Adaptive Software Speculation for Enhancing the Cost-Efficiency of Behavior-Oriented Parallelization

Yunlian Jiang
Xipeng Shen
The College of William and Mary
High-level Parallelism

- Parallel computing is becoming ubiquitous
- High-level parallelism exists in many programs
  - E.g. utilities, interpreters, scientific computations
  - Difficult to parallelize

<table>
<thead>
<tr>
<th>Complexity in the code</th>
<th>Uncertain parallelism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit-level operations,</td>
<td>Example*:</td>
</tr>
<tr>
<td>unrestricted pointers,</td>
<td>while ( s=nextSentence() )</td>
</tr>
<tr>
<td>exception handling,</td>
<td>{ parse(s);</td>
</tr>
<tr>
<td>custom mem. management,</td>
<td>if ( isCommand(s) )</td>
</tr>
<tr>
<td>third-party libraries</td>
<td>updateParsingEnv(s);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>
Software Behavior-Oriented Parallelization [Ding+:PLDI07]

- Speculatively execute programs in parallel
- Efficiently detect dependence during runtime
- But, blind speculation causes cost-inefficiency.
Cost and Speedup

- Speculation Success Rate
- Cost
- SpeedUp

<table>
<thead>
<tr>
<th>Speculation Success Rate</th>
<th>Cost</th>
<th>SpeedUp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.9</td>
<td>0.12</td>
</tr>
<tr>
<td>0.9</td>
<td>0.42</td>
<td>0.12</td>
</tr>
<tr>
<td>0.42</td>
<td>0.12</td>
<td>0.12</td>
</tr>
<tr>
<td>0.12</td>
<td>0.12</td>
<td>0.12</td>
</tr>
</tbody>
</table>
Cost and Speedup

Speculation Success Rate

Cost
SpeedUp
Outline

- Introduction to BOP
- Adaptive BOP
- Experimental Result
- Conclusion
Outline

● Introduction to BOP

● Adaptive BOP

● Experimental Result

● Conclusion
Behavior-Oriented Parallelization (BOP)

- A tool for parallelizing sequential programs
- Need no parallel programming or debugging
- Basic scheme: software speculation
- Correctness protected through runtime system
Basic Scheme of BOP

While (S = ReadSentence())
{
    Parse(S);
    Possibly Parallel Region (PPR)
    BeginPPR();
    Parse(S);
    EndPPR();
}

Two reasons for failed speculation:
1. Dependence violation
2. Spec runs too slow
Problem to tackle

- Cost inefficiency
  - BOP blindly speculates every PPR instance
  - Failed speculation may
    - Cause slowdown to applications
      - Protection overhead
      - Resource (cache, bus) contention
    - Waste computing resources
      - CPU --- multi-programming environment
      - Power --- Mobile computing
Outline

- Introduction to BOP
- Adaptive BOP
- Experimental Result
- Conclusion
Solution: Adaptive Speculation

- Basic strategy
  - Predict profitability of PPR
  - Speculate only likely profitable ones

- Prediction approaches
  - Extended last-value-based
  - Decayed-history-based
Extended last-value-based prediction

- Speculate or not?
  - Speculate only if $\text{PPRsToSkip} == 0$.

- Adjust $\text{PPRsToSkip}$
  - If this PPR is not speculated
    - $\text{PPRsToSkip} - -$
  - On a failed speculation
    - $\text{PPRsToSkip} = \text{NextPenalty}$;
    - $\text{NextPenalty} *= \alpha$; *(increase penalty exponentially)*
  - On a successful speculation
    - $\text{NextPenalty} = 1$; *(reset the penalty on the next failure)*
Extended last-value-based prediction

- PPRsToSkip = 0
- NextPenalty = 1
- $\alpha = 2$

Speculation!
Success
NextPenalty = 1
Extended last-value-based prediction

| : Profitable PPR | : Unprofitable PPR |

- PPRsToSkip = 0
- NextPenalty = 1
- \( \alpha = 2 \)

