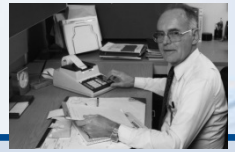
Overview of ALICE GPU computing and optimization strategies for GPU based LHC event filtering applications

Dr. David Rohr, drohr@cern.ch

Frankfurt Institute for Advanced Studies

CERN, 12.1.2016

INTRODUCTION

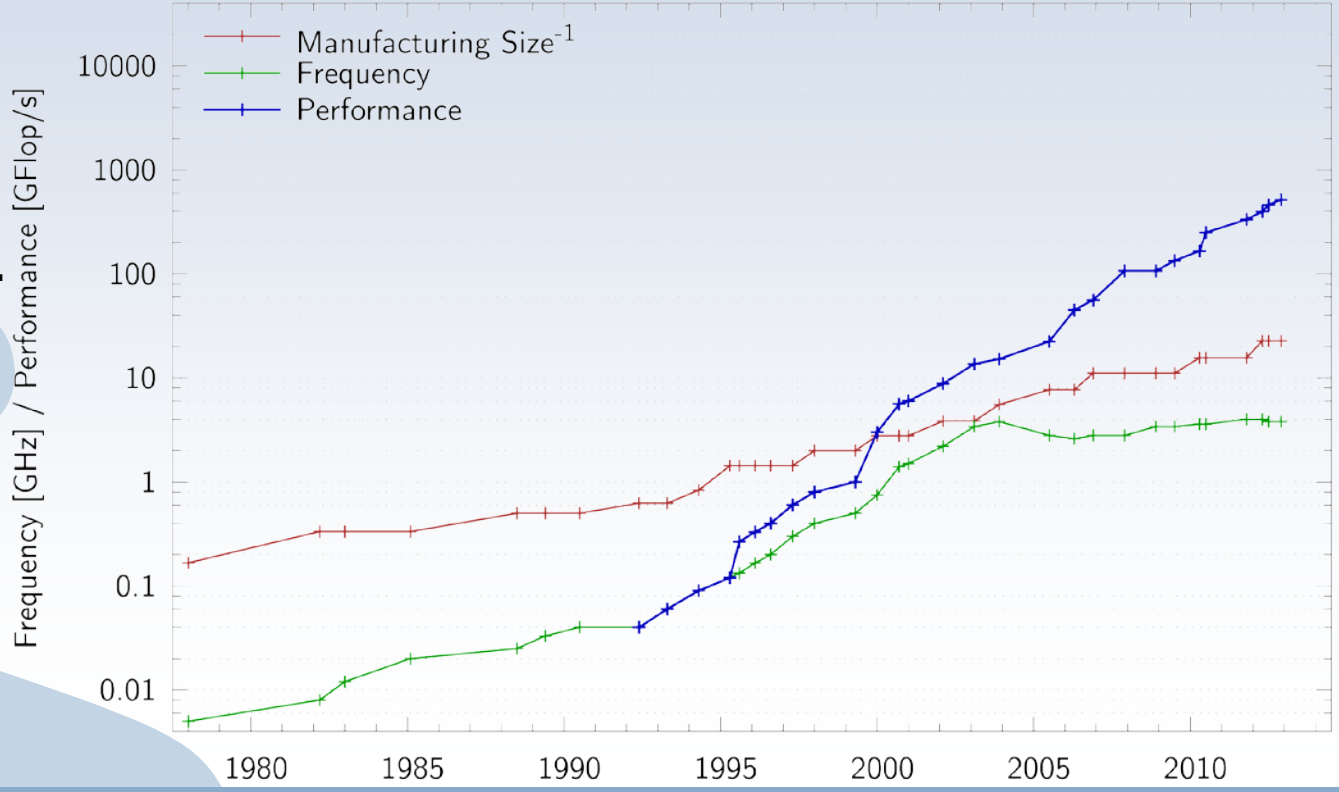


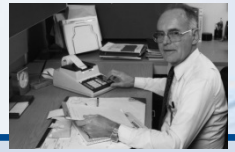
Introduction

FIAS Frankfurt Institute for Advanced Studies

Moore's Law: Manufacturing size, frequency, and performance grow exponentially.

Frequency began to stagnate 2003.





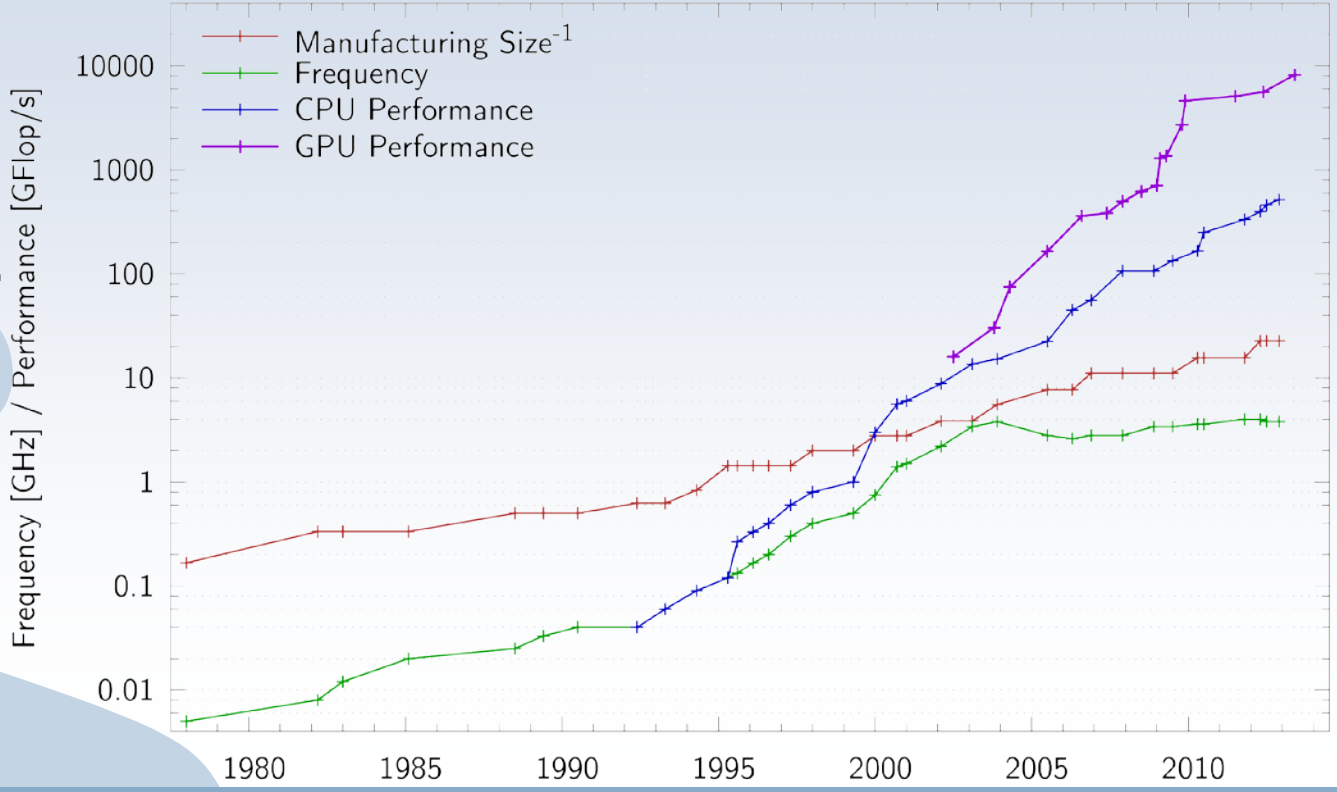
Introduction

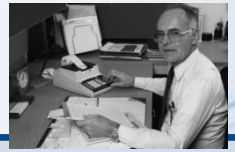
FIAS Frankfurt Institute for Advanced Studies

Moore's Law: Manufacturing size, frequency, and performance grow exponentially.

Frequency began to stagnate 2003.

GPUs are faster than CPUs.



