PeaPaw: Performance and Energy Aware Workload Partitioning on Heterogeneous Platforms

Li Tang
Advisor: X. Sharon Hu
University of Notre Dame
Challenges of heterogeneous computing

- Parameterized Apps
- Proxy Apps
- Full Apps
- Design space exploration
- Optimization
- Evaluation
- Parameterized machines
- Simulators
- Real machines

Levels:
- Abstraction: High to Low
- Cost: Low to High
PeaPaw Framework
What we want to achieve

- Given a CPU+GPU platform
  - Find the best workload partition between CPU and GPU
  - Achieve high performance or energy efficiency
Workload partitioning

• Workload
  – Amount of computation (# of flops)
  – Amount of memory traffic (# of bytes)

• Workload partition (WP)
  – Dataset partitioning (DP)
    • Pros: balanced CPU/GPU workload distribution
    • Cons: low execution efficiency
  – Code partitioning (CP)
    • Pros: high execution efficiency
    • Cons: high development cost
Examples of CP and DP

For (i=0; i<10000; i++) {
    a[i] = b[a[i]];
    c[i] = d[i]^10;
}

For (i=0; i<2000; i++) {
    a[i] = b[a[i]];
    c[i] = d[i]^10;
}

For (i=0; i<8000; i++) {
    a[i] = b[a[i]];
    c[i] = d[i]^10;
}

For (i=0; i<10000; i++) {
    a[i] = b[a[i]];
    c[i] = d[i]^10;
}
Overview of PeaPaw

- What parameters should be used?
- How to obtain parameter values?

Machine Parameter Values

WP_1
Pseudo Code on CPU & GPU
% of Dataset on CPU & GPU

WP_n
Pseudo Code on CPU & GPU
% of Dataset on CPU & GPU

Design Goal: Performance or Energy?

Identification of P-Paw or E-Paw

CPU+GPU Platform
Machine Parameter Values

WP
Pseudo Code on CPU & GPU
% of Dataset on CPU & GPU

Final WP
Pseudo Code on CPU & GPU
% of Dataset on CPU & GPU

Performance/energy Estimation & Comparison

Workload Partitioning Guideline

Workload
Pseudo Code
Dataset Description

WP Abstraction

Essential WP Parameter Values

PeaPaw
Application Developers

WP_1
WP_n
...
Modeling Performance/Energy Estimation in PeaPaw
Roofline models of performance & energy

• Basic roofline model of performance [Williams, CACM, 2009]
  – $I$: operational intensity = flop/byte
  – $G_{flop}/s = \min(Peak \ G_{flop}/s, \ Peak \ Mem \ BW \ * \ I)$
  – Simple extension to system level
    • CPU $G_{flop}/s = \min(Peak \ CPU \ G_{flop}/s, \ Peak \ CPU \ Mem \ BW \ * \ I_C)$
    • GPU $G_{flop}/s = \min(Peak \ GPU \ G_{flop}/s, \ Peak \ GPU \ Mem \ BW \ * \ I_G)$
    • Ideal CPU+GPU $G_{flop}/s = CPU \ G_{flop}/s + GPU \ G_{flop}/s$

• Other roofline models of performance [Nugteren, CF, 2012][Llic, CAL, 2014]
  – Only consider single processor platforms

• Roofline model of energy [Choi, IPDPS, 2013]
  – $E_{\text{processor}} = E_{\text{flops}} + E_{\text{bytes}} + p_{\text{static}} \ * T_{\text{processor}}$

CP-independent
CPU+GPU Gflops/s = \frac{N_{f-c} + N_{f-g}}{\max(t_{f-c} N_{f-c}, t_{b-c} N_{b-c}, t_{f-g} N_{f-g}, t_{b-g} N_{b-g})}

CPU cores
\[ t_{f-c} \text{(time per flop)} \]
\[ N_{f-c} \text{(\# of flops)} \]

GPU cores
\[ t_{f-g} \text{(time per flop)} \]
\[ N_{f-g} \text{(\# of flops)} \]

CPU memory
\[ t_{b-c} \text{(time per byte)} \]
\[ N_{b-c} \text{(\# of bytes)} \]

GPU memory
\[ t_{b-g} \text{(time per byte)} \]
\[ N_{b-g} \text{(\# of bytes)} \]
\[ E_{\text{total}} = e_{f-c} N_{f-c} + e_{b-c} N_{b-c} + e_{f-g} N_{f-g} + e_{b-g} N_{b-g} + (p_{p-c} + p_{p-g}) \max(t_{f-c} N_{f-c}, t_{b-c} N_{b-c}, t_{f-g} N_{f-g}, t_{b-g} N_{b-g}) \]