Speculation!
Failed
PPRsToSkip = 1
NextPenalty = 2
Extended last-value-based prediction

| : Profitable PPR | : Unprofitable PPR |

- PPRsToSkip = 1
- NextPenalty = 2
- $\alpha = 2$

No Speculation!
PPRsToSkip = 0
Extended last-value-based prediction

| : Profitable PPR | : Unprofitable PPR |

- PPRsToSkip = 0
- NextPenalty = 2
- $\alpha = 2$

Speculation! Failed

PPRsToSkip = 2
NextPenalty = 4
Extended last-value-based prediction

| : Profitable PPR | : Unprofitable PPR |

- PPRsToSkip = 2
- NextPenalty = 4
- $\alpha = 2$

No Speculation!
PPRsToSkip = 1
Extended last-value-based prediction

| : Profitable PPR | : Unprofitable PPR |

- PPRsToSkip = 1
- NextPenalty = 4
- $\alpha = 2$

No Speculation!
PPRsToSkip = 0
Extended last-value-based prediction

- Profitable PPR
- Unprofitable PPR

PROMPTsToSkip=0
NextPenalty=4
\( \alpha = 2 \)

Speculation! Success
NextPenalty = 1
Extended last-value-based prediction

| : Profitable PPR | : Unprofitable PPR |

- PPRsToSkip = 0
- NextPenalty = 1
- $\alpha = 2$

Speculation!
Success
NextPenalty = 1
Extended last-value-based prediction

- Limitations
  - Can not keep history well
    - Successful speculation
      → clean history

- Phase changes
Decayed-history-based prediction

- Cumulative gain (CG)
  \[ cg = \gamma^g + (1-\gamma) \times cg \]
  \( g = \begin{cases} 1 : \text{success} \\ 0 : \text{failed} \end{cases} \)

- Expected Profitability (EP)
  \[ EP = cg + \text{SkippedPPRs} \times \beta \]

- Speculate only if \( EP > TH_{EP} \)

* TH_{EP} : threshold of speculation;
* SkippedPPRs: reset to 0 on a success
  increase by 1 on a non-speculated PPR
Decayed-history-based prediction

| : Profitable PPR | : Unprofitable PPR |

- $\text{TH}_{EP} = 0.6$
- $\beta = 0.2$
- $\gamma = 0.5$
- SkippedPPRs = 0
- $cg = 1$
- $EP = 1$

Speculation!
Success

cg = 1
EP = 1
Decayed-history-based prediction

| : Profitable PPR | : Unprofitable PPR |

- $T_{HEP} = 0.6$
- $\beta = 0.2$
- $\gamma = 0.5$
- Skipped PPRs = 0
- $cg = 1$
- $EP = 1$

Speculation! Failed
- $cg = 0.5$
- $EP = 0.5$
Decayed-history-based prediction

| : Profitable PPR | : Unprofitable PPR |

- $\text{T}_\text{EP} = 0.6$
- $\beta = 0.2$
- $\gamma = 0.5$
- SkippedPPRs = 0
- $\text{cg} = 0.5$
- EP = 0.5

No Speculation!
SkippedPPRs = 1
EP = 0.7
Decayed-history-based prediction

| : Profitable PPR | : Unprofitable PPR |

- $TH_{EP} = 0.6$
- $\beta = 0.2$
- $\gamma = 0.5$
- Skipped PPRs = 1
- cg = 0.6
- EP = 0.7

Speculation!

Success

Skipped PPRs = 0

cg = 0.8

EP = 0.8
Decayed-history-based prediction

| : Profitable PPR | : Unprofitable PPR |

- $TH_{EP} = 0.6$
- $\beta = 0.2$
- $\gamma = 0.5$
- Skipped PPRs = 0
- $cg = 0.8$
- $EP = 0.8$

Speculation!
Failed
$cg = 0.4$
$EP = 0.4$
Decayed-history-based prediction

| : Profitable PPR | : Unprofitable PPR |

- $TH_{EP} = 0.6$
- $\beta = 0.2$
- $\gamma = 0.5$
- Skipped PPRs = 0
- $cg = 0.4$
- $EP = 0.4$

