MLP-Aware Dynamic Instruction Window Resizing for Adaptively Exploiting Both ILP and MLP

Yuya Kora
Kyohei Yamaguchi
Hideki Ando

Nagoya University
Problem to Solve

• Difficult to improve single-thread performance in memory-intensive programs
  – Memory wall

• Very large instruction window can overcome this problem by exploiting MLP
  – This degrades the clock cycle time
  – Can be solved by pipelining, but...
  – Pipelining prevents ILP from being exploited, degrading IPC in compute-intensive program
Dynamic Instruction Window Resizing

• Adapt window size to available parallelism
  – ILP or MLP

• As more exploitable MLP is predicted
  – Window resources are enlarged and pipeline depth is increased

• If prediction indicates less MLP is exploited (= ILP is more valuable)
  – Window resources are shrunk and pipeline depth is decreased
Prediction when MLP is Exploitable

• If an LLC miss occurs once
  – Predict that MLP is exploitable for a while

• If memory latency has lapsed after the last LLC miss
  – Predict that MLP will not be exploitable

• Rationale
  – LLC misses are clustered in terms of time
Dynamic resizing model achieves as good as best performance for levels 1 to 3 of fixed size model.
- It achieves similar performance to ideal model (no pipelined).
- Imply good adaptability
- 21% speedup for all programs
• Power is increased, but perf is improved $\Rightarrow$ Better energy efficiency
• Memory-intensive: 36% better
• Compute-intensive: 8% worse
• Overall: 8% better
## Cost Efficiency

<table>
<thead>
<tr>
<th>Additional cost</th>
<th>value (per core)</th>
<th>1.6mm²</th>
</tr>
</thead>
<tbody>
<tr>
<td>vs. base core</td>
<td>6%</td>
<td></td>
</tr>
<tr>
<td>vs. Sandy Bridge core</td>
<td>8%</td>
<td></td>
</tr>
<tr>
<td>vs. Sandy Bridge chip</td>
<td>3%</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Speedup</th>
<th>achieved</th>
<th>21%</th>
</tr>
</thead>
<tbody>
<tr>
<td>expected by Pollack’s law</td>
<td>3%</td>
<td></td>
</tr>
<tr>
<td>augmented L2 cache</td>
<td>1%</td>
<td></td>
</tr>
</tbody>
</table>

2MB, 4-way → 2.5MB, 5-way
(increased cost is 1.3x greater than the additional cost)

Good cost/performance ratio, that far exceeds that based on Pollack’s law
Conclusion

• Dynamic instruction window resizing
  – Exploit ILP and MLP adaptively
  – Based on prediction of available parallelism

• Features
  – Very simple
  – Very practical

• Our scheme achieves
  – Performance level similar to the best performance achieved with fix-sized resources
    – 21% speedup
    – 6% extra cost of a core, or 3% of an entire proc chip
  – 8% better energy efficiency