Question

- Do we still care about single thread performance?
Amdahl’s Law and Massive Parallelism

\[ \text{Speedup} = 1.0 \]
Amdahl’s Law and Massive Parallelism

**Speedup\**

\[
\frac{1}{0.55} = 1.82
\]
Amdahl’s Law and Massive Parallelism

Speedup

\[\frac{1}{0.55} = 1.82\]

\[\frac{1}{0.325} = 3.07\]
Amdahl’s Law and Massive Parallelism

Speedup

1.0

1/.55 = 1.82

1/.325 = 3.07

< 10
Non-traditional Parallelism
Non-traditional Parallelism

- Parallelism – Use multiple contexts to achieve better performance than possible on a single context.
Non-traditional Parallelism

- Parallelism – Use multiple contexts to achieve better performance than possible on a single context.
- Traditional Parallelism – We use extra threads/processors to offload computation. Threads divide up the execution stream.
Non-traditional Parallelism

- Parallelism – Use multiple contexts to achieve better performance than possible on a single context.
- Traditional Parallelism – We use extra threads/processors to offload computation. Threads divide up the execution stream.
- Non-traditional parallelism – Extra threads are used to speed up computation without necessarily off-loading any of the original computation
  - Primary advantage → nearly any code, no matter how inherently serial, can benefit from parallelization.
  - Another advantage – threads can be added or subtracted without significant disruption.
Traditional Parallelism
Traditional Parallelism
Traditional Parallelism

Thread 1  Thread 2  Thread 3  Thread 4
Non-Traditional Parallelism
Non-Traditional Parallelism
Non-Traditional Parallelism

Thread 1  Thread 2  Thread 3  Thread 4
Non-Traditional Parallelism

- Speculative precomputation, dynamic speculative precomputation, many others.
Non-Traditional Parallelism

- Speculative precomputation, dynamic speculative precomputation, many others.
- Most commonly – prefetching, possibly branch pre-calculation.
Background -- Helper Threads

- Chappell, Stark, Kim, Reinhardt, Patt, “Simultaneous Subordinate Micro-threading” 1999
  - Use microcoded threads to manipulate the microarchitecture to improve the performance of the main thread.
  - Use a regular SMT thread, with code distilled from the main thread, to support the main thread.
Outline

- Dynamic Speculative Precomputation
- Event-Driven Simultaneous Optimization
  - Value Specialization
  - Inline Prefetching
  - Thread Prefetching
Speculative Precomputation – Motivation

Speedup

- Perfect Memory
- Perfect Delinquent Loads (10)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Speedup Perfect Memory</th>
<th>Speedup Perfect Delinquent Loads</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>3.30</td>
<td>1.41</td>
</tr>
<tr>
<td>equake</td>
<td>6.28</td>
<td>2.76</td>
</tr>
<tr>
<td>gzip</td>
<td>1.14104</td>
<td>1.14104</td>
</tr>
<tr>
<td>mcf</td>
<td>4.79</td>
<td>2.47</td>
</tr>
<tr>
<td>health</td>
<td>32.64</td>
<td>27.90</td>
</tr>
<tr>
<td>mst</td>
<td>5.79</td>
<td>4.46</td>
</tr>
</tbody>
</table>

Benchmark: 1.000 8.910 16.821 24.731 32.642
Speculative Precomputation – Motivation

<table>
<thead>
<tr>
<th></th>
<th>art</th>
<th>equake</th>
<th>gzip</th>
<th>mcf</th>
<th>health</th>
<th>mst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup</td>
<td>3.30</td>
<td>6.28</td>
<td>1.14</td>
<td>4.79</td>
<td>5.79</td>
<td>4.46</td>
</tr>
</tbody>
</table>

- Perfect Memory
- Perfect Delinquent Loads (10)
Speculative Precomputation (SP)
Speculative Precomputation (SP)

- In SP, a \textit{p-slice} is a thread derived from a trace of execution between a \textit{trigger} instruction and the delinquent load.
Speculative Precomputation (SP)

- In SP, a *p-slice* is a thread derived from a trace of execution between a *trigger* instruction and the delinquent load.
- All instructions upon which the load’s address is not dependent are removed (often 90-95%).
Speculative Precomputation (SP)

- In SP, a *p-slice* is a thread derived from a trace of execution between a *trigger* instruction and the delinquent load.
- All instructions upon which the load’s address is not dependent are removed (often 90-95%).
- Live-in register values (typically 2-6) must be explicitly copied from main thread to helper thread.
Speculative Precomputation

