A Quantitative Framework for Pre-Execution Thread Selection

Amir Roth
University of Pennsylvania

Gurindar S. Sohi
University of Wisconsin-Madison

MICRO-35
Nov. 22, 2002
Pre-execution: proactive multithreading to tolerate latency

- Start with cache miss
- Extract computation (dynamic backward slice)
- Launch as redundant thread (p-thread)
  - **P-thread** = trigger (launch PC) + body (slice)

- **Effect I**: compressed fetch
  - P-thread fetches \( \gamma \) initiates \( \gamma \) completes miss earlier
- **Effect II**: decoupling
  - Miss in p-thread doesn’t stall main thread
- **Sum effect**: miss “moved” to p-thread
• This paper is not about pre-execution

• These papers are
  • Assisted Execution [Song+Dubois, USC-TR98]
  • SSMT [Chappell+, ISCA99, ISCA02, MICRO02]
  • Virtual Function Pre-Computation [Roth+, ICS99]
  • DDMT [Roth+Sohi, HPCA01]
  • Speculative Pre-Computation [Collins+, ISCA01, MICRO01, PLDI02]
  • Speculative Slices [Zilles+Sohi, ISCA01]
  • Software-Controlled Pre-Execution [Luk, ISCA01]
  • Slice Processors [Moshovos+, ICS01]
P-thread selection: pre-?-execution

- What p-threads? When to launch? (same question)

- **Static p-threads**
  - Target static problem loads
  - Choose <trigger:body> once
  - Launch many instances

- **Hard**: antagonistic criteria
  - Miss coverage
  - Latency tolerance per instance
  - Overhead per instance
  - Useless instances
  - Longer p-thread = better, worse
Quantitative p-thread selection framework

- Simultaneously optimizes all four criteria
- Accounts for p-thread overlap (later)
- Automatic p-thread optimization and merging (paper only)

- Abstract pre-execution model (applies to SMT, CMP)
  - 4 processor parameters
- Structured as pre-execution limit study
  - May be used to study pre-execution potential

This paper: static p-threads for L2 misses
Rest of Talk

- Propaganda
- **Framework proper**
  - Master plan
  - Aggregate advantage
  - P-thread overlap
- Quick example
- Quicker performance evaluation
Plan of Attack and Key Simplifications

- **Divide**
  - P-threads for one static load at a time

- **Enumerate all possible static p-threads**
  - Only p-threads sliced directly from program
  - A priori length restrictions

- **Assign benefit estimate to each static p-thread**
  - Number of cycles by which execution time will be reduced

- **Iterative methods to find set with maximum advantage**

- **Conquer**
  - Merge p-threads with redundant sub-computations
Estimating Static P-thread Goodness

**Key contribution:** simplifications for computational traction

1. One p-thread instance executes at a time (framework)
   - P-thread interacts with main thread only
2. No natural miss parallelism (framework, not bad for L2 misses)
   - P-thread interacts with one main thread miss only
3. Control-less p-threads (by construction)
   - Dynamic instances are identical
4. No chaining (by construction)
   - Fixed number of them

**Strategy**

- Model interaction of one dynamic instance with main thread
- Multiply by (expected) number of dynamic instances
Aggregate Advantage ($\text{ADV}_{\text{agg}}$)

- **$\text{ADV}_{\text{agg}}$:** static p-thread goodness function
  - Cycles by which static p-thread will accelerate main thread
  - Combines four criteria into one number

$$
\text{ADV}_{\text{agg}} = (\text{DC}_{\text{pt-cm}} \times \text{LT}) - (\text{DC}_{\text{trig}} \times \text{OH})
$$

- Collect raw materials from traces
  - $\text{DC}_{\text{trig}} = \# \text{instances launched}$
  - $\text{OH} = \text{overhead per } (\text{SIZE} / \text{BWSEQ}_{\text{proc}})$
  - $\text{DC}_{\text{pt-cm}} = \# \text{misses covered}$
  - $\text{LT} = \text{latency tolerance per } (\text{next slide})$
Dynamic P-thread Latency Tolerance (LT)

- **LT**
  - Starting at trigger
  - Main thread miss execution time ($SCDH_{mt}$)
  - Minus p-thread miss execution time ($SCDH_{pt}$)
  - Bounded by miss latency ($L_{cm}$)

\[
LT = \min(SCDH_{mt} - SCDH_{pt}, L_{cm})
\]

- **SCDH**: sequencing constrained dataflow height
  - Estimate execution time of instruction sequence
  - For each insn: dataflow + **fetch constraint (insn# / BWSEQ)**
    - Computation: explicit
    - Other main thread work: sparse insn #'s

"Pre-Execution Thread Selection". Amir Roth, MICRO-35
SCDH and LT Example

- **Dataflow constraints:**
  - Main-thread: miss latency = 8, other latency = 1, serial deps
  - P-thread: same

- **Sequencing constraints:** \texttt{insn# / fetch bandwidth}
  - Main thread: sparse / 2
  - P-thread: dense / 1

