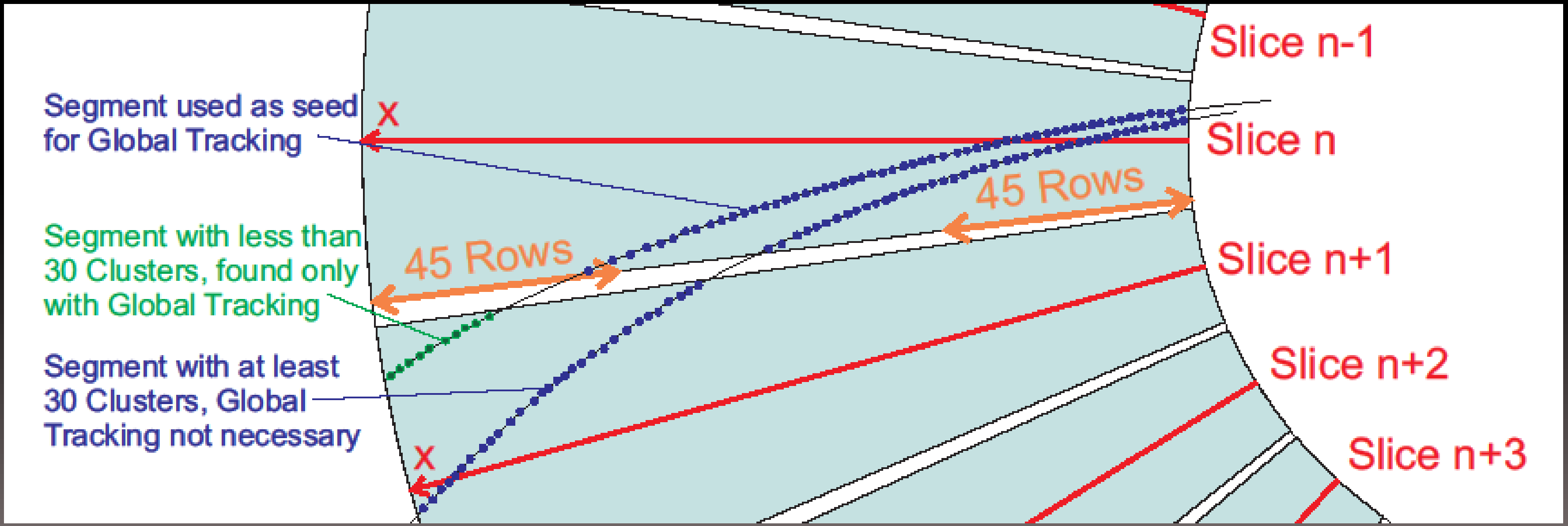
- Cluster per track statistic with improvements



GLOBAL TRACKING

Global Tracking

- Original HLT tracker did not find track segments of less than 30 clusters in a slice.
- An additional step before the merger can find these segments.

