



Application of GPUs in ATLAS offline tracking and Introduction to dOpenCL and SkelCL

Sebastian Fleischmann¹
in collaboration with
Rene Böing^{1,2}, Maik Dankel^{1,2}, Peter Mättig¹

¹University of Wuppertal

²University of Applied Sciences Münster

2nd April 2014

Annual Concurrency Forum Meeting



Fachhochschule
Münster University of
Applied Sciences



- ▶ Report on studies of tracking algorithms using GPU/MIC
- ▶ Two main parts of this talk:

- 1 Kalman Filter with reference trajectory on GPU
 - GPU comparison
 - Multi Track Fitter
- 2 GPUs/MIC integration in ATLAS software framework
 - Resource management
 - dOpenCL
 - SkelCL
- 3 Summary



Introduction

Kalman Filter in track reconstruction

GPUs in
ATLAS
tracking

Sebastian
Fleischmann

Introduction

Kalman Filter
on GPU

GPU
comparison
MTF

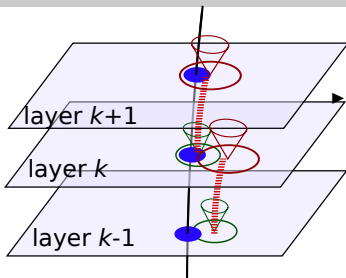
GPU in
Athena

Summary

Backup

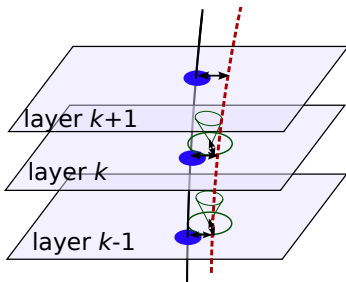


p. 3



Default Kalman filter implementation in ATLAS: Extended KF

- ▶ Measurement updates alternate with extrapolations



▶ Alternative: KF with reference trajectory

- ▶ Reference extrapolated through whole volume
- ▶ Fitter runs only on differences between measurements and reference trajectory
- ▶ Initial parameters for reference trajectory must not be too far away from final fit (esp. passed material)
- ▶ More stable in case of outliers
- ▶ Allows for separating extrapolation from actual fit (and measurement assignments): Ideal for offloading to accelerator devices

“Conventional” CPU implementation

GPUs in
ATLAS
tracking

Sebastian
Fleischmann

Introduction

Kalman Filter
on GPU

GPU
comparison
MTF

GPU in
Athena

Summary

Backup

- ▶ Reference method implemented in C++ CPU code in ATLAS software framework using eigen library
- ▶ Validation wrt. existing implementation of the Kalman Filter
 - ▶ Better CPU timing observed due to saving of extrapolations in case of outlier detection
- ▶ Basis for cross checks of GPU implementation
- ▶ Stand-alone CPU code implemented separately from ATLAS software framework (ATHENA) for GPU tests
 - ▶ Extrapolations done inside ATHENA and written to private storage
 - ▶ Allows for running same task using OpenMP, CUDA and OpenCL



Results of GPU implementation

GPUs in
ATLAS
tracking

Sebastian
Fleischmann

Introduction

Kalman Filter
on GPU

GPU
comparison

MTF

GPU in
Athena

Summary

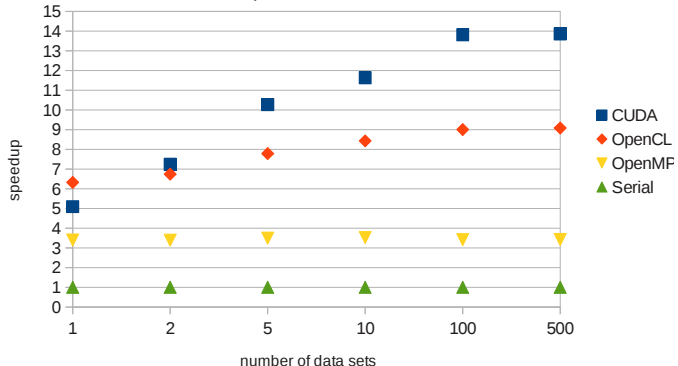
Backup



p. 5

Runtime of stand-alone (non-ATHENA) code compared to OpenCL and CUDA on NVIDIA GPU