**CPU dynamic energy**

**GPU dynamic energy**

**Total static energy**

**CPU cores**
- \( t_{f-c} \) (time per flop)
- \( e_{f-c} \) (energy per flop)
- \( N_{f-c} \) (# of flops)

**GPU cores**
- \( t_{f-g} \) (time per flop)
- \( e_{f-g} \) (energy per flop)
- \( N_{f-g} \) (# of flops)

**CPU memory**
- \( t_{b-c} \) (time per byte)
- \( e_{b-c} \) (energy per byte)
- \( N_{b-c} \) (# of bytes)

**GPU memory**
- \( t_{b-g} \) (time per byte)
- \( e_{b-g} \) (energy per byte)
- \( N_{b-g} \) (# of bytes)
Getting parameter values

• WP parameters values: $N_{f-c}, N_{b-c}, N_{f-g}, N_{b-g}$
  – Code analysis via control-data flow graph
  – Code inspection

• Hardware parameter values
  • $t_{f-c}, t_{f-g}$: peak performance from specification
  • $t_{b-c}, t_{b-g}$: realistic peak performance through benchmarks (e.g., STREAM)
  • $e_{f-c}, e_{b-c}, e_{f-g}, e_{b-g}, p_{p-c}, p_{p-g}$
    • $n_f * e_f + n_b * e_b + p_s * t = E$
    • Run different tests, each test has different $n_f$ and $n_b$
    • Measure $E$ and $t$ for each test
    • Use linear training to get $e_f$, $e_b$ and $p_s$
Essential WP parameters

- Reduce the WP parameters
  - Dataset size independent
  - Easy for data analysis

\[ \{N_{f-c}, N_{b-c}, N_{f-g}, N_{b-g}\} \rightarrow \{l, l_c, l_g\} \]

\[
\text{CPU+GPU Gflops/s} = \frac{N_{f-c} + N_{f-g}}{\max(t_{f-c} N_{f-c}, t_{b-c} N_{b-c}, t_{f-g} N_{f-g}, t_{b-g} N_{b-g})}
\]

\[
\text{CPU+GPU Gflops/s} = \max\{t_{f-c} \frac{l_c(l-l_g)}{|l_c-l_g|}, t_{b-c} \frac{l-l_g}{|l_c-l_g|}, t_{f-g} \frac{l_g(l-l_c)}{|l_g-l_c|}, t_{f-g} \frac{l-l_c}{|l_g-l_c|}\}
\]
PeaPaw Guidelines
Workload partitioning guidelines - performance

- **Machine balance: relative data handling capability**
  - $BL_c = \frac{t_{b-c}}{t_{f-c}}$
  - $BL_g = \frac{t_{b-g}}{t_{f-g}}$

- **Performance oriented guidelines (P-Paw)**
  - **CPU_DP-GPU_DP: $BL_c = BL_g$**
    - DP on both CPU and GPU
  - **CPU_MEM-GPU_COMP: $BL_c > BL_g$**
    - Memory intensive part on CPU, compute intensive part on GPU
  - **CPU_COMP-GPU_MEM: $BL_c > BL_g$**
    - Compute intensive part on CPU, memory intensive part on GPU
Workload partitioning guidelines - energy

• Gradient energy: dynamic and static energy tradeoff
  \[ \Delta G_{P_f} = \left| e_{f-c} - e_{f-g} \right| - (P_{s-C} + P_{s-G}) \times t_{f-g} \]
  \[ \Delta G_{P_b} = \left| e_{b-c} - e_{b-g} \right| - (P_{s-C} + P_{s-G}) \times t_{b-g} \]

• Energy-oriented guidelines (E-Paw)
  – GPU-only: \( \Delta G_{P_f} > 0, \Delta G_{P_g} > 0, e_{f-c} > e_{f-g} \) and \( e_{b-c} > e_{b-g} \)
    • Use GPU for the whole workload
  – CPU_MEM-GPU_COMP: \( \Delta G_{P_f} > 0, \Delta G_{P_g} > 0, e_{f-c} < e_{f-g} \) and \( e_{b-c} > e_{b-g} \)
    • Memory intensive code on CPU and computation intensive code on GPU
  – Race-to-halt: \( \Delta G_{P_f} + \Delta G_{P_f} < 0 \)
    • Follow the P-Paw to achieve the highest performance