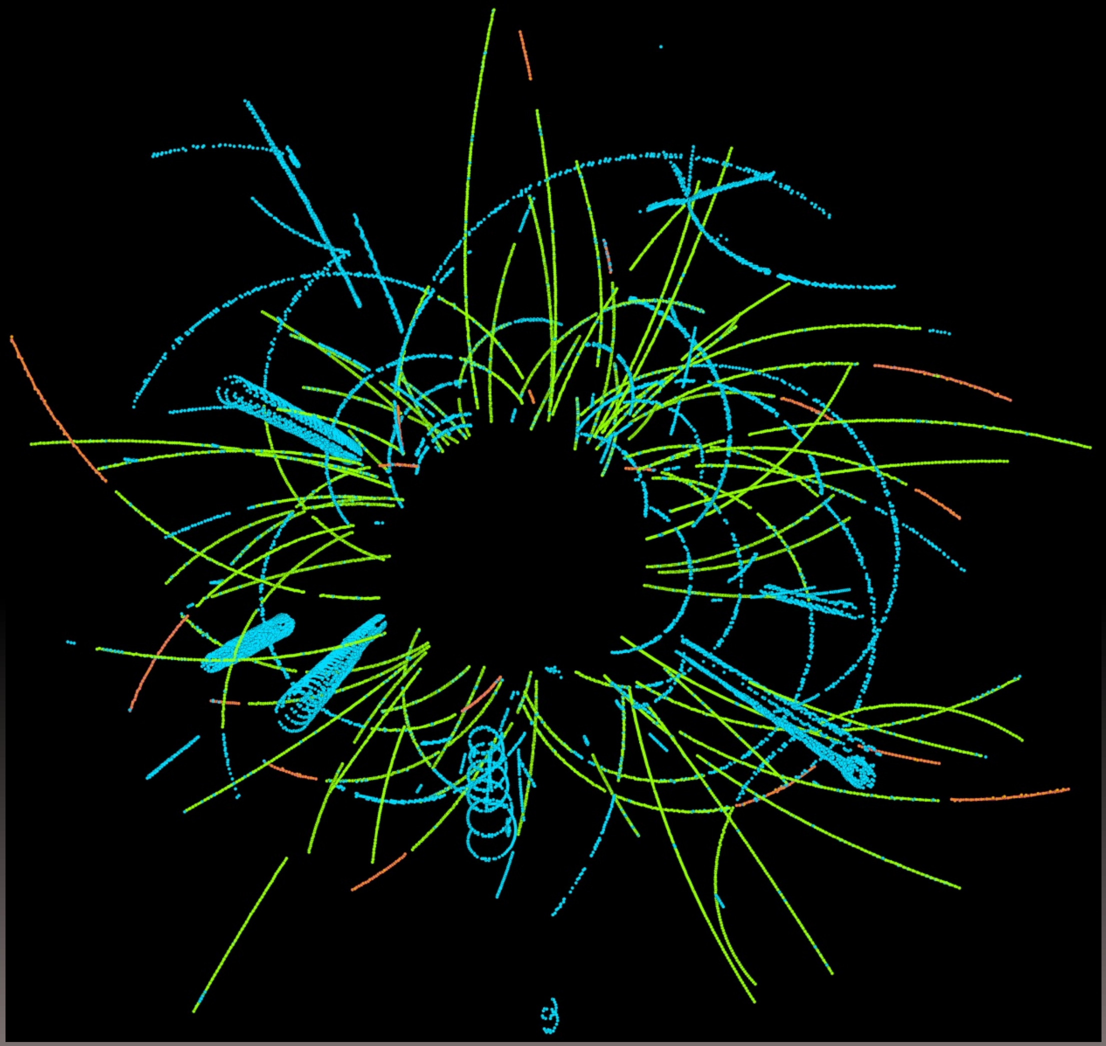


Global Tracking

- No additional tracks found.
- Newly found track segments automatically merged with the track used as seed.
- Efficiency / clone- / fake-rate unchanged.

