Track and Vertex Reconstruction on GPUs for the Mu3e Experiment

Dorothea vom Bruch
for the Mu3e Collaboration

GPU Computing in High Energy Physics, Pisa

September 11th, 2014

Physikalisches Institut Heidelberg
The Mu3e experiment
Readout and event selection
Track fit on the GPU
Current performance
Outlook
The Mu3e experiment searches for...

... the charged lepton-flavour violating decay $\mu \rightarrow e^+ e^+ e^-$ with a sensitivity better than $10^{-16}$

- Suppressed in Standard Model to below $10^{-54}$
- Any hint of a signal indicates new physics:
  - Supersymmetry
  - Grand unified models
  - Extended Higgs sector
  - ...
- Current limit on branching ratio: $10^{-12}$ (SINDRUM)
Signal versus Background

Signal
- Coincident in time
- Single vertex
- $\Sigma \vec{p}_i = 0$
- $E = m_{\mu}$

Combinatorial background
- Not coincident in time
- No single vertex
- $E = m_{\mu}$
- $\Sigma \vec{p}_i \neq 0$

Internal conversion background
- Coincident in time
- Single vertex
- $E \neq m_{\mu}$
- $\Sigma \vec{p}_i \neq 0$
Signal versus Background

Signal
- Coincident in time
- Single vertex
- $\Sigma \vec{p}_i = 0$
- $E = m_\mu$

Combinatorial background
- Not coincident in time
- No single vertex
- $E \neq m_\mu$
- $\Sigma \vec{p}_i \neq 0$
**Signal**
- Coincident in time
- Single vertex
- $\sum \vec{p}_i = 0$
- $E = m_\mu$

**Combinatorial background**
- Not coincident in time
- No single vertex
- $E = m_\mu$
- $\sum \vec{p}_i \neq 0$

**Internal conversion background**
- Coincident in time
- Single vertex
- $E \neq m_\mu$
- $\sum \vec{p}_i \neq 0$
Resolution

- $\mu$ decays at rest $\rightarrow p_e < 53 \text{ MeV}/c$
- Resolution dominated by multiple Coulomb scattering ($\propto 1/p$)

Minimize material
- High Voltage Monolithic Active Pixel Sensors thinned to 50 $\mu$m
- Ultralight mechanics
Detector Requirements

- Excellent momentum resolution: < 0.5 MeV/c
- Good timing resolution: 100 ps
- Good vertex resolution: 100 µm

The Detector
The Detector

GPUs in the Mu3e Experiment

Sep 11, 2014

Dorothea vom Bruch
The Detector

μ Beam

Inner pixel layers

Target

Sep 11, 2014

GPUs in the Mu3e Experiment

Dorothea vom Bruch
The Detector

Target

Inner pixel layers

μ Beam

Outer pixel layers
The Detector

[Diagram showing the detector setup with inner and outer pixel layers, scintillating fibres, and a target hit by a muon beam.

Sep 11, 2014  GPUs in the Mu3e Experiment  Dorothea vom Bruch
The Detector

Recurl pixel layers

Inner pixel layers

Target

μ Beam

Scintillating fibres

Outer pixel layers
The Detector

- Recurl pixel layers
- Scintillator tiles
- Inner pixel layers
- Target
- Scintillating fibres
- Outer pixel layers
- Outer pixel layers

μ Beam
- Beam provided by the Paul Scherrer Institut
- Currently: $10^8 \mu/s$
- In future: Up to $2 \times 10^9 \mu/s$
- Triggerless readout
- 1 Tbit/s data rate
- Online selection
  → Reduction by factor 1000

Sep 11, 2014
GPUs in the Mu3e Experiment
Dorothea vom Bruch
Readout Scheme

- **HVMAPS**
- **Pixel Sensors**
- **1116 Pixel Sensors**
- **FPGA**
- **38 FPGAs**
- **2 RO Boards**
- **12 PCs**
- **GPU**
- **Data Collection Server**
- **Mass Storage**

- Up to 45 800 Mbits/s links
- 1 6.4 Gbit/s link each
- 12 6.4 Gbits/s links per RO Board
- Gbit Ethernet
Readout Scheme

- HVMAPS
- FPGA
- 1116 Pixel Sensors
- FPGA
- FPGA
- 38 FPGAs
- FPGA
- 2 RO Boards
- 12 6.4 Gbits/s links per RO Board
- 1 6.4 Gbit/s link each
- 45 800 Mbits/s links
- Gbit Ethernet
- Data Collection Server
- Mass Storage
- 12 PCs
- GPUs in the Mu3e Experiment

