Performance Study of a GPU in Real-Time Applications for HEP Experiments

Wesley Ketchum¹,
Silvia Amerio², Denis Bastieri²,³, Matteo Baucë²,³, Pierluigi Catastini⁴, Kristian Hahn⁴, Young-Kee Kim¹,⁴, Tiehui Liu⁴, Donatella Lucchesi²,³, and Giorgio Urso⁵

(1) University of Chicago, (2) INFN Padova,
(3) University of Padova, (4) FNAL, (5) ORMA Software
Introduction

- Research project interested in basic R&D for new trigger techniques
- Use resources available at CDF trigger test stand
  - Hardware
  - Testing software
  - People
- Outline
  - Why we’re interested in GPUs
  - Our experimental setup
  - Current measurements
  - Looking ahead…
Motivation

- Power of GPUs has increased rapidly due to demands of 3D graphics
  - Highly parallelized architecture
  - High memory bandwidth
- Many applications of GPUs outside of imaging
  - Commercially available → cheaper than dedicated hardware
  - Application programming interfaces like nVidia’s CUDA C ease development of software for new applications

- **Are GPUs suitable for low-latency environments, like a HEP trigger?**
## GPU vs CPU Computation

<table>
<thead>
<tr>
<th>CPU (Intel Core i7-930)</th>
<th>GPU (nVidia GeForce GTX 285)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited number of simultaneous calculations possible</td>
<td>Designed for running many instances of same routine simultaneously</td>
</tr>
<tr>
<td>1 microprocessor</td>
<td>30 microprocessors</td>
</tr>
<tr>
<td>4 cores</td>
<td>240 cores</td>
</tr>
<tr>
<td>8 threads</td>
<td>1024 x 30 threads (max)</td>
</tr>
<tr>
<td>Large cache size</td>
<td>Small cache size</td>
</tr>
<tr>
<td>8 MB</td>
<td>8 kB / microprocessor</td>
</tr>
</tbody>
</table>
GPU vs CPU Computation

CPU (Intel Core i7-930)  GPU (nVidia GeForce GTX 285)

From nVidia CUDA C Programming Guide (v 3.2)
## GPU vs CPU Computation

<table>
<thead>
<tr>
<th>CPU (Intel Core i7-930)</th>
<th>GPU (nVidia GeForce GTX 285)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Limited number of simultaneous calculations possible</td>
<td>Designed for running many instances of same routine simultaneously</td>
</tr>
<tr>
<td>1 microprocessor</td>
<td>30 microprocessors</td>
</tr>
<tr>
<td>4 cores</td>
<td>240 cores</td>
</tr>
<tr>
<td>8 threads</td>
<td>1024 x 30 threads (max)</td>
</tr>
<tr>
<td>Large cache size</td>
<td>Small cache size</td>
</tr>
<tr>
<td>8 MB</td>
<td>8 kB / microprocessor</td>
</tr>
<tr>
<td>Sits directly on motherboard</td>
<td>Communicates with CPU through PCIe bus</td>
</tr>
<tr>
<td></td>
<td>Latency scale set by host (CPU) ↔ device (GPU) communication</td>
</tr>
</tbody>
</table>
GPU Memory Structure

- Various memory locations for storing/accessing data
  - **Global Memory**
    - Most available space
    - Read/Write
    - Slow access
  - **Constant/Texture Memory**
    - Smaller storage space
    - Read Only
    - Cacheable on multiprocessors (faster access)
  - **Registers/Shared Memory**
    - Limited storage space
    - Read/Write
    - Fast access for individual threads for thread blocks

*From NVIDIA CUDA C Best Practices Guide (v 4.0)*
Experimental Setup: Data Flow

**Steps in PC**
- Receive input data
- Copy input to GPU
- Perform calculations
- Copy results from GPU
- Send output

**Goal:** Measure total time for performing an HEP trigger algorithm from input going into the PC \(t_1\) and the output leaving the PC \(t_2\) and determine latency \(t_2 - t_1\)
Input/Output: PULSARS

- PULSAR (PULSer And Recorder) boards used in CDF Level 2 trigger system
  - Highly configurable
    - Transmit/receive CERN S-LINK
- Perform studies at CDF L2 Test Stand
  - Measure arrival time of data packets very well
  - PC running in real-time trigger environment
Input/Output: S-LINK PCI Cards

- S-LINK data received/sent on special PCI cards
  - FILAR (Four Input Links for Atlas Readout)
  - SOLAR (Single Output …)
- FILAR/SOLAR cards used in current CDF L2 system
  - Inherit drivers/operation code from L2 upgrade effort

FILAR (above) and SOLAR (left, without S-LINK mezzanine attached).