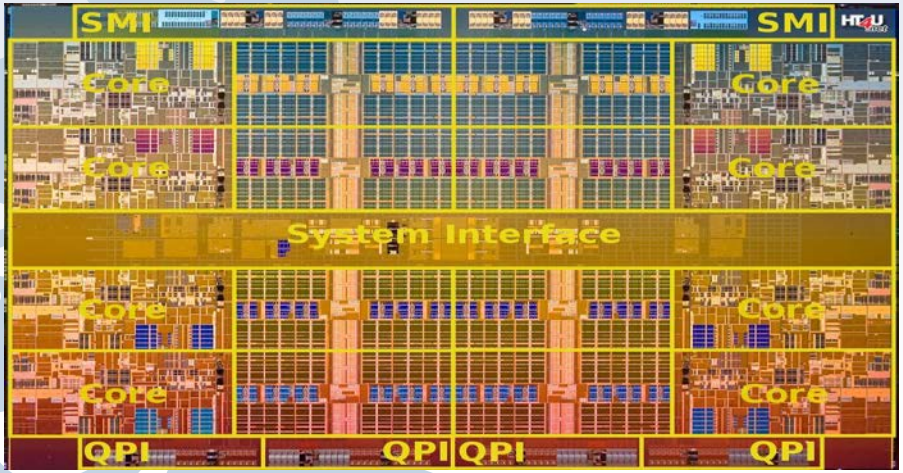


Why GPUs

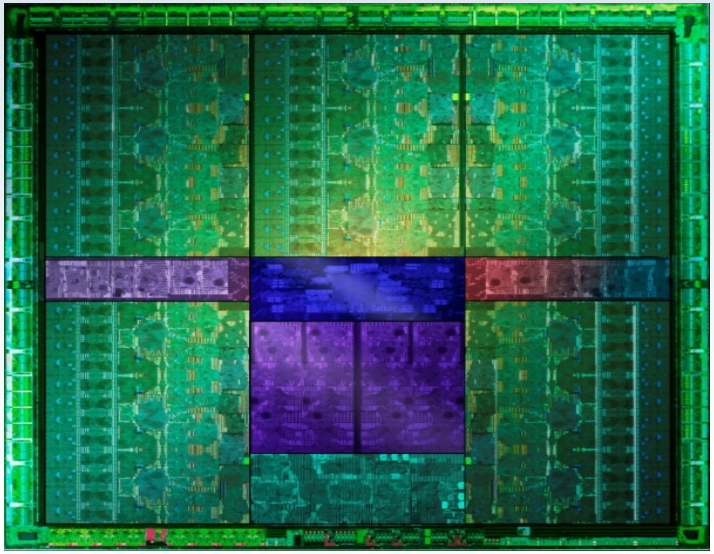
FIAS Frankfurt Institute for Advanced Studies

GPUs use their silicon for ALUs

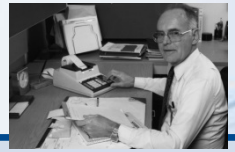
CPUs use their silicon mainly for caches, branch prediction, etc.



Intel Nehalem



NVIDIA Kepler



Why GPUs

FIAS Frankfurt Institute for Advanced Studies

CPUs are designed for fast execution of serial programs.

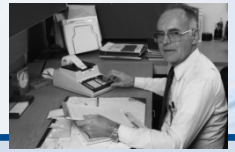
- Clocks have reached a physical limit.

  →Vendors use parallelization to increase performance.

GPUs are designed for parallel execution in the first place.

- The „only“ limit for GPU performance is heat dissipation.
- GPU clocks are usually lower than they could be.
  - This saves power
  - Hence more hardware can be powered in parallel

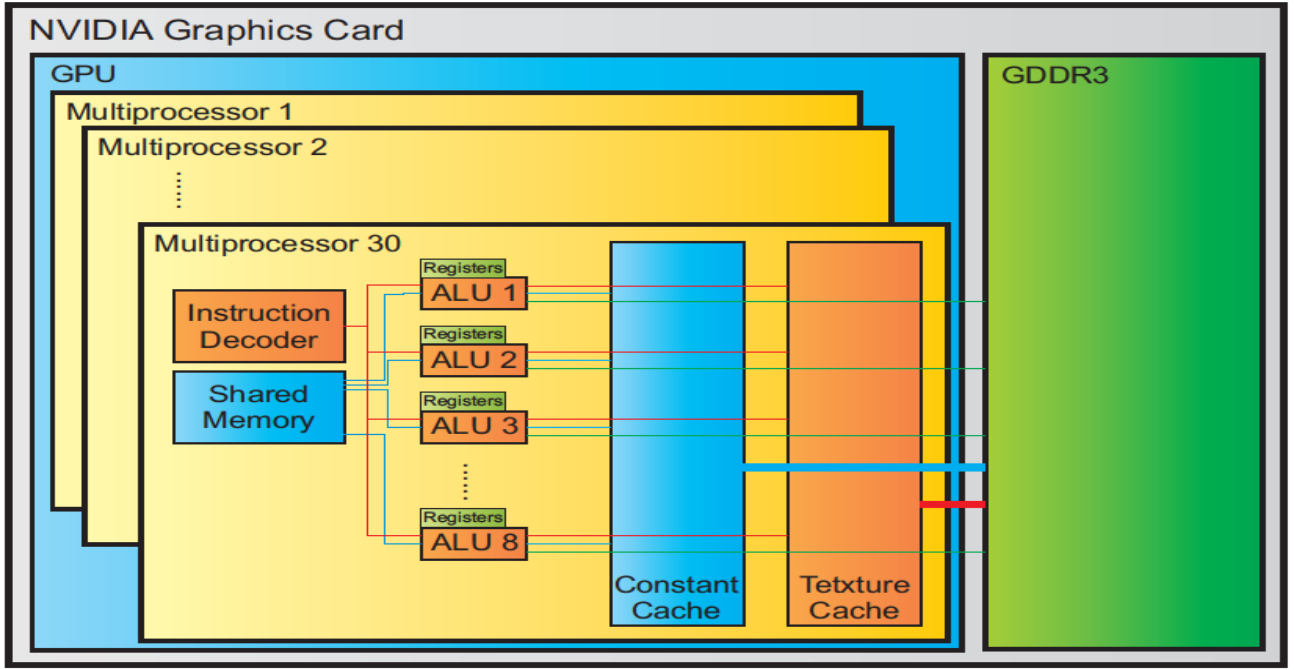
    → Better overall performance



Introduction

FIAS Frankfurt Institute for Advanced Studies

NVIDIA GTX280 GPU



FIAS Frankfurt Institute for Advanced Studies

WHAT PERFORMANCE CAN WE EXPECT FROM GPUS

Comparing CPU / GPU Performance

FIAS Frankfurt Institute for Advanced Studies

- **The compute performance alone is no reasonable metric!**
  - The GPU is the faster chip – by construction
- **We consider the following:**

$$\mathcal{E} = \frac{\text{Efficiency on GPU}}{\text{Efficiency on CPU}} = \frac{(a_g/p_g)}{(a_c/p_c)} = \frac{a_g/a_c}{p_g/p_c}.$$

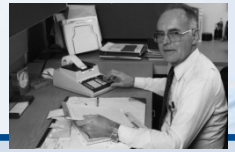
- **Most of our applications reach about 70% or more in this metric.**
- **There are exclusions:**
  - PCI Express can limit the performance (track merger, encoding with small n / k).
  - CPU Compilers are better and allow more flexible core (JIT-compiled encoding).
  - CPU caches can better hide memory latencies (Electron Microscopy).

Comparing CPU / GPU Performance

FIAS Frankfurt Institute for Advanced Studies

- **Overview of speedup in several applications:**

Benchmark	Type	Hardware	Performance	% of peak	Speedup	ε [%]
(old) HLT Tracker	Single	Nehalem 4C 3 GHz	1122 ms			
		GTX285 + CPU	312 ms		3.60	53
(new) HLT Tracker	Single	2×Magny-Cours 2.2 GHz	495 ms			
		GTX580 + CPU	155 ms		3.19	85
Track Merger	Single	Westmere 6C 4 GHz	65 ms			
		GTX580 + CPU	60 ms		1.10	13
DGEMM (Kernel)	Double	2×Magny-Cours 2.1 GHz	180 GFlop/s	89.3		
		5870	494 GFlop/s	90.8	2.74	102
		6970	624 GFlop/s	92.3	3.47	103
		7970	805 GFlop/s	84.4	4.47	95
DGEMM (System)	Double	2×Magny-Cours 2.1 GHz	180 GFlop/s	89.3		
		5870 + CPU	623.5 GFlop/s	83.6	3.46	94
		3×5870 + CPU	1435 GFlop/s	78.3	7.98	87
		2×6990	2292 GFlop/s	89.9	12.73	104
		2×S10000	2923 GFlop/s	79.8	16.24	89
One-Node HPL	Double	2×Magny-Cours 2.1 GHz	174.6 GFlop/s	86.6		
		5870 + CPU	563.2 GFlop/s	75.5	3.23	87
		3×5870 + CPU	1114 GFlop/s	60.7	6.38	70
		2×6990 + CPU	2007 GFlop/s	72.4	11.49	84
		2×S10000 + CPU	2679 GFlop/s	73.1	15.34	84
Erasure Codes (small n)	32-bit logical	Westmere 6 · 3.8 GHz	14.3 GB/s	74.7		
		GTX580	72.5 GB/s	75.3	5.32	102
		6970	51.1 GB/s	58.0	4.10	78
		Virtex 6 LX240 FPGA	2187.0 GB/s		152.94	
Erasure Codes (large n)	32-bit logical	Sandy Bridge 1 · 3.7 GHz	251.0 GAOp/s			
		Westmere 6 · 3.8 GHz	807.0 GAOp/s			
		GTX580	908.4 GAOp/s		1.13	19
		6970	1024 GAOp/s		1.27	26