Sep 11, 2014

Dorothea vom Bruch
- GPU gets 50 ns time slice
- Full detector information
- Find 3 tracks originating from common vertex
Multiple Scattering Fit

- Ignore spatial uncertainty
- Multiple scattering at middle hit of triplet
- Minimize multiple scattering $\chi^2$

Minimize $\chi^2 = \frac{\phi_{MS}^2}{\sigma_{MS}^2} + \frac{\Theta_{MS}^2}{\sigma_{MS}^2}$
Multiple Scattering Fit

- Describe track as sequence of hit triplets
- Non-iterative fit
GPU Specifications

- Use Nvidia’s CUDA environment
- GeForce GTX 680
- 8 Streaming Multiprocessors

Image source: http://www.pcmag.com/article2/0,2817,2401953,00.asp
Consider first three detector layers

Number of possible track candidates \( \sim n[1] \times n[2] \times n[3] \)

\( n[i] \): # hits in layer \( i \)

On GPU: Loop over all possible combinations
  - Geometrical selection cuts
  - Triplet fit
  - Vertex fit

⇒ Goal: Reduction factor of \( \sim 1000 \)
Sharing the Work

- **On FPGA:**
  - Sort hits
  - Copy hit arrays to global memory of GPU
- **Currently:** FPGA tasks are performed by CPU
- **Within kernel / thread:**
  - Apply geometrical selection cuts:
    For pairs of hits in layers [1,2] and [2,3] check proximity in x-y plane and in z
  - Do triplet fit
  - Cut on $\chi^2$ and fit completion status
  - If all cuts passed: Count triplets and save hits in global memory using atomic function
- Copy back global index array
Within kernel / thread:

- Apply geometrical selection cuts: For pairs of hits in layers [1,2] and [2,3] check proximity in x-y plane and in z
- Do triplet fit
- Cut on $\chi^2$ and fit completion status
- If all cuts passed: Count triplets and save hits in global memory using atomic function
Within kernel / thread:

- Apply geometrical selection cuts: For pairs of hits in layers [1,2] and [2,3] check proximity in x-y plane and in z
- Do triplet fit
- Cut on $\chi^2$ and fit completion status
- If all cuts passed: Count triplets and save hits in global memory using atomic function
Grid Alternatives: One Fit per Thread

grid dimension $x = n[1]$

grid dimension $y = n[2]$

block dimension $x = n[3]$

Block (0,0)  Block (0,1)  ...  Block (0,n[1])

Block (1,0)  Block (1,1)  ...  Block (1,n[1])

...  ...  ...  ...

Block (n[2],0)  Block (n[2],1)  ...  Block (n[2],n[1])

Thread (0,0)  Thread (0,1)  ...  Thread (0,n[3])

Sep 11, 2014  GPUs in the Mu3e Experiment  Dorothea vom Bruch  17
Grid Alternatives: Several Fits per Thread

grid dimension $x = n[1]$

Loop over $n[3]$ hits

block dimension $x = n[2]$

Sep 11, 2014

GPUs in the Mu3e Experiment

Dorothea vom Bruch
Separate Kernels

Launch grid with all possible hit combinations

Apply selection cuts

Store indices of selected triplets

grid dimension \( N = \frac{\text{# selected triplets}}{128} \)

block dimension \( x = 128 \) (or other multiple of 32)

Advantages

No idle threads in time-intensive fitting kernel

Block dimension: Multiple of 32 (warp size)
Kernel Profile

One kernel version

Selection Cuts

Fit

87 % branch divergence during fit procedure

Not passed

Separate kernel version

Selection Cuts

branch divergence in first kernel

Fit

No divergence during fit

⇒ Choose separate kernel version
<table>
<thead>
<tr>
<th>Idea</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count triplets by using atomicInc on shared variable</td>
<td>Synchronization of threads and copying back to global memory takes too long</td>
</tr>
<tr>
<td>Compose grid of only one block and n[1] threads; load hit arrays into shared memory for quicker access</td>
<td>Amount of shared memory per Streaming Multiprocessor not enough</td>
</tr>
<tr>
<td></td>
<td># of blocks too small to hide latency</td>
</tr>
</tbody>
</table>
## Current Performance