<table>
<thead>
<tr>
<th></th>
<th>SCDH_{mt}</th>
<th>SCDH_{pt}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{addi R5, R5, #16}</td>
<td>max(0/2, 0) + 1 = 1</td>
<td>max(0/1, 0) + 1 = 1</td>
</tr>
<tr>
<td>\texttt{lw R6, 4(R5)}</td>
<td>max(7/2, 1) + 1 = 5</td>
<td>max(1/1, 1) + 1 = 2</td>
</tr>
<tr>
<td>\texttt{slli R7, R6, #2}</td>
<td>max(10/2, 5) + 1 = 6</td>
<td>max(2/1, 2) + 1 = 3</td>
</tr>
<tr>
<td>\texttt{addi R8, R7, #rx}</td>
<td>max(11/2, 6) + 1 = 7</td>
<td>max(3/1, 3) + 1 = 4</td>
</tr>
<tr>
<td>\texttt{lw R9, 0(R8)}</td>
<td>max(12/2, 7) + 8 = 15</td>
<td>max(4/1, 4) + 8 = 12</td>
</tr>
</tbody>
</table>

\[ \text{LT} = \min(15 - 12, 8) = 3 \]
**ADVagg Example**

- **Params:** 40 misses, 8 cycles each
  - Max $\text{ADVagg} = [40*8] - [40*0] = 320$
  - Impossible to achieve

- **As p-thread length increases...**
  - $\text{LT } \eta, \text{DC}_{\text{pt-cm}}$
  - $\text{OH } \eta, \text{DC}_{\text{trig}}$

<table>
<thead>
<tr>
<th></th>
<th>DC_{pt-cm}</th>
<th>LT</th>
<th>LT_{agg}</th>
<th></th>
<th>DC_{trig}</th>
<th>OH</th>
<th>OH_{agg}</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>40</td>
<td>5</td>
<td>200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>8</td>
<td>240</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>30</td>
<td>8</td>
<td>240</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$- DC_{trig} \times OH = OH_{agg} = \text{ADVagg}$

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td>-40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>80</td>
<td>2/4</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>80</td>
<td>3/4</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>80</td>
<td>4/4</td>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>5/4</td>
<td>125</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>115</td>
</tr>
</tbody>
</table>

"Pre-Execution Thread Selection". Amir Roth, MICRO-35
**P-thread Overlap**

- **F:** B, C, D have positive $\text{ADV}^{\text{agg}}$
- **Q:** Why not choose all three?
- **A:** They cover same misses ($\text{LT}^{\text{agg}}$’s overlap)
- **P-thread overlap:** framework...
  - Represents it: slice tree (paper)
  - Corrects for it: reduces $\text{LT}^{\text{agg}}$ by shared component

<table>
<thead>
<tr>
<th>OH$^{\text{agg}}$</th>
<th>LT$^{\text{agg}}$</th>
<th>LT$^{\text{agg-ovlp}}$</th>
<th>LT$^{\text{agg-red}}$</th>
<th>ADV$^{\text{agg-red}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B 60</td>
<td>40*5 = 200</td>
<td>30*5 = 150</td>
<td>200–150 = 50</td>
<td>50–60 = −10</td>
</tr>
<tr>
<td>C 80</td>
<td>30*8 = 240</td>
<td></td>
<td></td>
<td>240–80 = 160</td>
</tr>
</tbody>
</table>

- Choose overlapping p-threads if $\text{ADV}^{\text{agg-red}}$ positive
  - Not in this example
Performance Evaluation

- SPEC2000 benchmarks
  - Alpha EV6, -O2 -fast
  - Complete train input runs, 10% sampling

- Simplescalar-derived simulator
  - Aggressive 6-wide superscalar
  - 256KB 4-way L2, 100 cycle memory latency
  - SMT with 4 threads (p-threads and main thread contend)

- P-threads for L2 misses
  - Prefetch into L2 only
Contribution of Framework Features

- Framework accounts for
  - Latency, overhead, overlap
  - Isolate considerations
- 4 experiments, 3 diagnostics
  - Measured via p-thread simulation

- **GREEDY**: as much LT as possible

---

**% misses covered (LT)**

**% exec increase (OH)**

**% speedup**

---

"Pre-Execution Thread Selection". Amir Roth, MICRO-35
Contribution of Framework Features

- Framework accounts for
  - Latency, overhead, overlap
  - Isolate considerations
- 4 experiments, 3 diagnostics
  - Measured via p-thread simulation

- GREEDY: as much LT as possible
- +LT: as much LT as needed
Contribution of Framework Features

- Framework accounts for
  - Latency, overhead, overlap
  - Isolate considerations
- 4 experiments, 3 diagnostics
  - Measured via p-thread simulation

- **GREEDY**: as much LT as possible
- **+LT**: as much LT as needed
- **+OH**: account for overhead
Contribution of Framework Features

- Framework accounts for
  - Latency, overhead, overlap
  - Isolate considerations
- 4 experiments, 3 diagnostics
  - Measured via p-thread simulation

- **GREEDY**: as much LT as possible
- **+LT**: as much LT as needed
- **+OH**: account for overhead
- **+OVLP**: account for overlap
Pre-Execution Behavior Study

- Example: max p-thread length
  - Full framework
  - 8, 16, 32

- Encouraging (intuitive) result
  - Flexibility increases performance

- Also in paper
  - Merging, optimization, input sets

- ADVagg just a model, not completely accurate
  - ADVagg validation: important part of paper
  - V1: ADVagg predictions should match simulated diagonostics
  - V2: lying to ADVagg should produce worse p-threads
Summary

- P-thread selection
  - Important and hard
- **Quantitative static p-thread selection framework**
  - Enumerate all possible static p-threads
  - Assign a benefit value (ADVagg)
  - Use standard techniques to find maximum benefit set
- Results
  - Accounting for overhead, overlap, optimization helps
  - Many more results in paper

- Future
  - ADVagg accurate? Simplifications valid?
  - Non-iterative approximations for real implementations