GPU Expectations

FIAS Frankfurt Institute for Advanced Studies

- **Many problems can be ported to GPUs efficiently.**
  - Some problems suffer inherent bottlenecks, when brought to GPUs:
    - PCI Express Bandwidth
    - Compiler Inefficiencies
  - Often easy to spot
    - Allows for easy plausibility checks whether GPU adaptation is possible.
- **Reasonable, very rough speedup expectation:**
  about factor 3 per GPU
- **Claims of larger speedup (factor 10 and above) are often comparisons to old / unoptimized / single-core CPU software.**

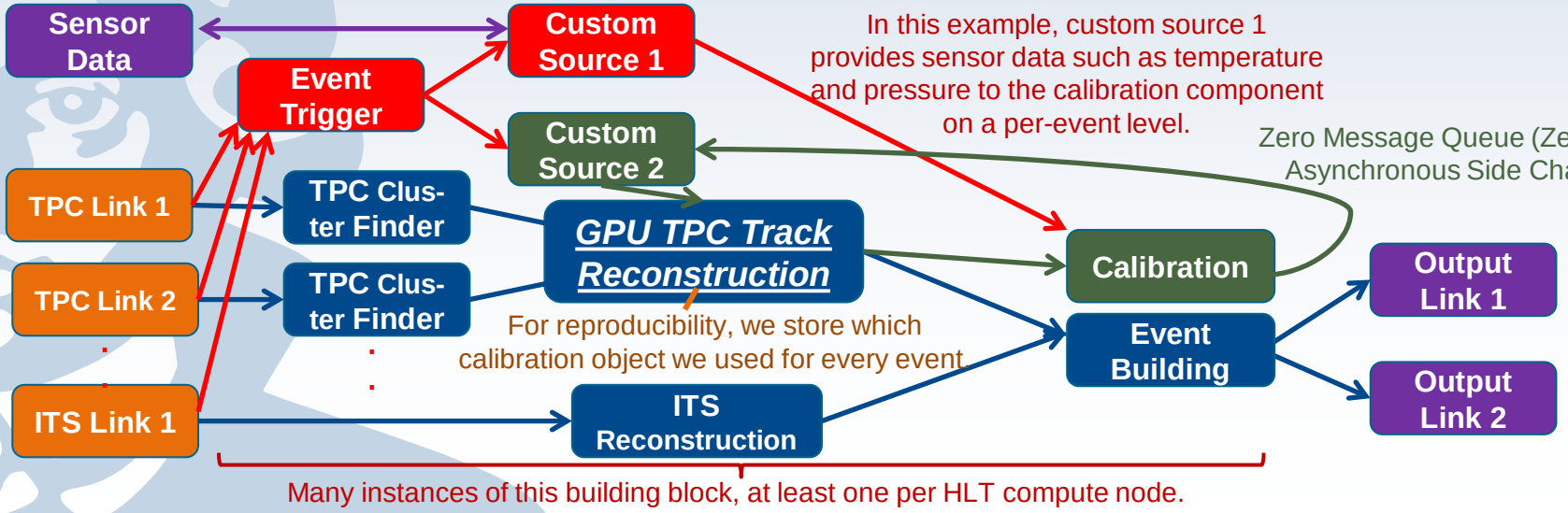
FIAS Frankfurt Institute for Advanced Studies

ALICE OVERVIEW

Overview of ALICE HLT Reconstruction

FIAS Frankfurt Institute for Advanced Studies

- **Based on the online reconstruction, the next feature of the HLT will be online calibration.**
  - Reconstructed data is required to generate the calibration.
  - Calibration objects fed back in processing chain and used for reconstruction.
  - TPC calibration stable over several minutes → Short time at the beginning needed to generate first calibration.



ALICE HLT Components

- **CPU utilization per compute node – at ca. 1kHz Pb-Pb Min-Bias:**
  - TPC Cluster Finder: FPGA
  - TPC Compression: 4 CPU cores
  - TPC Cluster Transformation: 1 CPU core
  - **TPC Track Finder: 2 CPU cores + GPU – Alternatively: 20 CPU cores**
  - TPC Track Fit: 1 CPU core
  - VZERO: < 1 CPU core
  - ZDC: < 1 CPU core
  - EMCAL RECO: 7 CPU cores
  - ITS Cluster Finder: 1-2 CPU cores
  - ITS SPD Vertexer / Standalone Tracker: 1 CPU core
  - TPC prolongation to ITS: 3 CPU cores
  - ESD / Flat-ESD Creation: 3 + 2 CPU cores
  - HLT QA: < 1 CPU core
  - Luminuous Region: < 1 CPU core
  - Trigger and Readout List creation: < 1 CPU core
  - TPC Calibration (140 Hz asynchronously): 6 CPU cores
  - Framework: 2 CPU cores (Total: 34 cores – virtual HT cores)

ALICE HLT Reconstruction Sumary

- **Significant processing requirements comes from:**
  - TPC cluster finding: Handled by FPGA
  - TPC Compression: 12% (CPU algorithm being improved)
  - EMCAL: 21% (New software, not fully optimized yet)
  - ESD Creation: 9% (No speedup possible due to root files → Flat ESD without root)
  - ITS Tracking: 9% (Speedup possible, perhaps combined TPC+ITS GPU tracking)
  - TPC Calibration: 21% (Complex and extensive offline code, speedup possible, GPU adaptation unlikely)

  - TPC Tracking: 38% - Single processing hotspot
    → TPC tracking ported to GPU

  - With RCU2: Higher rate, more out-of-bunch pile-up: fraction of TPC tracking in total workload will increase.

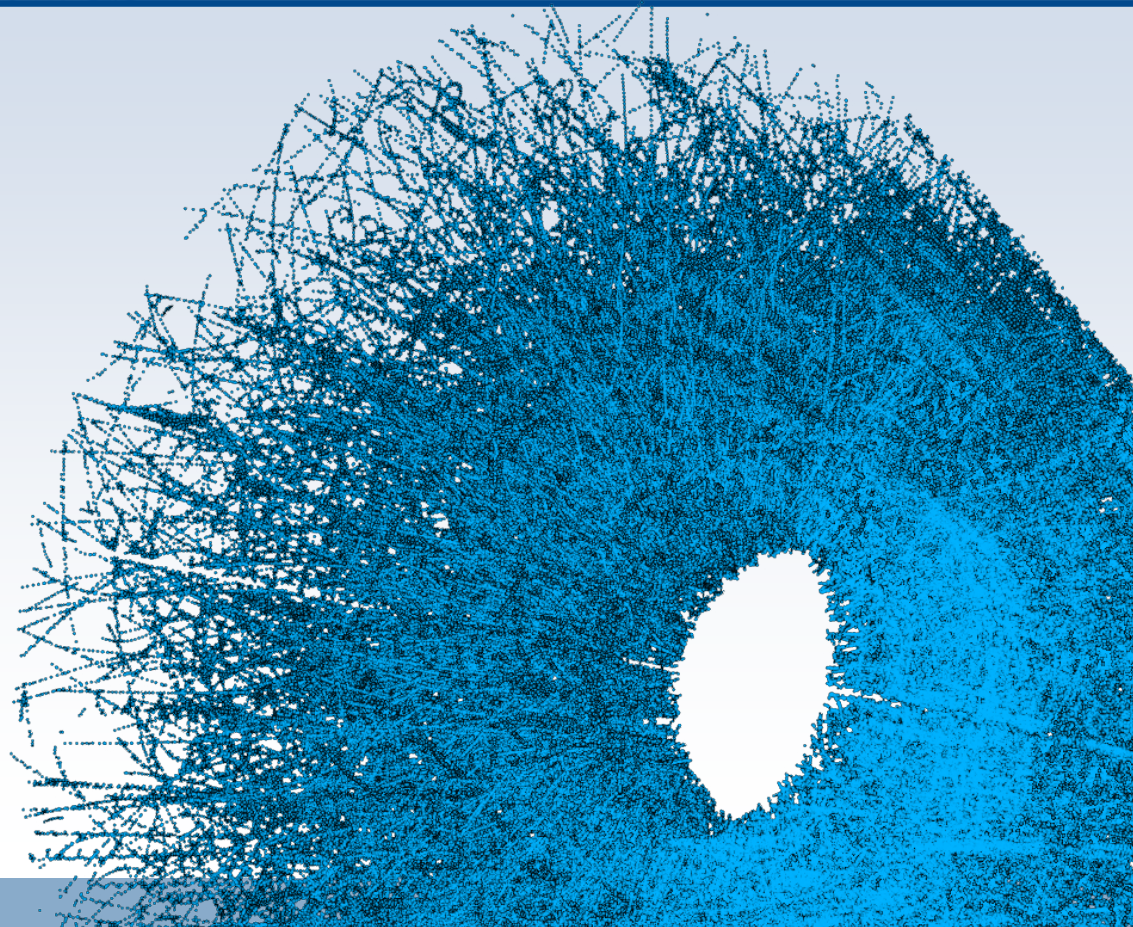
FIAS Frankfurt Institute for Advanced Studies

TRACKING ALGORITHM

Tracking at ALICE

FIAS Frankfurt Institute for Advanced Studies

ALICE events:

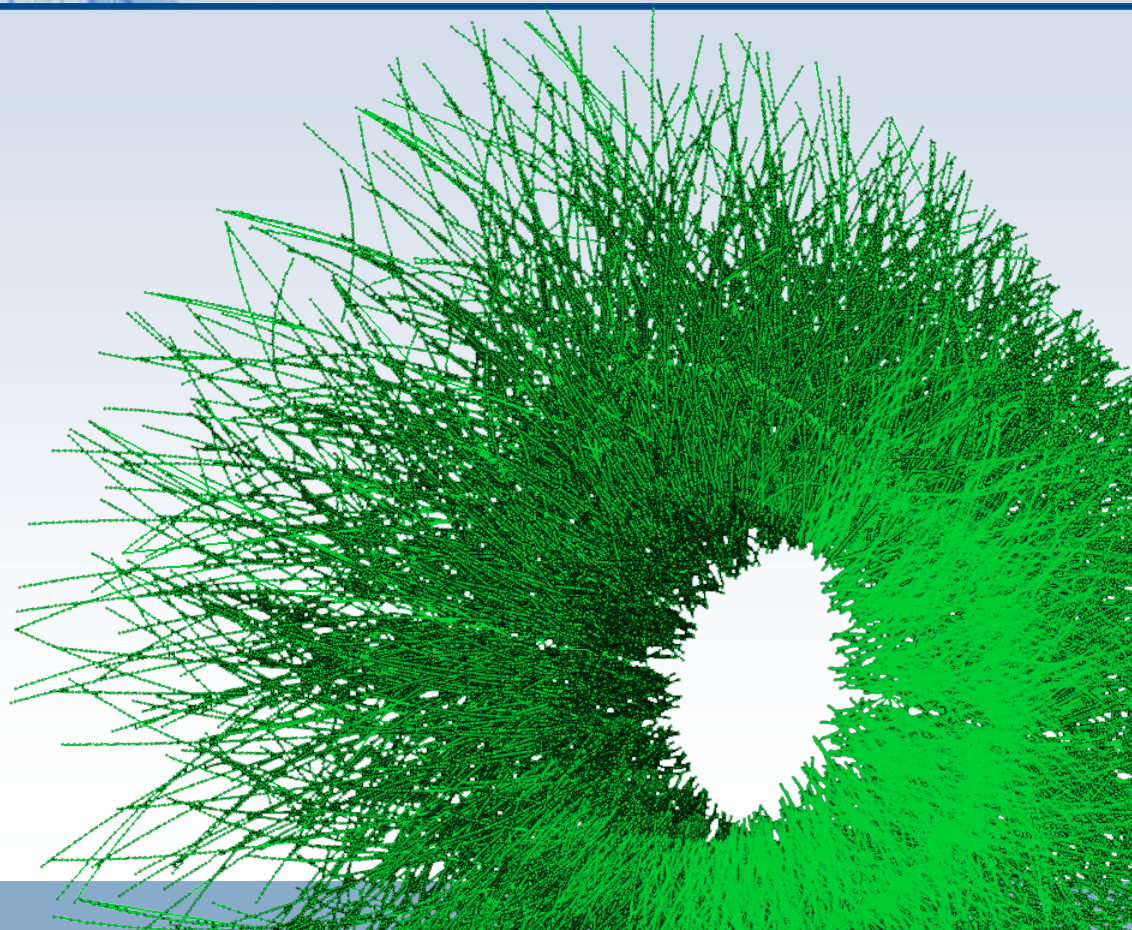


TPC Hits in heavy ion collision.

Tracking at ALICE

FIAS Frankfurt Institute for Advanced Studies

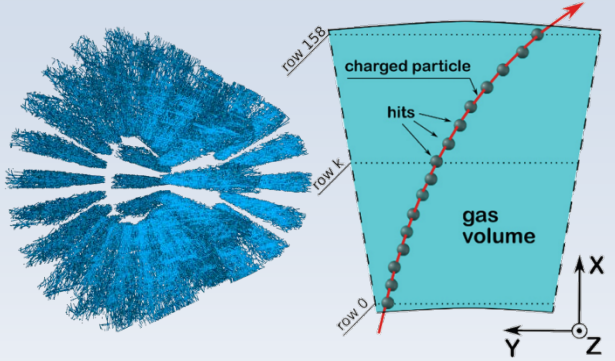
ALICE events:



TPC tracks in heavy ion collision.

Tracking Algorithm

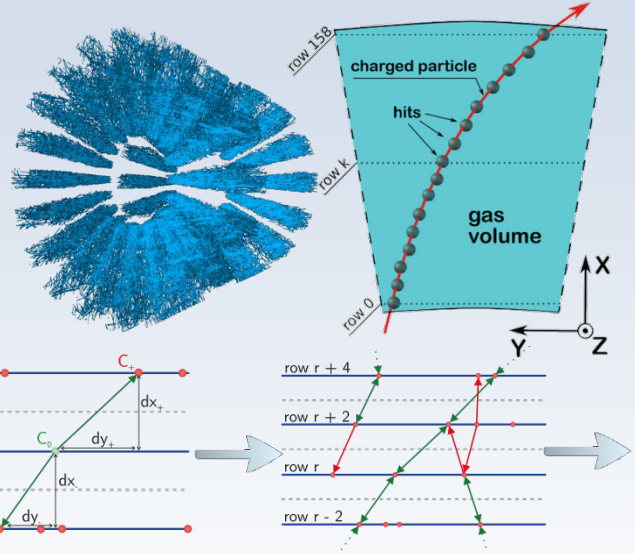
- **TPC Volume is split in 36 sectors.**
  - The tracker processes each sector individually.
  - Increases data locality, reduce network bandwidth, but reduces parallelism.
  - Each sector has 160 read out rows in radial direction.



Tracking Algorithm

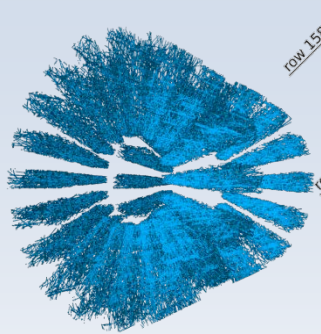
- **TPC Volume is split in 36 sectors.**
  - The tracker processes each sector individually.
  - Increases data locality, reduce network bandwidth, but reduces parallelism.
  - Each sector has 160 read out rows in radial direction.

- **1. Phase: Sektor-Tracking (within a sector)**
  - Heuristic, combinatorial search for track seeds using a Cellular Automaton.
    - A) Looks for three hots composing a straight line (**link**).
    - B) Concatenates links.

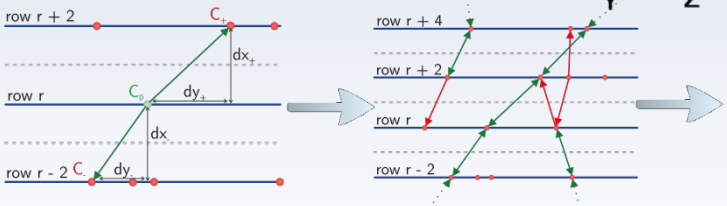


Tracking Algorithm

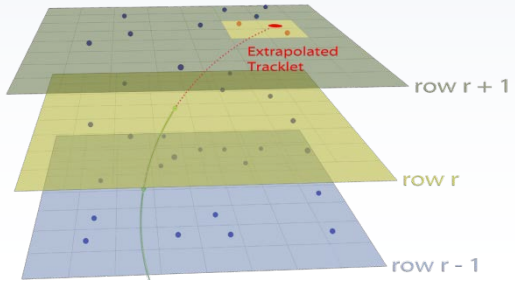
- **TPC Volume is split in 36 sectors.**
  - The tracker processes each sector individually.
  - Increases data locality, reduce network bandwidth, but reduces parallelism.
  - Each sector has 160 read out rows in radial direction.



- **1. Phase: Sektor-Tracking (within a sector)**
  - Heuristic, combinatorial search for track seeds using a Cellular Automaton.
    - A) Looks for three hots composing a straight line (**link**).
    - B) Concatenates links.

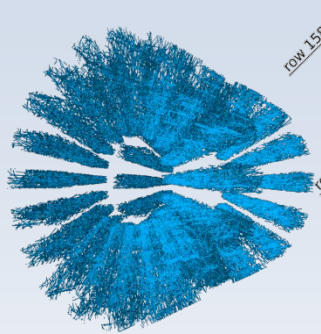


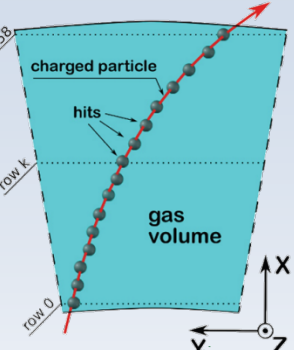
  - Fit of track parameters, extrapolation of track, and search for additional clusters using the Kalman Filter.



