EVALUATION OF GPUS FOR HIGH-LEVEL TRIGGERS IN HIGH ENERGY PHYSICS

Presenter: Hannes Mohr - KIT, IPE


29TH SEPTEMBER, TWEPP 2016
OUTLINE

- Implement track trigger using GPUs
- Use established methods for seeding
- Present our own version of the Hough transformation
- Compare different GPUs/vendors
- Investigate data transfer/latencies
- Estimate impact of technological advances

OUR GOAL IS TO ACHIEVE COMPETITIVE RESULTS, WHILE GAINING FLEXIBILITY
CMS DETECTOR

*Image CMS Collaboration*
BASELINE GEOMETRY - 6 LAYERS 5 DISKS

SILICON TRACKER

*Image CMS Collaboration
UPGRADE TIMELINE

- Peak luminosity
- Integrated luminosity

Luminosity [cm$^{-2}$s$^{-1}$]

Year

LS1

LS2

LS3

300~350 fb$^{-1}$

*Image CMS Collaboration
Current CMS trigger won’t be able to handle:

- Increased data rates
- Increased pile-up

Currently proposed solution:

- Data reduction on detector
- Raise latency of trigger from 3.4 to 12.5 μs
- L1 track trigger
Readout at 40 MHz, BX every 25 ns

6 μs each for L1 Trigger and Global Trigger

L1 Tracking to combine Track seeding and Fitting
STUB BUILDING

- Applies momentum cut to hits
- Delivers estimate on track bend
- Drastically decreases number of hits by a factor of 100
CURRENT APPROACHES

▸ Associative Memory approach (ASICs)
▸ Time-multiplexed FPGA Hough transformation
▸ ...

CURRENT APPROACHES USE SPECIALIZED HARDWARE
## COMPARISON GPU VS. FPGA

Nvidia Tesla K40c vs. XILINX VIRTEX-7 XC7VX1140T (both 28nm)

<table>
<thead>
<tr>
<th>Metric</th>
<th>GPU</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O BW [GB/S]</td>
<td>16</td>
<td>348</td>
</tr>
<tr>
<td>MEMORY BANDWIDTH [GB/S]</td>
<td>288</td>
<td>1.8</td>
</tr>
<tr>
<td>GFLOPS*</td>
<td>5040</td>
<td>850</td>
</tr>
<tr>
<td>GFLOPS/WATT*</td>
<td>21.44</td>
<td>20.8</td>
</tr>
<tr>
<td>GFLOPS/$*</td>
<td>1</td>
<td>0.24</td>
</tr>
</tbody>
</table>

*SINGLE PRECISION, FLOATING POINT
<table>
<thead>
<tr>
<th>GPU</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapid development cycles and high flexibility</td>
<td>Huge I/O Bandwidth</td>
</tr>
<tr>
<td>Large bandwidth to external memory</td>
<td>Deterministic timings/ runtimes</td>
</tr>
<tr>
<td>High Floating-point performance</td>
<td>High bit-level performance</td>
</tr>
</tbody>
</table>
Equivalent to FPGA approach, used by collaboration of KIT and UK Track Trigger Group*

- Uncompress data
- Perform Hough transformation
  - Uses module bend information
- Apply layer condition
- Reject or return track candidates

*An FPGA-Based Track Finder for the L1 Trigger of the CMS Experiment at the High Luminosity LHC (DOI: 10.1109/RTC.2016.7543102)
Hough Transformation

Calculate possible $(\phi_0, q/p_t)$ pairs for each hit

Make histogram in Hough space
Hough Transformation

Calculate possible $(\phi_0, q/p_t)$ pairs for each hit

Make histogram in Hough space
Hough Transformation

Calculate possible $(\varphi_0, q/p_t)$ pairs for each hit

Make histogram in Hough space
CLUSTERING POINTS ARE TRACK CANDIDATES

TTBar event - PU 140
FILTERING BY LAYER CONDITION

TTBar event - PU 140
GPU implementation specifics:

- Optimized for minimum latency
- Computes $q/p_t$-bins in parallel
- Almost no dependence on number of stubs
OVERHEADS...

- Kernel scheduling
- Kernel launch time
- Allocation of shared memory
- ...

Invocation and setup of kernels is too costly, we need to keep it running continuously.

