Coordinating the Use of GPU and CPU for Improving Performance of Compute Intensive Applications

George Teodoro\textsuperscript{1}, Rafael Sachetto\textsuperscript{1}, Olcay Sertel\textsuperscript{2}, Metin Gurcan\textsuperscript{2}, Wagner Meira Jr.\textsuperscript{1}, Umit Catalyurek\textsuperscript{2}, Renato Ferreira\textsuperscript{1}

1. Federal University of Minas Gerais, Brasil
2. The Ohio State University, US
Motivation

• High performance computing
  – Large cluster of-the-shelf components
  – Multi-core/Many-core
  – GPGPU
    • Massively parallel
    • High speedups compared to the CPU
Motivation

• But... GPU is not so fast in all scenarios...
• Current frameworks
  – Assume exclusive use of GPU or CPU
Goal

• Target heterogeneous environments
  – Multiple CPU-cores/GPUs
  – Distributed environments
• Efficient coordination of the devices
  – Scheduling tasks according to their specificities
• High level programming abstraction
Outline

• Anthill
• Supporting heterogeneous environments
• Experimental evaluation
• Conclusions
Anthill

• Based on the filter-stream model (DataCutter)
  – Application decomposed into a set of filters
  – Communication using streams
  – Transparent instance copy
  – Data flow
  – Multiple dimensions of parallelism
    • Task parallelism
    • Data parallelism
Anthill
Filter programming abstraction

- Event driven interface
  - Aligned with the data flow model
- User provide data processing functions to be invoked upon availability of data
- System controls invocation of user function
  - Dependency analysis
  - Parallelism
Event handlers

• User provided functions
• Operate on data objects
  – Updates filter state (global)
  – May trigger communication
  – Returns after processing the data element
• Gets invoked automatically when data is available
  – And dependencies are met
Supporting heterogeneous resources

• Event handler implemented to multiple devices
  – Each filter may be implemented targeting the appropriate device

• Multiple devices used in parallel

• Anthill run-time chooses the device for each event
Heterogeneous support overview

[Diagram showing the interaction between different components such as Event Handler 1, Event Handler 2, Event Handler X, Device Scheduler, Event Executor (CUDA/Pthreads), and Anthill. The diagram illustrates the flow of events and messages between these components.]
Device scheduler

- Assumes
  - Events are independent
  - Out-of-order execution

- Scheduling policies
  - FCFS – first-come, first-served
  - DWDR – dynamic weighted round robin
    - Orders events according to its performance to each device
    - Selects the event with the highest speedup
    - User given function
Neuroblastoma Image Analysis System

• Classify tissues in different subtypes of prognostic significance
• Very high resolution slides
  – Divided in smaller tiles
• Multi-resolution image analysis
  – Mimics the way pathologists examine them
Anthill implementation

**IMAGE READER**
- Input RGB image: tileId, resolution
- Compute tile: tile Id, resolution

**COLOR CONVERSION**
- LAB image: tileId, resolution

**STATISTICAL FEATURES**
- tileId, statistics

**CLASSIFIER**
- CPU only handlers
- CPU and GPU handlers

**START/OUTPUT**
Experimental results

• Setup
  – 10 PCs with an Intel Core 2 Duo CPU 2.13GHz / NVIDIA GeForce 8800GT GPU
  – 4 PCs with a dual quad-core AMD Opteron 2.00GHz processor/ NVIDIA GeForce GTX 260
  – Input data: images of 26,742 tiles using two resolution levels: 32x32 and 512x512
NBIA tasks analysis - performance variation

Dual quad-core AMD Opteron 2.00GHz/NVIDIA GeForce GTX260
Heterogeneous scheduling analysis

Recalc (%) | 12
---|---
Resolution | Low | High
1 CPU core - FCFS | 263 | 215
1 CPU core – DWRR | 21592 | 4

16 + 1 = 30 ??
Heterogeneous scheduling analysis

FCFS

DWDR
Heterogeneous scheduling analysis

<table>
<thead>
<tr>
<th># of CPU cores</th>
<th>FCFS</th>
<th></th>
<th>DWRR</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>1</td>
<td>637</td>
<td>58</td>
<td>10714</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>117</td>
<td>133</td>
<td>15748</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1925</td>
<td>173</td>
<td>18614</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>2090</td>
<td>219</td>
<td>18634</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>2872</td>
<td>286</td>
<td>20070</td>
<td>40</td>
</tr>
<tr>
<td>6</td>
<td>3819</td>
<td>393</td>
<td>20147</td>
<td>76</td>
</tr>
<tr>
<td>7</td>
<td>4726</td>
<td>478</td>
<td>20266</td>
<td>57</td>
</tr>
</tbody>
</table>
Distributed environment evaluation
Conclusions

• Relative performance between CPU/GPU is data dependent
• Adequate scheduling among heterogeneous processors doubled the performance of the application
• Neglect the CPU is a mistake
• Data-flow is an interesting model to exploit parallelism
Future work

• New scheduling techniques

• Execution in cluster with heterogeneity among the computing nodes
Questions?