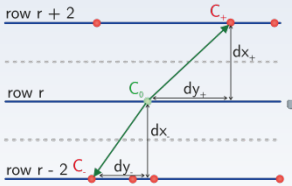
Tracking Algorithm

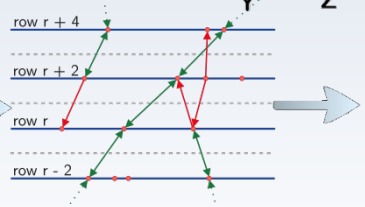
- **TPC Volume is split in 36 sectors.**
  - The tracker processes each sector individually.
  - Increases data locality, reduce network bandwidth, but reduces parallelism.
  - Each sector has 160 read out rows in radial direction.



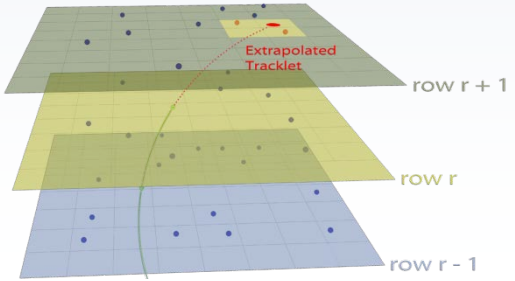


- **1. Phase: Sektor-Tracking (within a sector)**
  - Heuristic, combinatorial search for track seeds using a Cellular Automaton.
    - A) Looks for three hots composing a straight line (**link**).
    - B) Concatenates links.





  - Fit of track parameters, extrapolation of track, and search for additional clusters using the Kalman Filter.

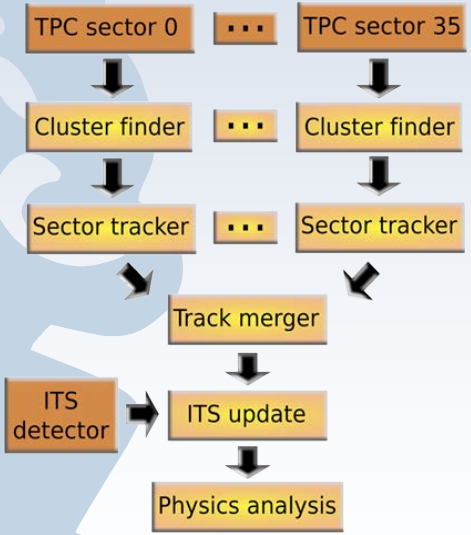


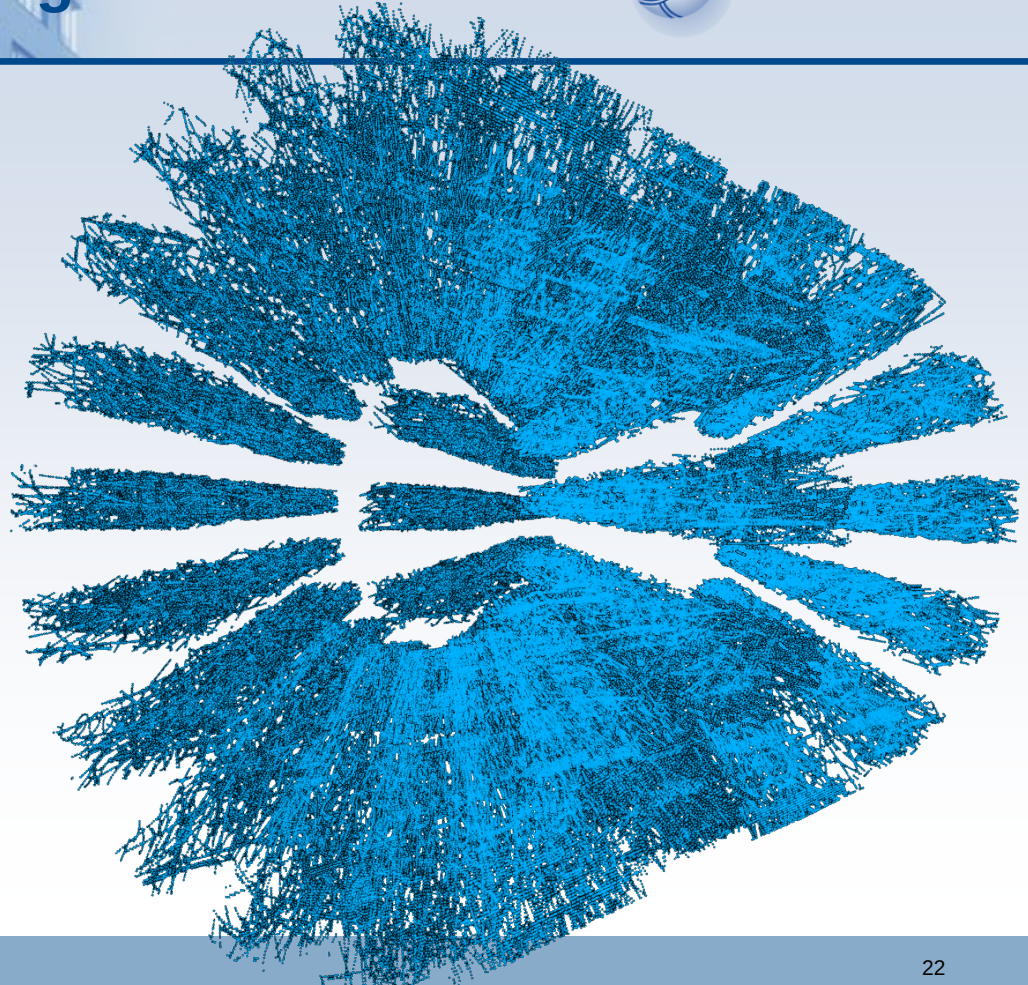
- **2. Phase: Track-Merger**
  - Combines the track segments found in the individual sectors.

Illustration of splitting in sectors

FIAS Frankfurt Institute for Advanced Studies

Sectors of ALICE TPC:





Sector-local seeding can lead to some inefficiencies at sector borders.

Structure of HLT TPC Tracking

FIAS Frankfurt Institute for Advanced Studies

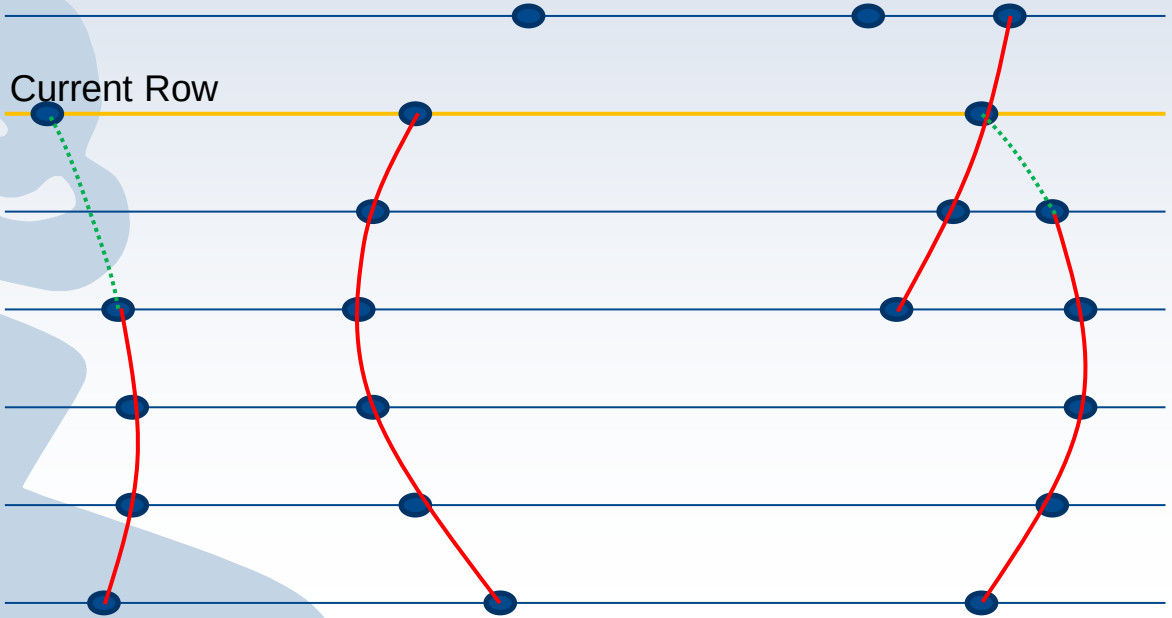
Tracking split in 4 main (abstract) steps.

- Each step is internally split in technical substeps.
- Phase 1 (Steps 1 and 2) on GPU, Phase 2 (Steps 3 and 4) and CPU!