NEEDS SPINNING KERNEL
BENCHMARK KERNEL RUNTIME - SPINNING

CURRENTLY NOT POSSIBLE IN OPENCL: CACHE CAN'T BE FLUSHED FROM KERNEL
WHAT ABOUT DATA TRANSFER?
DMA SETUP – CPU ONLY STARTS THE KERNEL

(Red) Conventional transfer  (Green) RDMA transfer
DMA MEASUREMENT SETUP

PRELOAD DATA

1 GPU: request data

FPGA

KERNEL RUNS CONTINUOUSLY

GPU
DMA MEASUREMENT SETUP

1 GPU: request data

1 GPU: request data

2 FPGA: Data Transfer

2 GPU: Poll for data

FPGA

GPU

SIMULTANEOUSLY

KERNEL RUNS CONTINUOUSLY
DMA MEASUREMENT SETUP

PRELOAD DATA

1 GPU: request data

2 FPGA: Data Transfer

SIMULTANEOUSLY

 kernels runs continuously

2 GPU: Poll for data

3 GPU: Computes

At the moment we don’t write back into the FPGA,
DMA BENCHMARK: POLLING – SPINNING KERNEL

read and write 160 stubs (64 bits each)

➤ Start transfer ➤ Poll for data ➤ Write back result

LATENCY LIMIT

* ➤ estimated response time
DMA BENCHMARK – HOUGH TRANSFORMATION

➤ Read/Uncompress data ➤ Compute ➤ Poll

Time [μs]

8,00
6,00
4,00
2,00
0,00

TRANSFER COMPUTATION TOTAL

Mean Max

* latency limit

160 stubs, 1 sector

* estimated response time
INCREASING THROUGHPUT

- Computation time is higher than data transfer
- We can hide the transfer behind the computation

INTERLEAVED APPROACH

- Start data transfer for current data set
- Do calculations on previous dataset (lies in register memory)
- Poll new data (should take less time)

INCREASES THROUGHPUT AT COST OF LATENCY
DMA BENCHMARK: INTERLEAVED HT

Data older, throughput higher

➤ (poll) Read/Uncompress data ➤ Ask for data ➤ Compute

- Mean
- Max

* estimated response time
NEW APPROACH - HEXAGONAL HOUGH-SPACE

- Hexagonal bins in hough space
- Suppresses fake candidates
- Runtime comparable
- only 1 possible bin per row
- less algorithmic branching

CAVEAT: NEEDS MORE BINS (FACTOR OF 2)
COMPARISON OF FAKE RATES - REGULAR VS. HEXAGONAL

Preliminary Results

**TRACK CANDIDATES**

- **REGULAR**
- **HEXAGONAL**

Results for TTBar Dataset PU140, whole detector, 1 event
DMA BENCHMARK: HEXAGONAL HT

➤ (poll) Read/Uncompress data ➤ Ask for data ➤ Compute

- (poll) Read/Uncompress data
- Ask for data
- Compute

Mean
Max

* latency limit

* estimated response time

Time [μs]

0,00
1,00
2,00
3,00
4,00
5,00
6,00
7,00
8,00

POLLING
COMPUTATION
TOTAL
CONCLUSIONS

Performance:

- Computational time of around 4 μs
- Transfer time of around 2 μs

Surpassed our expectations

Development is faster

More complex algorithms are possible:

- Example: hexagonal approach

Data transfer using standard interfaces is challenging
Need to process multiple sectors per card in future

Look at performance of newer cards

- High Bandwidth Memory,
  already in consumer model cards,
  promises 2-4x better throughput

Investigate new transfer technologies

- PCIe 4.0 (2x faster)
- nv-link (5-10x faster)

MOORS LAW IS OUR FRIEND!
THANK YOU!

QUESTIONS?
CUDA 7.5 – TESLA K40C

TTBar PU140 CUDA 7.5, spinning kernel

- time [μs]
- stubs

The graph shows the distribution of time in microseconds (μs) against stubs for TTBar PU140 using CUDA 7.5 on a TESLA K40C GPU.
CUDA 8 – TESLA K40C

TTBar PU140 CUDA 8, spinning kernel

Time [μs] vs. Stubs

Graph showing the distribution of time versus stubs for TTBar PU140 using CUDA 8 on a Tesla K40C.