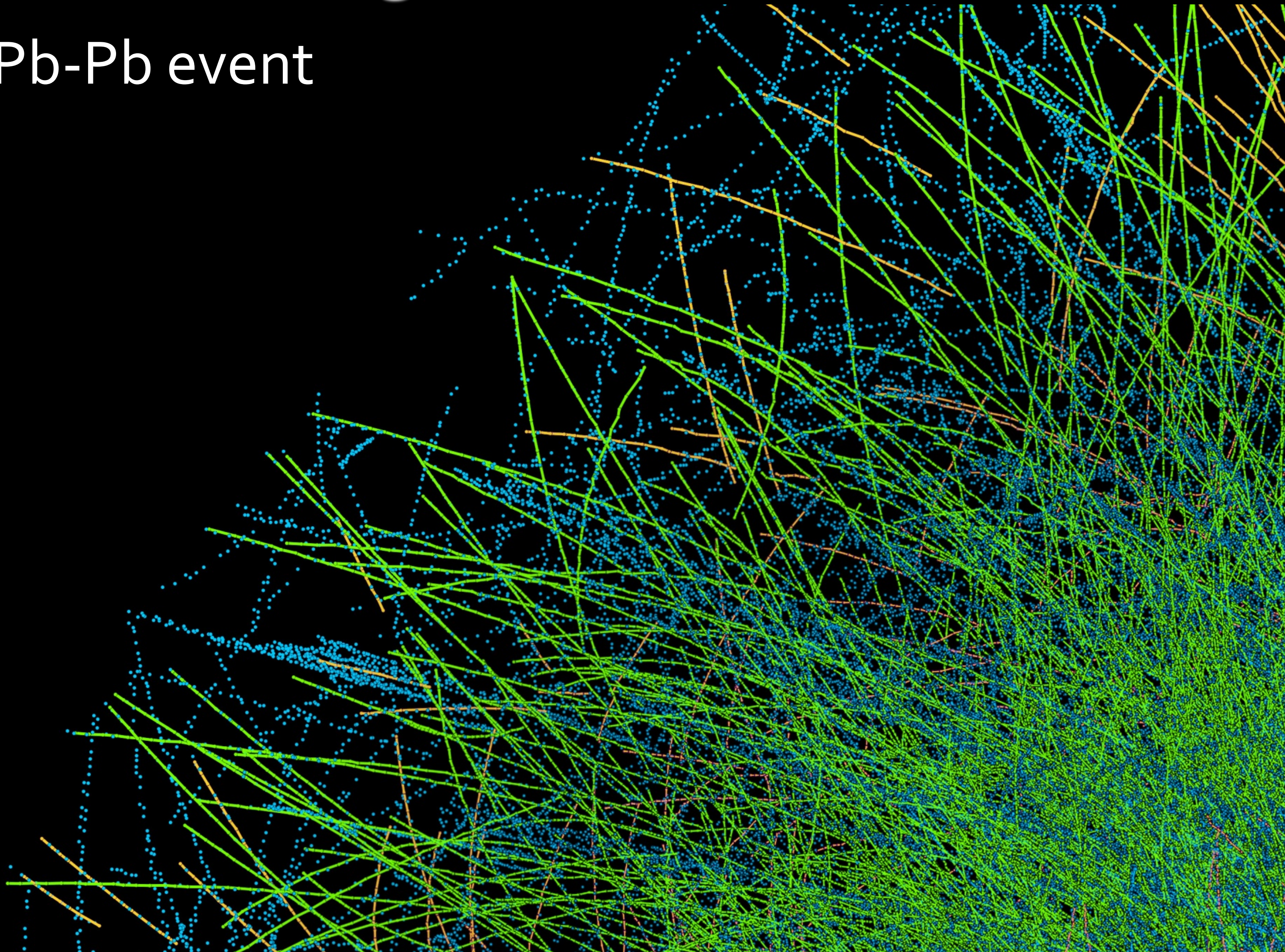
Global Tracking

- PP event
- Original segments green
- Additional segments orange



Global Tracking

- Pb-Pb event



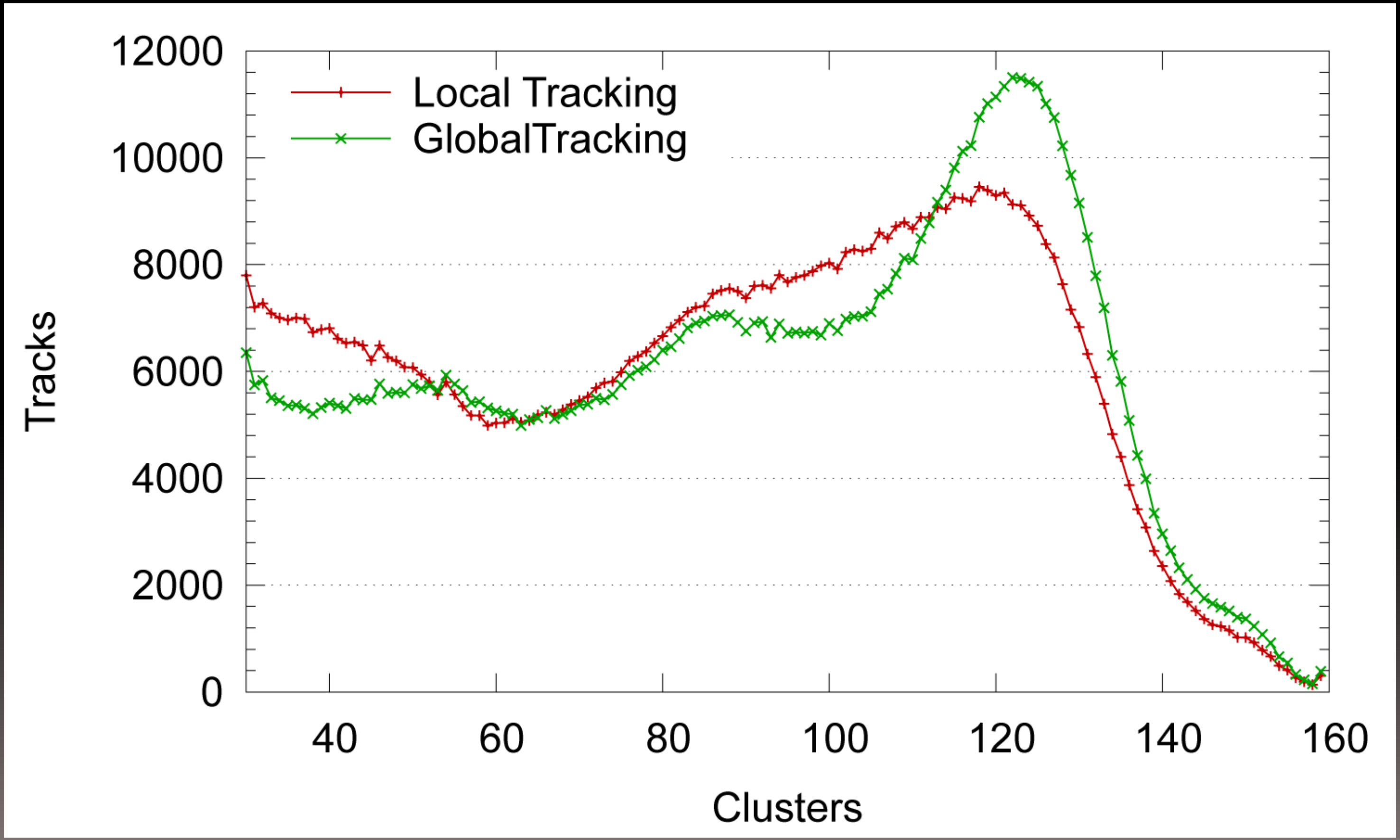
Global Tracking Performance

Tracking time increase with global tracking enabled.

Tracker	Time increase
CPU Tracker (single threaded)	2.19%
CPU Tracker (multi-threaded)	12.60%
GPU Tracker	9.03%