Moving 1 flop from GPU to CPU
Experimental Evaluation
# Processors

<table>
<thead>
<tr>
<th>Processors</th>
<th>Types</th>
<th>Architectures</th>
<th># of cores</th>
<th>Frequency (GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel i7 2600K</td>
<td>CPU</td>
<td>Sandybridge</td>
<td>4</td>
<td>3.4</td>
</tr>
<tr>
<td>Intel i3 2100T</td>
<td>CPU</td>
<td>Sandybridge</td>
<td>2</td>
<td>2.5</td>
</tr>
<tr>
<td>NVIDIA GTX Titan</td>
<td>GPU</td>
<td>Kepler</td>
<td>2688</td>
<td>0.84</td>
</tr>
<tr>
<td>NVIDIA GTX 750</td>
<td>GPU</td>
<td>Kepler</td>
<td>750</td>
<td>0.51</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processors</th>
<th>$T_f$ (pS)</th>
<th>$T_b$ (pS)</th>
<th>$E_f$ (pJ)</th>
<th>$E_b$ (pJ)</th>
<th>$P_s$ (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel i7 2600K</td>
<td>9.5</td>
<td>65.9</td>
<td>118</td>
<td>462</td>
<td>26.8</td>
</tr>
<tr>
<td>Intel i3 2100T</td>
<td>25</td>
<td>73</td>
<td>135</td>
<td>581</td>
<td>9.7</td>
</tr>
<tr>
<td>NVIDIA GTX Titan</td>
<td>0.4</td>
<td>4.2</td>
<td>57</td>
<td>187</td>
<td>64.1</td>
</tr>
<tr>
<td>NVIDIA GTX 750</td>
<td>1.9</td>
<td>14.8</td>
<td>78</td>
<td>169</td>
<td>16.4</td>
</tr>
</tbody>
</table>
## Guidelines – performance & energy

<table>
<thead>
<tr>
<th>Platforms</th>
<th>$\Delta G_P^f$</th>
<th>$\Delta G_P^b$</th>
<th>$P_{s-C} + P_{s-G}$</th>
<th>E-Paw category</th>
</tr>
</thead>
<tbody>
<tr>
<td>i7+Titan</td>
<td>9.5</td>
<td>65.9</td>
<td>118</td>
<td>CPU_MEM-GPU_COMP</td>
</tr>
<tr>
<td>i7+750</td>
<td>25</td>
<td>73</td>
<td>135</td>
<td>Race-to-halt</td>
</tr>
<tr>
<td>i3+Titan</td>
<td>0.4</td>
<td>4.2</td>
<td>57</td>
<td>GPU-only</td>
</tr>
<tr>
<td>i3+750</td>
<td>1.9</td>
<td>14.8</td>
<td>78</td>
<td>GPU-only</td>
</tr>
</tbody>
</table>
## Workload partitions (WPs)

<table>
<thead>
<tr>
<th>Workloads</th>
<th>CPU-only</th>
<th>GPU-only</th>
<th>p-DP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>{7.6, 7.6, 0}</td>
<td>{7.6, 0, 7.6}</td>
<td>{7.6, 7.6, 7.6}</td>
<td>{7.6, 0.1, 8}</td>
</tr>
<tr>
<td>LA</td>
<td>{0.24, 0.24, 0}</td>
<td>{0.24, 0, 0.24}</td>
<td>{0.24, 0.24, 0.24}</td>
<td>{0.24, 0, 0.25}</td>
</tr>
<tr>
<td>DA</td>
<td>{4.4, 4.4, 0}</td>
<td>{4.4, 0, 4.4}</td>
<td>{4.4, 4.4, 4.4}</td>
<td>{4.4, 1.1, 4.9}</td>
</tr>
</tbody>
</table>

- Synthetic application (SA)
- Linear algebra (LA)
- Data assembly (DA)
Validation – performance of SA

(a) i7+Titan
(b) i3+Titan
(c) i7+750
(d) i3+750
(e) SA
Validation – energy efficiency of SA

(a) i7+Titan
(b) i3+Titan
(c) i7+750
(d) i3+750
(e) SA
Validation – performance of DA

(a) i7+Titan

(b) i3+Titan

(c) i7+750

(d) i3+750

(e) DA
Validation – energy efficiency of DA
Summary and future work

• The PeaPaw framework
  • Performance/energy modeling of heterogeneous platforms
  • Performance oriented and energy oriented guidelines for workload partitioning
  • P-Paw and E-Paw categories for guideline selection

• Future work
  • Improve the guideline quality
  • Improve the accuracy of performance/energy estimation
  • Add support to integrated CPU+GPU platforms
Thank you!