Determining a Device Crossover Point in CPU/GPU Systems for Streaming Applications

Sudeep Kanur, Wictor Lund, Leonidas Tsipoulos, Johan Lilius
Embedded Systems Lab, Åbo Akademi University
{skanur, wlund, ltsiopou, jolilus}@abo.fi
Context: Accelerating streaming data-flow application

Introduces latency

Bursty chunks

Data-flow application

N/w
Efficient processing of streaming data

- Bursty
- No. of workitems
- Execution time
- Crossover point
- Virtual startup time
- CPU
- GPU
- N/w
- Chunks

Åbo Akademi University | Domkyrkotorget 3 | 20500 Åbo | Finland
To find Device Crossover Point

Application characteristics

Analytical Model

u-benchmarks

Define Region of Interest

Refine the model

Get Device Crossover point
Analytical Model

• Refinement of model proposed by Hong and Kim [1]
  – MWP: Number of warps a core can concurrently access during one memory access request
  – CWP: Number of warps a core can compute during memory request of warp is being serviced

• Two major modification in the model and then applied to both CPUs and GPUs

Modifications

- Differences in the way threads are executed in devices
- GPUs group threads and run them in parallel
- Active threads on GPU increases with a step of parallel elements
- Active threads on CPU increases linearly in CPU
- Also, a typical CPU core has a max limit of 1-2 based on SMT capabilities
- Memory access patterns simplified for AMD devices
Refinement

- Model doesn't take hardware optimizations into account → Overestimation of execution time
- We define a region of interest for a chunk size: $n_{CU} \times N_{\text{eff}} \times \text{(group size)}$
- Find correction factor at this point
Application Characteristics

- Validation on kernels of H.264/MPEG-4 AVC and Motion-JPEG
- Applications written in RVC-CAL dataflow language.
- OpenCL code generation through ORCC compiler
  - SDF actor that are stateless are translated to OpenCL code [2]
- OpenCL code generated uses global memory only and has no synchronization primitives

## Experiments

- CPU - AMD A10-7870K Accelerated Processing Unit
- GPU – R7 integrated GPU
- OpenCL 1.2
- Kernels are run for varying chunk sizes and execution times are noted.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>CPU</th>
<th>GPU</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>WarpSize</td>
<td>128</td>
<td>64</td>
<td>-</td>
</tr>
<tr>
<td>nCU</td>
<td>4</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>Freq</td>
<td>2.4</td>
<td>1.2</td>
<td>GHz</td>
</tr>
<tr>
<td>Mem rate</td>
<td>3.4</td>
<td>29</td>
<td>GBps</td>
</tr>
<tr>
<td>DRAM lat</td>
<td>5</td>
<td>70</td>
<td>cycles</td>
</tr>
<tr>
<td>Inst lat</td>
<td>72</td>
<td>84</td>
<td>cycles</td>
</tr>
</tbody>
</table>
Results

<table>
<thead>
<tr>
<th>Kernel Name</th>
<th>Model</th>
<th>Measured</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D IDCT</td>
<td>1600</td>
<td>1600</td>
</tr>
<tr>
<td>Zigzag</td>
<td>100</td>
<td>200-250</td>
</tr>
</tbody>
</table>
Discussion

- Large deviation from predicted execution time from the model to the measured
- Only execution times are considered, no startup or transfer times
- Despite shortcomings, prediction of device crossover point is fairly accurate
- Decision is simple once the model is tuned initially around the region of interest
Future Work

• Better model to predict CPU execution time
  – Analyze CFG of the kernel
  – Analysis for arbitrary OpenCL kernel

• Better way to obtain region of interest

• Validation against more devices/kernel pair.
Thank you

Sudeep Kanur, Wictor Lund, Leonidas Tsipoulos, Johan Lilius
Embedded Systems Lab, Åbo Akademi University
{skanur, wlund, ltsiopou, jolilus}@abo.fi