#	Task	How	Locality	Description	Time	Device
1	Seeding	Cellular Automaton	Very Local (hit and adjacent hits)	Find short track candidates of 3 to about 10 clusters.	Ca 30%	GPU or CPU
2	Track following	Kalman Filter (simplified)	Sector-local	Fit parameters to candidate, find full track segment in one sector via track following with simplified Kalman filter (e.g. constant B-field, y and x uncorrelated, …)	Ca 60%	
3	Track Merging	Combinatorics / Mathematics	Global	Merge track segments within a sector and between sectors	Ca 2%	CPU only
4	Track Fit	Kalman Filter (full)	Global	Full track fit with full Kalman filter (polynomial approximation of B-field)	Ca 8%	

Parallel Track Construction

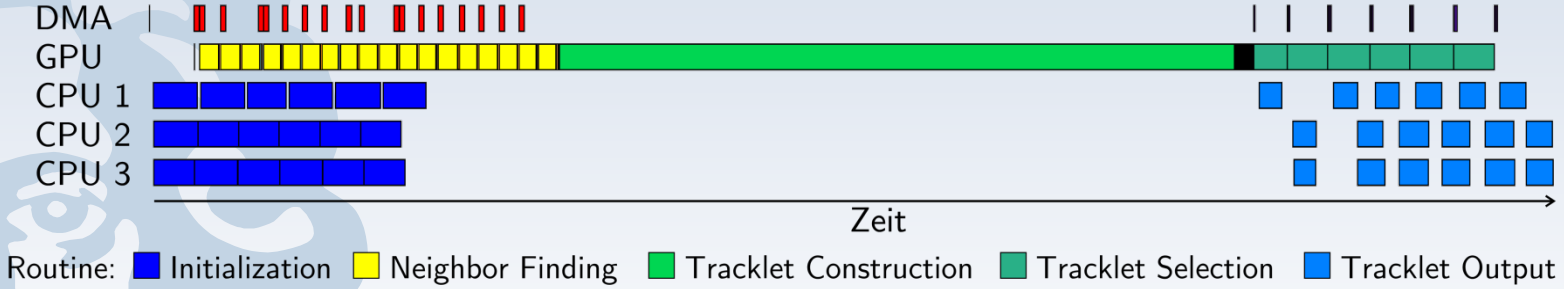
Tracks are independent and can be processed simultaneously
Because of Data Locality the Tracks are processed for a common Row



PERFORMANCE

Optimizations (Performance)

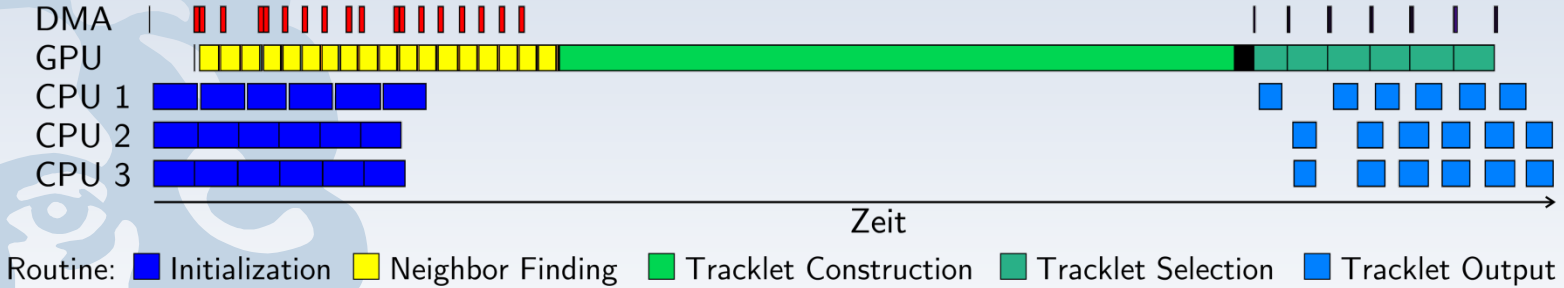
- **Separation of event in sectors enables the use of a pipeline Pipeline:**
  - Tracking on GPU, pre-/postprocessing on CPU, and data transfer run in parallel.



Optimizations (Performance)

FIAS Frankfurt Institute for Advanced Studies

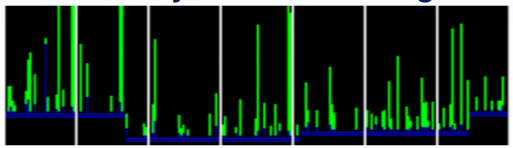
- **Separation of event in sectors enables the use of a pipeline Pipeline:**
  - Tracking on GPU, pre-/postprocessing on CPU, and data transfer run in parallel.



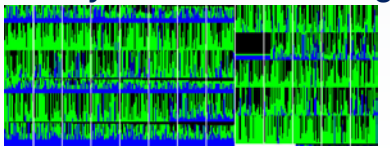
- **Problem: tracks are differently long. Through dynamic scheduling, GPU can be fully used.**