Delinquent load
Speculative Precomputation

Trigger instruction

Delinquent load
Speculative Precomputation

- Trigger instruction
- Spawn thread
- Delinquent load
Speculative Precomputation

Trigger instruction

Spawn thread

Prefetch

Delinquent load
Speculative Precomputation

- Trigger instruction
- Spawn thread
- Prefetch
- Memory latency

Delinquent load
Advantages over Traditional Prefetching
Advantages over Traditional Prefetching

- Because SP uses actual program code, can precompute addresses that fit **no predictable pattern**.
Advantages over Traditional Prefetching

- Because SP uses actual program code, can precompute addresses that fit no predictable pattern.
- Because SP runs in a separate thread, it can interfere with the main thread much less than software prefetching. When it isn’t working, it can be killed.
Advantages over Traditional Prefetching

- Because SP uses actual program code, can precompute addresses that fit no predictable pattern.
- Because SP runs in a separate thread, it can interfere with the main thread much less than software prefetching. When it isn’t working, it can be killed.
- Because it is decoupled from the main thread, the prefetcher is not constrained by the control flow of the main thread.
Advantages over Traditional Prefetching

- Because SP uses actual program code, can precompute addresses that fit no predictable pattern.
- Because SP runs in a separate thread, it can interfere with the main thread much less than software prefetching. When it isn’t working, it can be killed.
- Because it is decoupled from the main thread, the prefetcher is not constrained by the control flow of the main thread.
Advantages over Traditional Prefetching

- Because SP uses actual program code, can precompute addresses that fit no predictable pattern.
- Because SP runs in a separate thread, it can interfere with the main thread much less than software prefetching. When it isn’t working, it can be killed.
- Because it is decoupled from the main thread, the prefetcher is not constrained by the control flow of the main thread.
- All the applications in this study already had very aggressive software prefetching applied, when possible.
SP Optimizations
SP Optimizations

- On-chip memory for transfer of live-in values.
SP Optimizations

- On-chip memory for transfer of live-in values.
- Chaining triggers – for delinquent loads in loops, a speculative thread can trigger the next p-slice (think of this as a looping prefetcher which targets a load within a loop)
  - Minimizes live-in copy overhead.
  - Enables SP threads to get arbitrarily far ahead.
  - Necessitates a mechanism to stop the chaining prefetcher.
Advantages from Chaining Triggers

- Chaining triggers executed without impacting main thread
- Target delinquent loads arbitrarily far ahead of non-speculative thread
  - Speculative threads make progress independent of main thread
- *Use basic triggers to initiate precomputation, but use chaining triggers to sustain it*
SP Performance

<table>
<thead>
<tr>
<th></th>
<th>2 Thread Contexts</th>
<th>4 Thread Contexts</th>
<th>8 Thread Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>art</td>
<td>0.80</td>
<td>1.85</td>
<td>2.90</td>
</tr>
<tr>
<td>equake</td>
<td>1.85</td>
<td>3.95</td>
<td>5.00</td>
</tr>
<tr>
<td>gzip</td>
<td>0.80</td>
<td>1.85</td>
<td>2.90</td>
</tr>
<tr>
<td>mcf</td>
<td>1.85</td>
<td>3.95</td>
<td>5.00</td>
</tr>
<tr>
<td>health</td>
<td>2.90</td>
<td>5.00</td>
<td></td>
</tr>
<tr>
<td>mst</td>
<td>1.85</td>
<td>3.95</td>
<td>5.00</td>
</tr>
<tr>
<td>Average</td>
<td>1.85</td>
<td>3.95</td>
<td>5.00</td>
</tr>
</tbody>
</table>
SP Performance

<table>
<thead>
<tr>
<th></th>
<th>2 Thread Contexts</th>
<th>4 Thread Contexts</th>
<th>8 Thread Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup over Baseline</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>art</td>
<td>0.80</td>
<td>0.80</td>
<td>0.80</td>
</tr>
<tr>
<td>equake</td>
<td>1.85</td>
<td>1.85</td>
<td>1.85</td>
</tr>
<tr>
<td>gzip</td>
<td>2.90</td>
<td>2.90</td>
<td>2.90</td>
</tr>
<tr>
<td>mcf</td>
<td>3.95</td>
<td>3.95</td>
<td>3.95</td>
</tr>
<tr>
<td>health</td>
<td>4.00</td>
<td>4.00</td>
<td>4.00</td>
</tr>
<tr>
<td>mst</td>
<td>5.00</td>
<td>5.00</td>
<td>5.00</td>
</tr>
<tr>
<td>Average</td>
<td>2.90</td>
<td>3.95</td>
<td>4.00</td>
</tr>
</tbody>
</table>