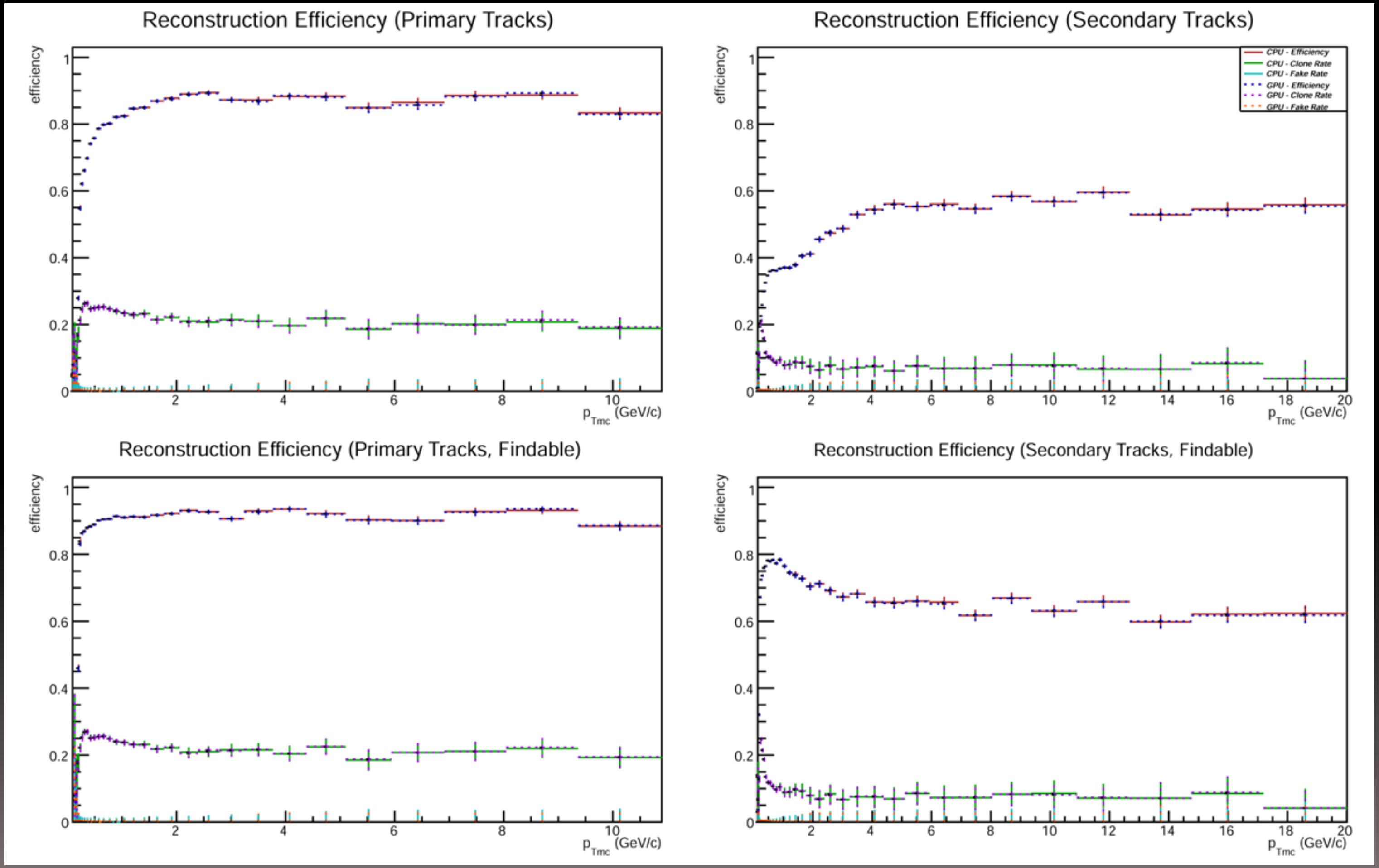
Global Tracking

- Cluster per track statistics comparison:



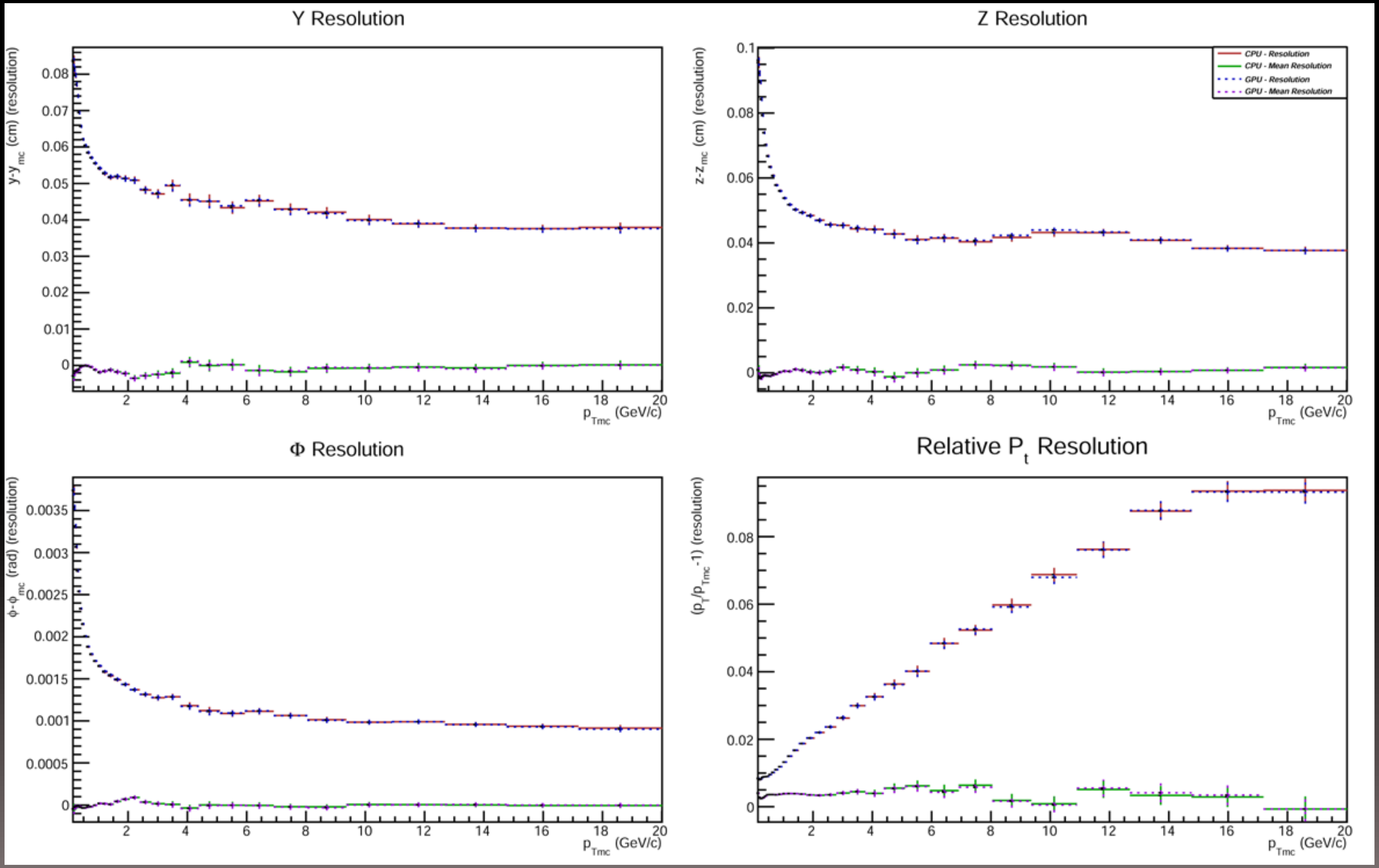
CPU / GPU Tracker Consistency

- Efficiency Comparison



CPU / GPU Tracker Consistency

- Resolution Comparison



Summary

- Threefold performance increase of GPU tracker compared to all CPUs of a node, tenfold increase in a reasonable HLT scenario.
- GPU tracker performance is independent from CPU and depends linearly on data size.
- Results of GPU and CPU tracker match almost completely. Only 0.00024% of the clusters differ due to non-associative floating-point arithmetic.
- Common source code ensures great maintainability, separation from libAliHLTTPC makes a common binary work on all nodes – with and without GPU.
- With global tracking the tracker can track across slice boundaries but still explot data locality