No Speculation! Skipped PPRs = 1

EP = 0.6
Decayed-history-based prediction

<table>
<thead>
<tr>
<th>: Profitable PPR</th>
<th>: Unprofitable PPR</th>
</tr>
</thead>
</table>

- $\text{TH}_{\text{EP}} = 0.6$
- $\beta = 0.2$
- $\gamma = 0.5$
- SkippedPPRs = 1
- $cg = 0.4$
- $EP = 0.6$

Speculation!
Success
SkippedPPRs = 0
$EP = 0.7$
Outline

- Introduction
- Adaptive-Algorithms
- Experimental Result
- Conclusion
Experimental Result

- **Prediction Accuracy**
  - Choose the best parameters for the algorithms
  - Evaluate two algorithms

- **Computation Efficiency**
  - Finishing time
  - Time spent on all CPUs (Cost)
Accuracy for Last-value-based

- Adjust non-speculate numbers
  - Non-speculation
    - PPRsToSkip -1
  - Success Speculation
    - NextPenalty= 1
  - Failed Speculation
    - PPRsToSkip = NextPenalty
    - NextPenalty *= $\alpha$
Accuracy for Last-value-based

Accuracy: 81.6

\( \alpha = 1.4 \)
Accuracy for Decayed history based

- $G_{TH}$: gain threshold
- Current state weight
  - $gain^+ \ quota^* \beta$
  - Non-speculative execution
    - $quota++$
  - Speculative execution
    - $gain = \gamma*g + (1-\gamma) * gain$
    - $quota \to 0$
Accuracy for Decay based

- **Cumulative gain** ($CG$)
  \[ cg = \gamma g + (1-\gamma) \cdot cg \]
  \[ g = \begin{cases} 1 : & \text{success} \\ 0 : & \text{failed} \end{cases} \]

- **Expected Profitability** ($EP$)
  \[ EP = cg + \text{SkippedPPRs} \cdot \beta \]

- Speculate only if $EP > TH_{EP}$

* $TH_{EP}$ : threshold of speculation;
* SkippedPPRs: reset to 0 on a success
  increase by 1 on a non-speculated PPR
Accuracy for Decayed history based

- $TH_{EP} = 0.25$
- $\beta = 0.0075$
- $\gamma = 0.4$

$\rightarrow$ Accuracy $= 85.6\%$
Computation Efficiency

- Machine: Intel Pentium-D dual-core processors
- Compiler: gcc4.1
- Benchmarks
  - Gzip, Parser, Reduction
- Metrics
  - Cost
    - Total running time of all the processes
  - Time
    - Finishing time of a program
Efficiency comparison on gzip

![Graph showing cost (s) vs buffer size for different data sizes and methods: seq, org-bop, and adapt-bop.]
Efficiency comparison on gzip

![Graph showing efficiency comparison on gzip](image.png)
Efficiency comparison on parser

![Bar chart showing efficiency comparison between different parsers.](chart.png)

- seq
- org-bop
- adapt-bop

<table>
<thead>
<tr>
<th>Num of Sentences</th>
<th>Cost(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

- bar 1: seq, org-bop, adapt-bop
- bar 2: seq, org-bop, adapt-bop
Efficiency comparison on parser

![Bar chart showing the comparison of different parsers.](chart)

- seq
- org-bop
- adapt-bop

Num of Sentences: 50, 10, 2

Time (S):
- 20
- 15
- 10
- 5
- 0
Efficiency comparison on Reduction

![Graph showing efficiency comparison between seq, org-bop, and adapt-bop methods at different levels of dependence (0%, 10%, 50%, 90%) with cost values on the y-axis (0 to 50 cost(s))]
Efficiency comparison on Reduction

![Bar chart](chart.png)

- **seq**
- **org-bop**
- **adapt-bop**

Time(s)

0 10 20 30

Dependence

0 10% 50% 90%
Outline

- Introduction to BOP
- Adaptive Algorithms
- Experimental Result
- Conclusion
Conclusions

- Failed Speculation is a problem
- Two adaptive algorithms
  - Last-value-based prediction
  - Decayed-history-based prediction
- Performance
  - High accurate prediction
  - Keep fast running speed
  - Reduce cost
Thanks!

Questions?