<table>
<thead>
<tr>
<th></th>
<th>One kernel</th>
<th>Separate kernels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wall time of CPU &amp; GPU</td>
<td>$7.6 \times 10^9$ triplets/s</td>
<td>$1.4 \times 10^{10}$ triplets/s</td>
</tr>
<tr>
<td>Run time measured over</td>
<td>$&gt; 11$ days</td>
<td>$&gt; 15$ hours</td>
</tr>
</tbody>
</table>

- Most time spent on selection cuts
- Can be improved by using FPGA for selection
- Currently: Fit performed on CPU and GPU to compare output → Contributes to computation time
Summary

- Searching for $\mu \rightarrow e^+ e^+ e^-$ with a sensitivity better than $10^{-16}$
- Goal: Find $2 \times 10^9$ tracks/s online
- Achieved: Process $10^{10}$ triplets/s
Outlook

- Include vertex fit or alternative vertex selection criteria
- Outsource pre-fit selection to FPGA
- Write data to GPU via Direct Memory Access from FPGA
Thank you for your attention!
Backup Slides
More Detailed Performance for Separate Kernels

Wall time of CPU & GPU without fit on CPU | $1.4 \times 10^{10}$ triplets/s

<table>
<thead>
<tr>
<th>GPU time only</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time per fit</td>
<td>26 µs</td>
</tr>
<tr>
<td>Average time for fit &amp; memory copying</td>
<td>30 µs</td>
</tr>
<tr>
<td>Fit &amp; copying</td>
<td>$1.1 \times 10^7$ fits/s $^1$</td>
</tr>
</tbody>
</table>

$^1$Time measured by nvprof, includes profiling overhead.
Multiple Scattering

\[ \Omega \sim \pi \]

\[ \theta_{MS} \]

\[ B \]

Sep 11, 2014

GPUs in the Mu3e Experiment

Dorothea vom Bruch
Kernel profile for one selected kernel

87% thread divergence
Kernel profile of one selected fitting kernel (without selection kernel):

No thread divergence when fit is called
CPU - GPU Communication

- Nvidia: API extension to C: CUDA (Compute Unified Device Architecture)
- Compile with nvcc and gcc → runs on host (= CPU) and device (= GPU)
- Very similar to C / C++ code
- Compatible with other languages

Diagram:
- Host
  - Memory
  - Host code
  - Allocate
  - Launch kernel
- Device
  - DRAM
  - GPU
  - Streaming Multiprocessor (SM)
  - Copy back
  - Allocate
- Host
  - Memory
  - Allocate
  - Copy back
  - Host code
  - Launch kernel
Host code

Some CPU code
...
GPU function (kernel) launched as grid on GPU
...
Some more CPU code

CUDA Grid

Grid: Consists of blocks
Block: Consists of threads

CUDA: special variables / functions introduced for
- Identification of GPU code
- Allocation of GPU memory
- Access to grid size
- Options for grid launch
- ...

Sep 11, 2014
GPUs in the Mu3e Experiment
Dorothea vom Bruch
CUDA Architecture (GTX 680 as example)

- One kernel per thread
- Up to 3 dimensions for block and thread indices
- Up to 1024 threads per block
- Max dimension of grid: 65535 x 65535 x 65535
- Access to thread & block index via built-in variable within kernel
Hardware Implementation (GTX 680 as example)

- All threads in grid execute same kernel
- Execution order of blocks is arbitrary
- Scheduled on Streaming Multiprocessors (SMs) according to
  - Resource usage: memory, registers
  - Thread number limit

Limitations:
- Max. 2048 threads per SM
- Limits # blocks per SM
Hardware Implementation: Warps

- After block is assigned to SM
  → Division into units called warps
- On GTX 680: 1 warp = 32 threads
Warp Scheduling

Warps execute

In SIMD fashion (Single Instruction, Multiple Data)
Not ordered

1 warp = 32 threads

SM instruction scheduler

warp 22, instruction 13
warp 13, instruction 4
warp 22, instruction 14
warp 96, instruction 33
warp 13, instruction 5
GPU Memory

**Block 0**
- 48 kB Shared Memory
- Registers
- Thread 0
- Thread 1

**Block 1**
- 48 kB Shared Memory
- Registers
- Thread 0
- Thread 1

**Host**
- 4 GB Global Memory
- 64 kB Constant Memory

- Extremely fast, highly parallel
- Fastest, limited to 65536 registers per block
- High access latency (400 - 800 cycles), finite access bandwidth
- Read only, short latency
Memory Access

Coalesced memory access

Non-coalesced memory access

Warp Memory Access

128 bytes in single transaction