- Black : Idle
- Blue : Track Fit
- Green : Track Extrapolation

No dnymic scheduling



With dynamic scheduling



Common tracker source code

- **CPU and GPU tracker (in CUDA) share 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);
  }
}
```

cuda_wrapper.cpp:

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

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

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

→ **Same source code for CPU and GPU version**

- The macros are used for API-specific keywords only.
- The fraction of common source code is above 90%.

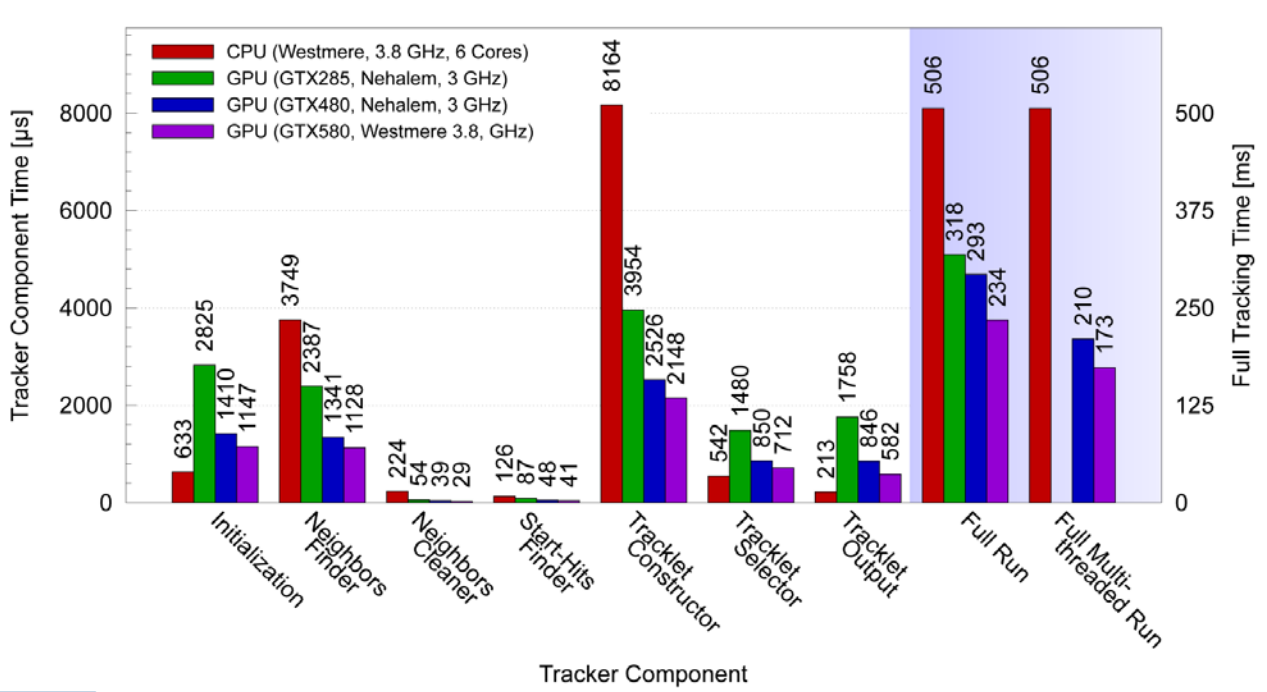
GPU Tracker Performance

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

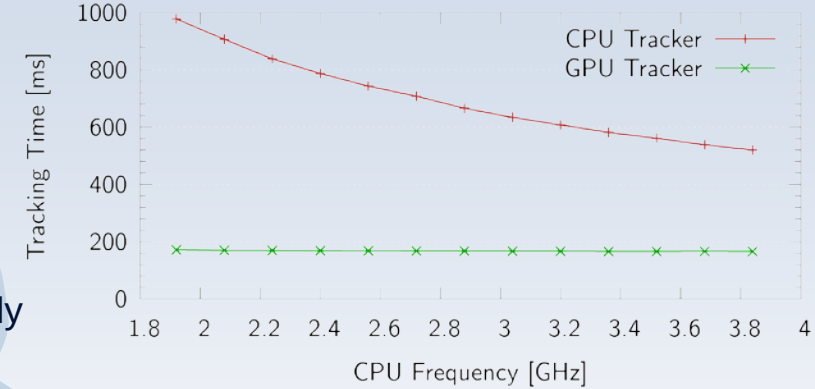
Compute-intense substeps show speedup on GPU.

Some irrelevant substeps run on GPU but without speedup.

Still necessary to avoid data movement.

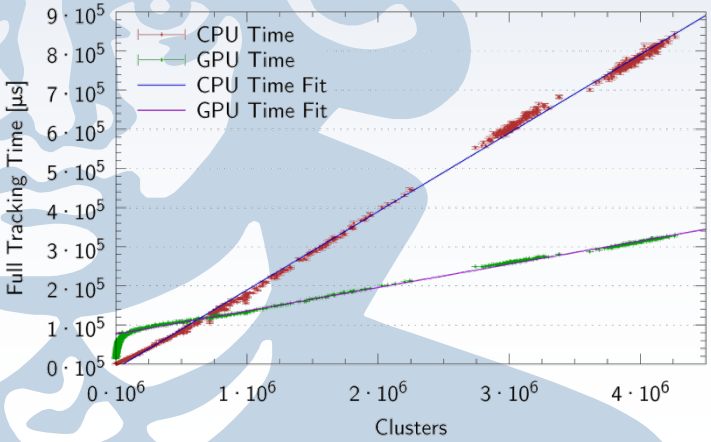


Results

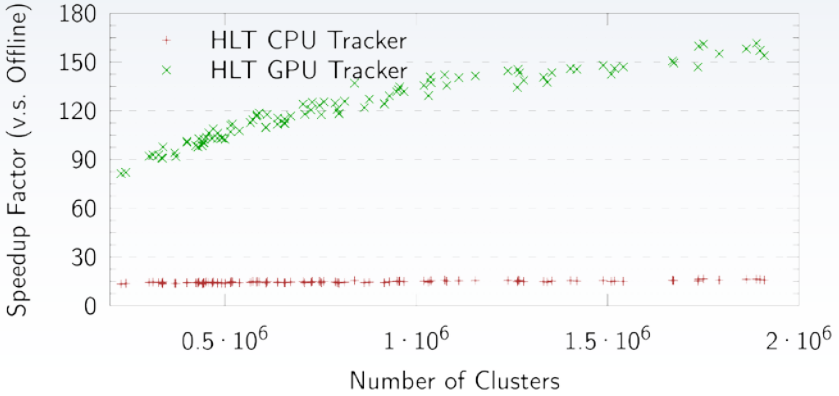


GPU performance independent from CPU.

Tracking times scales linearly with input data size.



Approx. 150 times faster than offline tracking.



Performance on new GPUs

- **Important new GPU feature relevant for GPU tracker:**
  - GPUs can run multiple different kernels in parallel.
  - This can improve GPU utilization. Preliminary tests show 15% improvement already.

- **GPU tracking time on central PbPb event.**

NVIDIA GTX480 (Fermi)	448 shader, 1215 MHz	174 ms	(used in the old HLT)
NVIDIA GTX780 (Kepler)	2304 shader, 863 MHz	155 ms	
NVIDIA Titan (Kepler)	2688 shader, 837 MHz	146 ms	
AMD S9000 (Tahiti)	**1792 shader, 900 MHz**	**145 ms**	(used in the new HLT)
NVIDIA GTX980 (Maxwell)	2048 shader, 1126 MHz	120 ms	

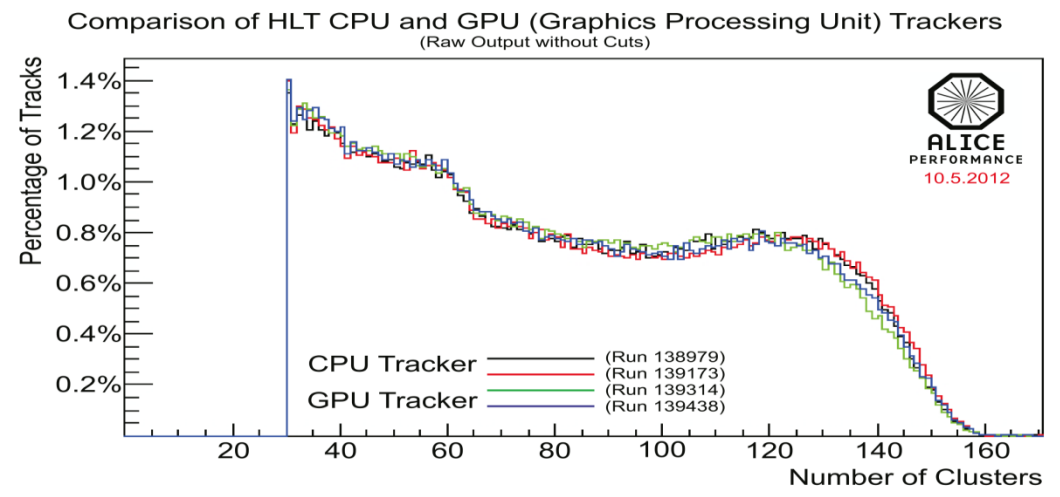
- **With both NVIDIA and AMD as possible vendors, we are no longer vendor-locked!**

- **New GPUs with more shaders not optimally used yet. We assume a speed benefit of up to 30% by further tuning the tracker for the new GPU chips.**

Consistency of Tracking Results

The first runs showed some inconsistencies in cluster to track assignment statistics, but not in physical observables. Three causes were identified:

- Cluster to track assignment
- Track Merger
- Non-associative floating point arithmetic



Consistency of Tracking Results

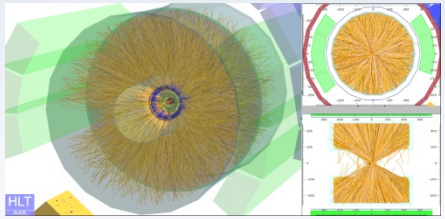
- **Cluster to track assignment**
  - Problem: Cluster to track assignment was depending on the order of the tracks.
    - Each cluster was assigned to the longest possible track. Out of two tracks of the same length, the first one was chosen.
    - Concurrent GPU tracking processes the tracks in an undefined order.
  - Solution: Both the chi² and the track length are used as criteria. It is extremely unlikely that two tracks coincide in both values.
- **Similar problem in track merger, which depended on track order.**

Consistency of Tracking Results

- **Non associative floating point arithmetic**
  - Problem: Different compilers perform the arithmetic 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%

Summary (ALICE Tracking)

- **HLT Tracking 15 times faster than offline tracking.**
- **With GPU additional speedup of 10 compared to CPU → Total speedup 150.**
- **GPU and CPU results consistent and reproducible.**
- **GPU Tracker runs on CUDA, OpenCL, OpenMP – one common shared source code.**
- **Now: 180 compute nodes with GPUs in the HLT**
  - First deployment: 2010 – 64 GPUs in LHC Run 1.
  - Since 2012 in 24/7 operation, no problems yet.



- **Cost savings compared to an approach with traditional CPUs:**
  - About 500.000 US dollar during ALICE Run I.
  - Above 1.000.000 US dollar during Run II.
  - Mandatory for future experiments, e.g.. CBM (FAIR, GSI) with >1TB/s data rate.

Strategies for GPU optimization

- **Nearly all GPU programs I implemented use a pipeline and asynchronous processing:**