- ▶ CPU-code uses same flat data structures as GPU codes
- ▶ OpenMP version runs multi-threaded on CPU (8 threads)
- ▶ Using 5×5 GPU threads for each track (corresponding to entries in covariance matrix)



- ▶ Significant speed-up observed ($13.8\times$ compared to serial)

- ▶ Different mathematical implementations of the Kalman filter compared to each other
- ▶ Tried approximation of covariance matrix of track parameters to do inversion of 4×4 matrix instead of 5×5 .
 - ▶ Leads to slightly worse residuals, not considered further
- ▶ Tests using Intel Many Integrated Core architecture (Xeon Phi)

Summary

- ▶ Kalman Filter for track fits using reference trajectory available in various implementations
- ▶ Ideal for offloading, because split of dependency on magnetic field and geometry
- ▶ Physics performance comparable to default Kalman Filter, even slightly more stable numerically
- ▶ Speed improvement observed by using GPUs, only small differences between OpenCL and CUDA



Outlook: Multi Track Fitter

Alternative approach for processing of track seeds

GPUs in
ATLAS
tracking

Sebastian
Fleischmann

Introduction

Kalman Filter
on GPU

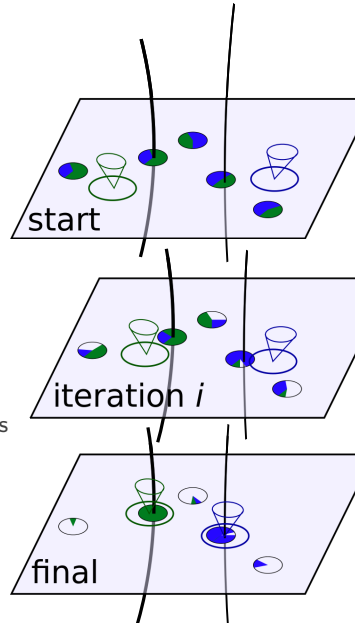
GPU
comparison
MTF

GPU in
Athena

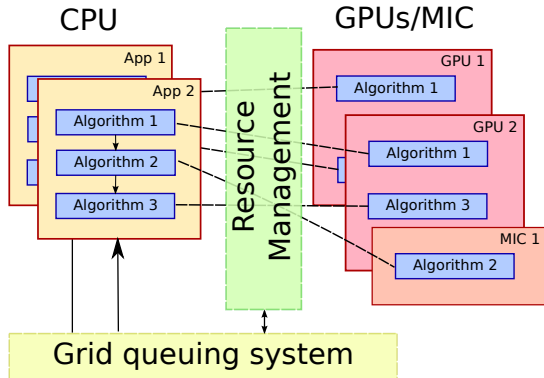
Summary

Backup

- ▶ Track seeding from hit triplets creates duplicate and ghost tracks
 - ▶ typically sharing large number of measurements between tracks
- ▶ “Ambiguity processing” tries to solve this by
 - ▶ Scoring tracks and removal of shared measurements
 - ▶ Iteratively removing hits and refits
- ▶ Alternative: Multi Track Fitter
 - ▶ Fit several tracks at once and dynamically adjust hit assignments
 - ▶ Hit assignments are probabilistic
 - ▶ Fixed number of (annealing) iterations
 - ▶ Better for parallelisation than current ambiguity processing
 - ▶ Physics performance of ATHENA implementation currently being optimised
 - ▶ GPU implementation ongoing



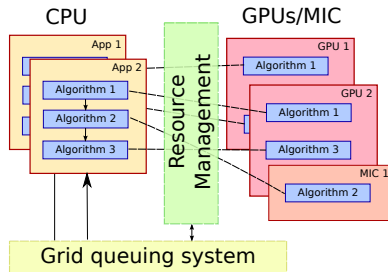
- ▶ Application of GPUs/MICs in the reconstruction chain needs resource management
 - ▶ Different processes on host CPU
 - ▶ Heterogeneous hardware



