An Exploration of CUDA and CBEA for gravitational wave data-analysis

Jens Breitbart\(^1\)  Gaurav Khanna\(^2\)

\(^1\)Research Group Programming Languages / Methodologies  
University of Kassel

\(^2\)Physics Department  
University of Massachusetts at Dartmouth

Rethymno, Crete, Greece  
September 19, 2009
Overview

Einstein@Home
data-parallel application

Playstation 3
> 200 GFLOP

GeForce 280 GTX
~ 1 TFLOP
The application

Performance relevant parts:

1. F-Statistics
2. Hough transformation

Runtime:

<table>
<thead>
<tr>
<th>Test case</th>
<th>F-Statistics</th>
<th>Hough transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full work unit</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Small work unit</td>
<td>90%</td>
<td>10%</td>
</tr>
</tbody>
</table>
The application

Performance relevant parts:

1. F-Statistics
2. Hough transformation \(\leftrightarrow\) will be replaced by a faster algorithm

Runtime:

<table>
<thead>
<tr>
<th>Test case</th>
<th>F-Statistics</th>
<th>Hough transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full work unit</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Small work unit</td>
<td>90%</td>
<td>10%</td>
</tr>
</tbody>
</table>
F-Statistics pseudo code

```plaintext
// iterations are independent
for each (f in frequency_band(...))

// reduction in both loops
for each (d in detectors)
    for each (SFT in SFTs(f))
        result[f] += calc(f, d, SFT)
```
Overview

Source: M. Gschwind et al., Hot Chips-17, August 2005
CBEA implementation

```
// iterations are independent
for each (f in frequency_band(...))

// reduction in both loops
for each (d in detectors)
    for each (SFT in SFTs(f))
        result[f] += calc(f, d, SFT)
```

Parallelization

Execute the outer loop in parallel at all available SPEs.
CBEA implementation

Parallelization

Execute the outer loop in parallel at all available SPEs.

Two implementations

<table>
<thead>
<tr>
<th></th>
<th>DMA-FStat</th>
<th>EA-FStat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data transfer</td>
<td>Explicit DMA transfers</td>
<td>Software cache</td>
</tr>
<tr>
<td>LOC for data transfer</td>
<td>122</td>
<td>9</td>
</tr>
<tr>
<td>Runtime small test case</td>
<td>2:30 min</td>
<td>2:34 min</td>
</tr>
<tr>
<td>Supports full test case</td>
<td>X</td>
<td>✓</td>
</tr>
</tbody>
</table>
**CBEA implementation**

### Parallelization

Execute the outer loop in parallel at all available SPEs.

### Two implementations

<table>
<thead>
<tr>
<th></th>
<th>DMA-FStat</th>
<th>EA-FStat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data transfer</td>
<td>Explicit DMA transfers</td>
<td>Software cache</td>
</tr>
<tr>
<td>LOC for data transfer</td>
<td>122</td>
<td>9</td>
</tr>
<tr>
<td>Runtime small test case</td>
<td>2:30 min</td>
<td>2:34 min</td>
</tr>
<tr>
<td>Supports full test case</td>
<td>X</td>
<td>✓</td>
</tr>
</tbody>
</table>
CBEA Performance at a full work unit

![CBEA Performance Chart]

Breitbart, Khanna

An Exploration of CUDA and CBEA for gravitational wave data-analysis
Overview

- All threads execute the same sequential program.
- Threads in one `threadblock` can be synchronized and share fast on chip memory (`shared memory`).
- Threads in different threadblocks cannot be synchronized.
- CPU- and all GPU-threads share slow global memory.
  - GPU access: $\sim 100$ times slower than shared memory access.
  - CPU access: via DMA transfers with high initialization cost.
CUDA implementation

// iterations are independent
for each (f in frequency_band(...))

// reduction in both loops
for each (d in detectors)
    for each (SFT in SFTs(f))
        result[f] += calc(f, d, SFT)

Parallelization

Every inner loop iteration is executed by a separate thread. Every outermost loop iteration is executed by a threadblock.
CUDA implementation

Parallelization

Every inner loop iteration is executed by a separate thread. Every outermost loop iteration is executed by a threadblock.

Development was an evolutionary approach with three versions.

Version 1

- Only the innermost loop is executed at the device.
- Problem: runtime is about 70 times slower than the original code, due to memory transfers to global memory.
Version 2 - old and new data structure
Version 3 uses the new data structures and executed the whole F-Statistics at the GPU.

### Performance

<table>
<thead>
<tr>
<th>Test case</th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small work unit</td>
<td>2:19 min</td>
<td>1:22 min</td>
</tr>
<tr>
<td>Full work unit</td>
<td>11:00 min</td>
<td>7:00 min</td>
</tr>
</tbody>
</table>

Hardware: AMD Opteron (2 GHz), GeForce 280 GTX
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>CBEA</th>
<th>CUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Application</strong></td>
<td>No perfect match for the architecture</td>
<td></td>
</tr>
<tr>
<td><strong>Performance (FStat only)</strong></td>
<td>$\sim 2$ min</td>
<td>$\sim 1.5$ min</td>
</tr>
<tr>
<td><strong>Data structure</strong></td>
<td>Use existing one</td>
<td>Required new one</td>
</tr>
<tr>
<td><strong>Future</strong></td>
<td>?</td>
<td>Scales with new hardware</td>
</tr>
</tbody>
</table>
Comparison

<table>
<thead>
<tr>
<th></th>
<th>CBEA</th>
<th>CUDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>No perfect match for the architecture</td>
<td></td>
</tr>
<tr>
<td>Performance (FStat only)</td>
<td>$\sim 2 \text{ min}$</td>
<td>$\sim 1.5 \text{ min}$</td>
</tr>
<tr>
<td>Data structure</td>
<td>Use existing one</td>
<td>Required new one</td>
</tr>
<tr>
<td>Future</td>
<td>?</td>
<td>Scales with new hardware</td>
</tr>
</tbody>
</table>

Thank you 😊