Smart Multi-Task Scheduling for OpenCL Programs on CPU/GPU Heterogeneous Platforms

Yuan Wen, Zheng Wang, Michael O’Boyle

The University of Edinburgh
Introduction

Hundreds of applications across a wide range of fields already optimised for GPUs

CPU+GPU system needs to move from application dedicated devices to platforms that support multiple concurrent user applications
OpenCL is functionally BUT NOT performance portable.
Motivation

Scheduling policy matters

Good scheduling policies are able to improve system throughput several times higher than the poor ones.
Information needed for task scheduling

**Best Information**
- Necessary for BEST scheduling

**Execution Time**
- Online profiling needed
- Changing with different runtime inputs
- Have a very wide range
- Impossible to predict

**Speedup**
- Easy to classify
- Limited range
- Profiling free
- Changing with various input data
- Not enough for BEST scheduling

**Second Best**
Speedup Classification

Speedup of an OpenCL kernel on GPU over CPU is decided by the characteristic of the kernel together with runtime parameters.

Kernel Static Features:
- # instructions
- # blocks
- # math_func
- # int operations
- # control instructions
- # barriers
- # load/store
- # br/condbranches
- # vector operations
- # float operations
- # logic operations
- # atomic operations

Runtime Parameters:
- global work size
- local work size
- input size
- output size

Speedup Classifier

Classified Speedup
Implementation

Static Feature Extraction

- OpenCL kernel files
- LLVM based feature extraction library
- JIT

Static features

Runtime Parameter Extraction

- By wrapping OpenCL API calls, runtime parameters are able to be extracted and set on the fly

- Create context on both CPU and GPU
- Create command queue for each device
- Get global and local work items
- Hook input buffer size and output buffer size
- Launch kernel to selected device on the fly
Implementation

Machine learning based classification

- Two kinds of classifiers are used

Decision Tree

Support Vector Machine

High Speedup

Low Speedup

Small Margin

High Speedup

Large Margin

Low Speedup
Scheduling

- Sort tasks according to their classified speedups
- Dequeue task from high speedup end to GPU when it is idle
- Dequeue task from low speedup end to CPU when it becomes idle
- Launch the last task to both CPU and GPU; the duplicate which runs slower will be abandoned immediately after its faster counterpart finished on the other device
## Experiment

<table>
<thead>
<tr>
<th></th>
<th>Intel CPU</th>
<th>Nvidia GPU</th>
<th>AMD GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>Core i7 2600K</td>
<td>GeForce GTX 590</td>
<td>Radeon HD 7970</td>
</tr>
<tr>
<td><strong>Core Clock</strong></td>
<td>3.4 GHz</td>
<td>1215 MHz</td>
<td>1000 MHz</td>
</tr>
<tr>
<td><strong>Core Count</strong></td>
<td>4</td>
<td>1024</td>
<td>2048</td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td>8 GB</td>
<td>3 GB</td>
<td>3 GB</td>
</tr>
<tr>
<td><strong>Memory Bandwidth</strong></td>
<td>21 GB</td>
<td>327 GB</td>
<td>288 GB</td>
</tr>
</tbody>
</table>

## Hardware Platforms

## Benchmark Suites

- Nvidia
- Parboil
- AMD
- Polybench
Metrics

System Throughput

$$STP = \frac{\sum T_{FCFS}^i}{\max(\sum T_{cpu}^m, \sum T_{gpu}^n)}$$

Average Normalised Turnaround Time

$$ANTT = \frac{1}{n} \sum_{i=1}^{n} \frac{T_{sch}^i}{T_{FCFS}^i}$$
Results

Speedup classifier accuracy

SVM: 87%

Decision Tree: 72%
Results

Alternative Scheduling Policies

**All_on_CPU:** dispatch all kernels to CPU according to arriving order

**All_on_GPU:** dispatch all kernels to GPU according to arriving order

**FCFS (baseline):** this is first come first served approach. Tasks will be dispatched to either device, CPU or GPU, when it becomes idle

**Input size guided:** dispatch task with larger input data size to GPU, smaller input data size to CPU.

**Work item guided:** dispatch tasks with larger number of work items to GPU, small number of work items to CPU
Results

Experiment result on Nvidia platform

System throughput

on average 21% improvement over baseline (FCFS)

Average normalised turnaround time

on average 56% improvement over baseline (FCFS)
Results

Experiment result on AMD platform

System throughput

on average 25% improvement over baseline (FCFS)

Average normalised turnaround time

on average 65% improvement over baseline (FCFS)
Results

Compared to OpenCL kernel partition approach

**System throughput**

On polybench:
- Our STP is 1.15x of baseline
- FludiCL is 0.93 of baseline

<table>
<thead>
<tr>
<th>Task Group Sizes</th>
<th>FluidiCL</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>0.8</td>
<td>0.9</td>
</tr>
<tr>
<td>medium</td>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>large</td>
<td>1.1</td>
<td>1.2</td>
</tr>
<tr>
<td>Geomean</td>
<td>1.15</td>
<td>1</td>
</tr>
</tbody>
</table>

**Average normalised turnaround time**

On polybench:
- Our ANTT is 0.55 of baseline
- FludiCL is 1.08 of baseline

<table>
<thead>
<tr>
<th>Task Group Sizes</th>
<th>FluidiCL</th>
<th>Our Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>small</td>
<td>1.25</td>
<td>1</td>
</tr>
<tr>
<td>medium</td>
<td>1.25</td>
<td>1</td>
</tr>
<tr>
<td>large</td>
<td>1.25</td>
<td>1</td>
</tr>
<tr>
<td>Geomean</td>
<td>1.25</td>
<td>1</td>
</tr>
</tbody>
</table>
Limit Study

Compared to Oracle scheduling (experiment on Nvidia platform)

Reasons for performance gaps between our approach and oracle scheduling:

- Accuracy of our classifier
- Binary speedup classification
- Task execution time is unknown
Limit Study

Classifiers with different accuracy:

- Decision Tree: 72%
- SVM: 87%
- Perfect: 100% (classify tasks into high or low speedup category with 100% accuracy)

System throughput

Average normalised turnaround time
Limit Study

Finer Grained Speedup Classifiers

Throughput Normalized to Oracle

Classifier Granularity

- Binary classifier upper bound
- Upper bound
Conclusion

- Developed a speedup classifier for OpenCL kernels
- Proposed a classified-speedup based scheduling approach
- Achieved 21% and 25% throughput improvement on Nvidia and AMD platform
- Decreased ANTT to 0.56 and 0.65 on Nvidia and AMD platform
- Provided detailed limit study
  - Accuracy of the classifier has an impact on performance
  - Fine grained classifiers are able to improve performance
  - Upper bound exists for fine grained classifier
  - For oracle scheduling, execution time must be known in advance
Thank you

Q & A