From http://hsi.web.cern.ch/HSI/s-link/devices
Experimental Setup: Data Flow

**Steps in PC**
- Receive input data
- Copy input to GPU
- Perform calculations
- Copy results from GPU
- Send output

1. **PULSAR S-LINK Tx**
2. **PULSAR S-LINK Rx**
PC↔PULSAR Communication

- PULSAR sends hit combinations to PC
  - Default: 500 S-LINK words \(\rightarrow 20.5 \mu s\) latency
- PC sends back some of results to PULSAR
  - Default: 100 S-LINK words
    - doesn’t contribute much to latency

**Steps in PC**
- Receive packets on FILAR
- Copy input to GPU
- Perform calculations
- Copy results from GPU
- Send output to PULSAR
The Computation: Linearized Track Fitting

- Want to run algorithm that would be used in HEP trigger
- CDF Silicon Vertex Trigger (SVT) finds displaced vertices at L2
  - Pattern Recognition to form hit combinations (roads)

Event Hits

Compare to Pattern Bank

- Perform track-fitting inside roads using simple scalar product

\[ p_i = \vec{f}_i \cdot \vec{x} + q_i \]

Known constants. Precalculated and stored in memory

track parameters (output)

track coordinates (input hit information)
Calculations in CPU Only

- Run track-fitting algorithm to “fit” fixed number of tracks
  - Fixed input and output word lengths
  - Fit 1 track (= 1 word) at a time
- Small spread in CPU latency times
- Mean latency increases linearly

Summary of Data Flow
- Receive packets on FILAR
- Copy input to GPU
- Perform calculations (CPU)
- Copy results from GPU
- Send output to PULSAR
CPU↔GPU (Host↔Device) Communication

- Copy input words to GPU global memory
  - Default: 500 words → 6 μs latency
- Copy results from GPU back to CPU
  - Default: 2000 words (4 output words for each input word) → 19 μs

Summary of Data Flow
- Receive packets on FILAR
- Copy input to GPU
- Perform calculations
- Copy results from GPU
- Send output to PULSAR
Calculations in GPU

- Run track-fitting algorithm: one track fit per thread
  - Amount of memory transfer between CPU and GPU held constant
- As compared to CPU…
  - Latencies much longer in GPU (~60 μs total)
  - Spread of latencies much larger in GPU

Summary of Data Flow
- Receive packets on FILAR
- Copy input to GPU
- Perform calculations
- Copy results from GPU
- Send output to PULSAR
Varying GPU Memory Lookup

- Algorithm accesses predefined constants for track fitting

\[ p_i = \mathbf{f}_i \cdot \mathbf{x} + q_i \]

- Location in memory affects latency

- Significant dependence of latency on handling of memory lookup

- Differences ~ 10 µs between register and global memory

- Good management \( \rightarrow \) optimized performance

*From nVidia CUDA C Best Practices Guide (v 4.0)*
Future Measurements

- Further testing of runtime properties of GPU
  - Optimal thread management in GPU
  - Strategies for addressing long latency of host ↔ device communication
  - Memory access strategies within GPU
  - **ALL within context of real-time trigger system**
- Capable of testing more complex code:
  - Construct silicon hit combinations inside GPU
  - Perform calorimeter tower cluster for jet triggers
  - Directly compare performance of full trigger algorithms to current CDF L2 benchmarks
Conclusions

- GPUs promising idea for future HEP trigger applications
  - Designed for running parallel algorithms with high memory bandwidth
  - Familiar software development
  - Commercial product in a consumer-driven market
- Still, some limitations to be investigated and understood
  - Slow latency for host↔device communication
  - Sensitivity to memory access requires careful optimization
- CDF L2 test stand hosts detailed performance studies in a real-time trigger environment
  - Established some base line performance marks
  - More detailed studies underway!
Typical Spread in GPU

- Mean of GPU latency measurements can vary from run to run
  - Means vary by ~ 0.3 $\mu$s
Outline of Results

- IO Time for Receiving and Sending Signals
  - As function of Number of Input Words
  - As function of Number of Output Words

- Latency for Host→Device and Device→Host Copying
  - As function of Input/Output Words

- Latency for CPU measurements
  - Varying number of calculations

- Latency for GPU measurements
  - For constant number of calculations
  - Varying number of calculations
  - Varying type of memory access
Experimental Setup: Data Flow

- Inputs loaded into S-LINK transmitter PULSAR
  - Use each S-LINK word to represent one “hit combination” set
- Send patterns to PC and S-LINK receiver directly
- In the PC:
  - Receive packets on FILAR
  - Copy input to GPU
  - Perform calculations
  - Copy output from GPU
  - Send output to Pulsar on SOLAR
- S-LINK Receiver PULSAR measures time of incoming packets for latency