- ▶ Compare different solutions and new ones concerning flexibility and overhead
- ⇒ Develop framework transparent to users



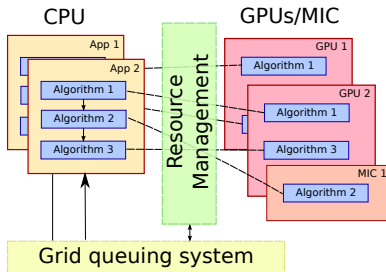
- ▶ Problems in direct usage of GPUs/MIC inside ATLAS software framework due to
 - ▶ specialised compiler
 - ▶ dependencies on hardware specific libraries
 - ▶ handling of multiple ATHENA processes and sharing of GPUs
- ▶ Client-server architecture being investigated
 - ▶ Existing approach by Dmitry Emelianov and Sami Kama for high-level trigger
 - ▶ Currently testing and extending this approach for offline reconstruction
- ▶ Investigating alternatives





Open questions to be addressed for a resource manager

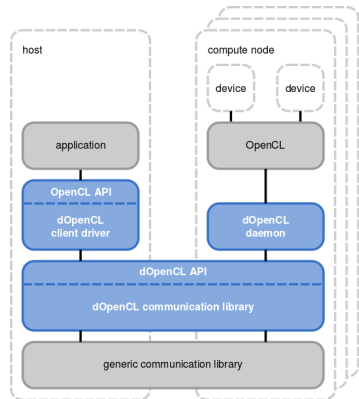
- ▶ Integration as “algorithm” or “service” inside ATHENA
- ▶ Handling of static data on accelerator devices, like geometry or magnetic field data
- ▶ Overhead due to data transfers
- ▶ Integration into grid infrastructure
- ▶ Dynamic way of decision when to offload algorithms
- ▶ Future integration in AtlasHive or similar:
 - ▶ Can one create one device request per CPU thread?





- ▶ General solution for heterogeneous environments developed by P. Kegel, M. Steuwer, S. Gorlatch (University of Münster): **distributed OpenCL** ("dOpenCL")

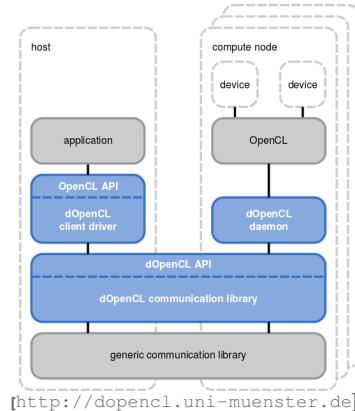
- ▶ Addresses several problems:
 - ▶ Usage of heterogeneous multi-core devices, like multi-core CPUs, GPUs, Intel MIC
 - ▶ Handling of multiple devices
 - ▶ Multi-user access to devices
 - ▶ Abstraction of distributed system from client



[<http://dopencl.uni-muenster.de>]

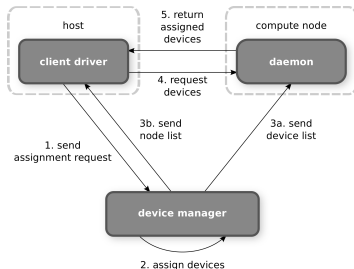
- ▶ Implemented as client-server approach using “Generic Communication Framework” (GCF) for (network) communication
 - ▶ Each server can handle arbitrary number of local accelerator devices (also CPUs)
 - ▶ Extension of OpenCL API
 - ▶ Existing OpenCL programs can directly make use of distributed systems

- ▶ Client driver handles memory transfers and consistency of memory between client and compute nodes





- ▶ Client processes can request computing resources from (central) device manager
 - ▶ Device manager handles concurrent requests
 - ▶ Devices can be dynamically added to device manager



[P. Kegel, S. Gorlatch, "dOpenCL: Towards Uniform Programming for Distributed Systems with Multi-Cores and GPUs"]