  - Concurrent GPU processing, CPU processing, DMA transfer.

- **A shared common source code for CPU and GPU version improves**
  - Verification
  - Debugging
  - Maintainability

- **Sometimes the algorithm can be changed slightly to neglect concurrency effects**
  - Improves consistency and reproducibility

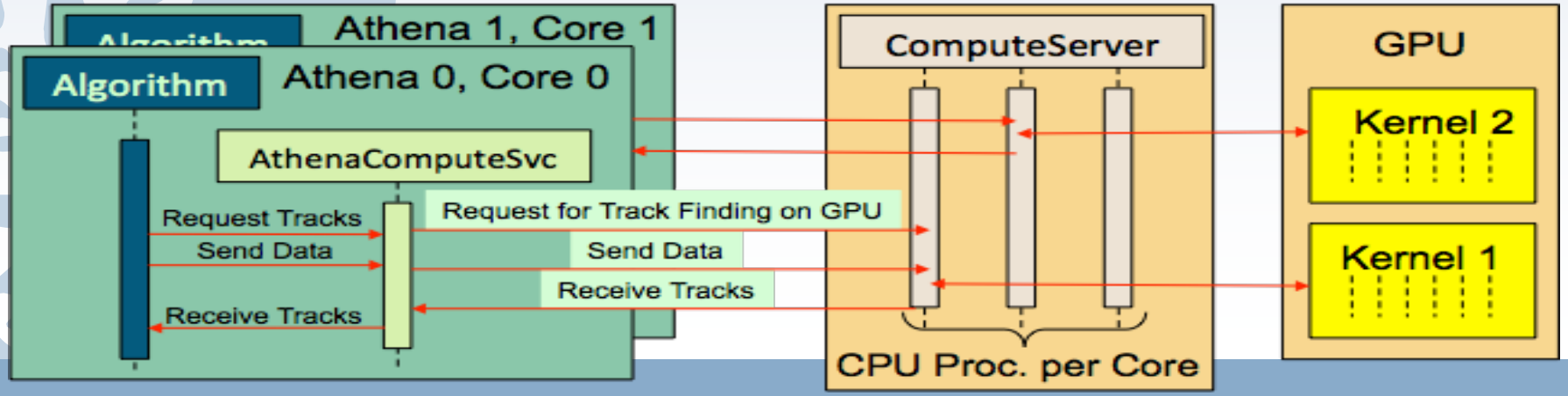
- **Every problem might require custom optimization:**
  - Like track length variation in ALICE TPC

FIAS Frankfurt Institute for Advanced Studies

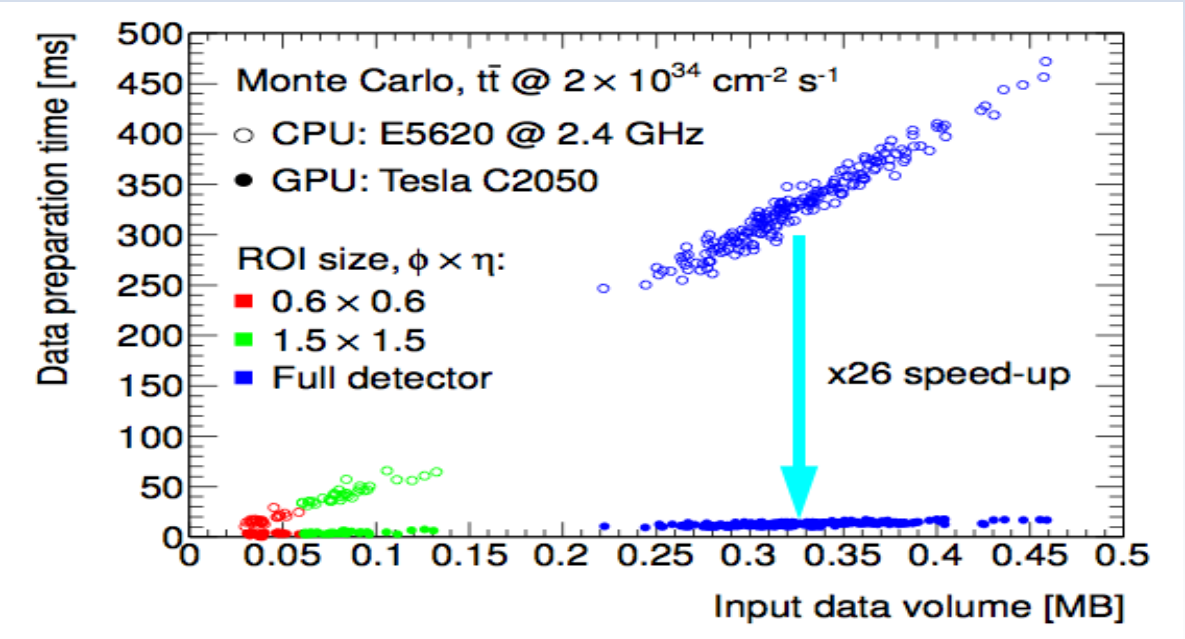
A BRIEF LOOK AT OTHER EXPERIMENTS

ATLAS – Client-Server-Architecture

- **Client-server architecture allows GPU resources to be shared amongst multiple trigger instances Data transfer is done over shared memory segment**
  - Also used as CUDA host buffer
- **Minimizes integration surface in trigger software - only POSIX required**
- **Allows for GPU memory resources (e.g. hardware maps) to be shared**



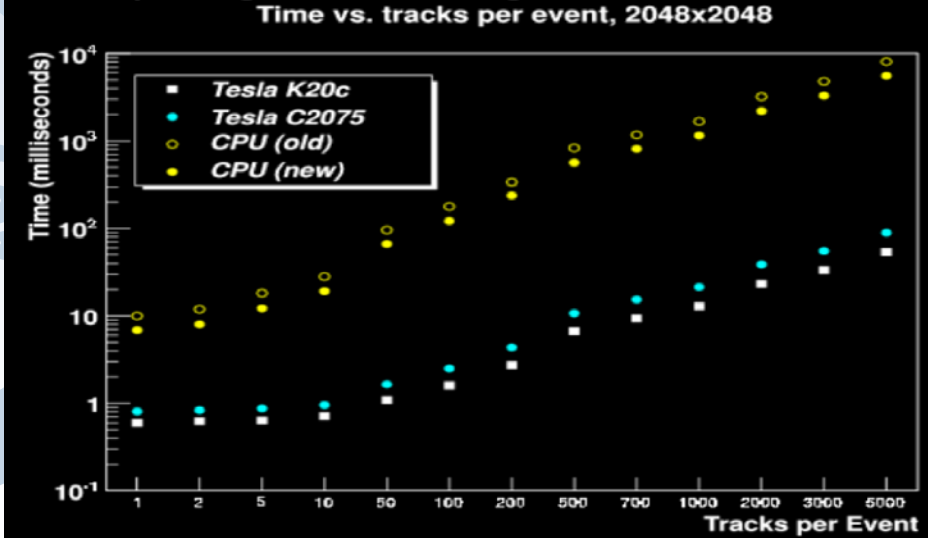
ATLAS – Data Preparation Results



Bytestream decoding and clustering show a **26x** speed-up against single-threaded CPU

CMS – GPU Implementation

- Hough transform is a **natural candidate** for GPU acceleration using general-purpose GPU programming with CUDA.



CPU implementation before (open) and after (filled) optimization (performed on Intel Core i7-3770)

GPU implementation on Tesla C2075 and K20c
–10-60x faster!

- Also a candidate for investigating with Xeon Phi

See **arXiv:1309.6275** for more on these implementations

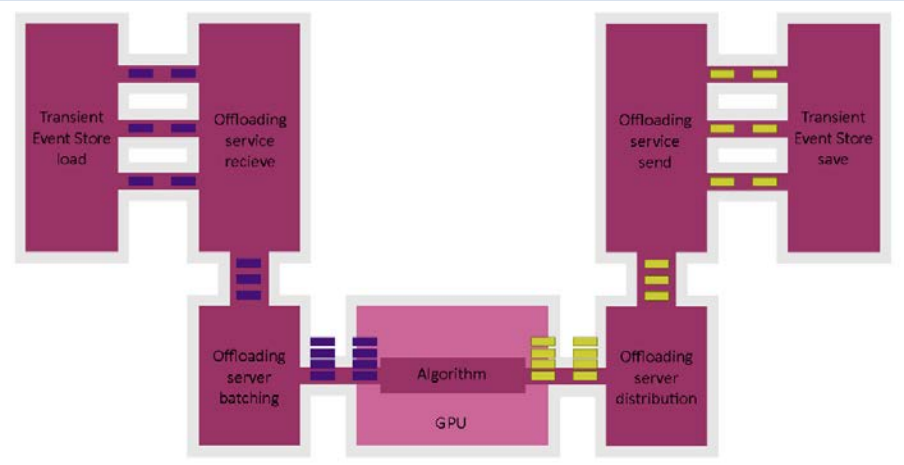
LHCb – GPU Manager Offload Tool

Socket client-server tranmission

Scheduler First-Come First-Served, gathers multiple events and ships them for concurrent processing

Some goodies

- Algorithm exceptions propagated to callers
- Centralized profiling, logging
- File input / output configurable
- Outside framework execution possible



Analysis

- **Clustering and Tracking are dominant applications at LHC experiments.**
  - ALICE has deployed FPGA based cluster finding and GPU based tracking.
  - ATLAS, CMS, LHCb are investigating solutions.

- **ALICE has isolated hotspot for GPU: TPC track finding**
  - ALICE employs a custom solution only for tracking.

- **ATLAS does not have such an isolated hotspot, but many more software components that contribute equally.**

- **ATLAS and LHCb hence use a slightly more general approach, with GPU Manager Offload Engine / Client-Server architecture.**
  - This enables easier offloading of multiple different and smaller applications.

Summary (Strategy Ideas)

- **Find hotspot, port hotspot to GPU**
  - If several hotspot problems are tightly coupled, bring them to GPU as one instance without transfers:
    - For instance, ALICE TPC track finding, track fit, ITS tracking.
  - Small intermediate steps can be run on GPU without optimization effort (no speedup but less transfers).
  - Try to optimize for CPU first, do not bring applications to GPU if not needed.
- **Multi-GPU usage can be facilitated easily via event-paralelism (perfect scaling at ALICE).**
  - Two processes can also run two events on the same GPU (more memory needed, 10% speedup)
- **If there is no single hotspot, many subprograms must be ported to GPU for reasonable speedup.**
  - Some client-server offload approach can be used to run various problems in parallel.
- **If DMA transfer and CPU pre-/postprocessing is involved, a pipeline should be used.**
- **A common source code for CPU and GPU simplifies the development significantly.**
- **Expect a speedup of around 3 for GPU versus full processor.**
- **Employ an algorithm that allows parallelization, and do not run into a memory wall.**
  - For instance, 100 MB map of inhomogeneous magnetic field → memory access will dominate everything.