- ▶ Authors of dOpenCL quote only small overhead due to dynamic handling compared to using OpenCL+MPI
- ▶ Data transfer via network can cause significant overhead for memory-bound problems (e.g. adding vectors), ok for compute-bound problems (e.g. multiplication of large matrices)
- ▶ Authors present results for application from medical imaging and standard benchmark algorithms

Currently testing this approach for ATLAS using test setup of Kalman filter

- ▶ OpenCL is very useful to write parallel code for different accelerator platforms
- ▶ However, rather clumsy code needed for handling data transfers to and from devices and queuing of kernels
- ▶ **SkelCL** provides skeletons of recurring operations: `Map`, `Zip`, `Reduce`, ...
- ▶ User has to write only functions for single data item
- ▶ Example: Dot product $\sum_{k=1}^n a_k \cdot b_k = \text{reduce}(+)(\text{zip}(\cdot)(a, b))$

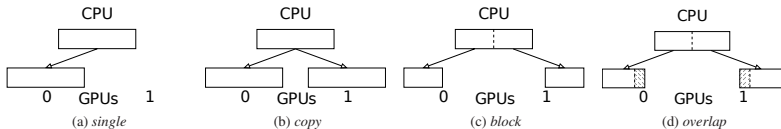
```

1 SkelCL::init() ; // initialize SkelCL
2 /* create skeletons */
3 Reduce <float> sum (
4     "float func(float x, float y) {return x + y;}");
5 Zip <float> mult (
6     "float func(float x, float y) {return x * y;}");
7 /* create input vectors */
8 Vector <float> A(SIZE); fillVector(A);
9 Vector <float> B(SIZE); fillVector(B);
10 /* execute skeletons */
11 Vector <float> C = sum( mult(A, B) );
12 /* print result of dot product */
13 cout << " Result : " << C.front() << endl;

```



- ▶ SkelCL provides vector and matrix containers on which operations are performed in parallel
- ▶ Data transfers are hidden from the user
- ▶ Vectors and matrices can automatically be shared or split between devices (GPUs)



[M. Steuer, S. Gorlatch, "High-Level Programming for Medical Imaging on Multi-GPU Systems using the SkelCL Library"]

- ▶ Authors of SkelCL quote performance very similar to optimised OpenCL code for example problems
- ▶ Integrates nicely with dOpenCL to support multi-users and distributed devices
- ▶ Next steps: Implementation of simple exemplary algorithms as student projects
 - ▶ Testing demands of HEP problems and compatibility with SkelCL



- ▶ Kalman Fitter with reference trajectory tested as
 - ▶ CPU-based version using full ATLAS Tracking event data model
 - ▶ stand-alone serial code using flat data structures
 - ▶ GPU implementation in OpenCL and CUDA
 - ▶ MIC tests ongoing
 - ▶ Extension ongoing for parallel treatment of ambiguities of track seeds
- ▶ Started project to investigate options of integrating accelerator devices inside ATLAS software framework
 - ▶ device/resource management
 - ▶ data transfer
 - ▶ handling of detector geometry and magnetic field data
 - ▶ grid integration
 - ▶ mid-term goal: Provide full chain test setup of (part of) offline software on grid-enabled cluster
- ▶ “distributed OpenCL” may be nice solution for accessing accelerator cards from different CPU processes
- ▶ SkelCL may be useful to write code based on OpenCL which is easier to understand and maintain



**GPUs in
ATLAS
tracking**

**Sebastian
Fleischmann**

Introduction

Kalman Filter
on GPU

GPU in
Athena

Summary

Backup



- ▶ Test system:
 - ▶ CPU: Intel Xeon E5-1620 3.60GHz
 - ▶ GPU: ZOTAC GeForce GTX TITAN 6 GB
 - ▶ about 900 EUR
- ▶ Multiple threads on host CPU needed to fully exploit GPU
- ▶ Factor 14 achieved wrt. serial standalone code using single thread
- ▶ Fair comparison to multi-threaded CPU code (based on OpenMP and GSL), including data transfer to/from device:
 - ▶ Factor 4.1 for CUDA
 - ▶ Factor 2.7 for OpenCL

