A Compiler Framework for Speculative Analysis and Optimizations
Jin Lin, Tong Chen, Wei-Chung Hsu, Per-Chung Yew, Roy Dz-Ching Ju (Intel), Tin-Fook Ngai (Intel), Sun Chan (Intel)

Outline
- Motivation
- Speculative Compiler Analysis Framework
  - Speculative Static Single Assignment (SSA) Form
  - Speculative Partial Redundancy Elimination
- Experimental Results
- Conclusions

Control Speculation – Instruction Scheduling

```
 CMP  
  true, probability 90%  
  ld r3 = [r8]  

 (a) control flow graph of a program

 if (c) {
  chk.s r3, recovery  
  next: ...
  }

 (b) speculative version
```

Data Speculation – Instruction Scheduling

```
ld.s r3 = [r8]  
if (c) {
  chk.s r3, recovery  
  next: ...
  }
```

```
dl.a r3 = [r8]  
add ...
st ...
```

```
... = ... + ...
*q = ..
... = *p +...
```

```
add r3, ...
```

(a) original program

(b) speculative version
Control Speculation – Redundancy Elimination

(a) control flow graph of a program

- \ldots = *p
- \text{cond}
- \text{probabilty 90%}
- \ldots = *p

(b) speculative version

- \text{ld } r3 = [r8]
- \text{chk.s } r3, \text{rec}
- \ldots

Data Speculation – Redundancy Elimination

(a) original program

- \ldots = a[i]*b
- *q = ..
- \ldots = a[i]*b

(b) speculative version

- \text{ld.a } r3 = [r8]
- \text{mult } ..
- \text{st }..
- \text{chk.a } r3, \text{recovery}

Observations

- More speculative optimization opportunities exist.
- For effective speculative optimizations, a framework must provide probabilistic aliasing information

Our Approach

- A general compiler framework to support both control speculation and data speculation.
- Extends SSA form to incorporate probabilistic aliasing information, thus enabling more aggressive compiler optimizations such as PRE.
A Framework of Speculative Analysis and Optimizations

Speculative alias and dataflow analysis

Control flow graph (control speculation)

Speculative use-def chain/ SSA form (data speculation)

Control flow graph

Heuristic rules

Edge/path profile

Speculative optimizations
- PRE based optimizations: PRE for expressions, register promotion, global value numbering based redundancy elimination, strength reduction, ...
- Instruction scheduling
- ...

Speculative SSA Form

χ operator: MayMod

µ operator: MayUse

χ<sub>s</sub> operator:
- the variable with χ<sub>s</sub> is very likely to be updated by the corresponding definition statement

µ<sub>s</sub> operator:
- the variable with µ<sub>s</sub> is very likely to be referenced by the indirect reference

Example of Speculative SSA Form

\( a_1 = \ldots \)
\( *p_1 = 4 \)
\( a_2 \leftarrow \chi ( a_1 ) \)
\( b_2 \leftarrow \chi ( b_1 ) \)
\( v_2 \leftarrow \chi ( v_1 ) \)
\( \ldots = a_2 \)
\( a_3 = 4 \)
\( \mu(a), \mu(b), \mu(v) \)
\( \ldots = \#p_1 \)

(a) traditional SSA graph

The points-to set of pointer \( p \) obtained by alias profiling is

\( (b) \) speculative SSA graph

Creation of µ<sub>s</sub> and χ<sub>s</sub> lists

For a given indirect memory reference, we determine whether a variable \( v \) is highly likely modified/used from profiling or compiler heuristics

For such a highly likely mod/used \( v \),
- if \( v \) is in the µ/χ list, update µ/χ to µ<sub>s</sub>/χ<sub>s</sub>
- otherwise (e.g. heap object from profiling), add µ<sub>s</sub>/χ<sub>s</sub> into the µ/χ list
Overview of Traditional SSAPRE
- Phi insertion
- Rename
- Down_safety
- Will_be_available
- Finalize
- Code motion

Steps in Traditional SSAPRE

(a) SSA graph

\[
\begin{align*}
... &= a_1 * c_1 \\
... &= a_1 * c_1 \\
\pi_1 &= ...
\end{align*}
\]

\[
\begin{align*}
v_1 &\leftarrow \chi(v_1) \\
a_2 &\leftarrow \chi(a_1) \\
b_2 &\leftarrow \chi(b_1) \\
... &= c_1 \\
\end{align*}
\]

(b) after rename step

\[
\begin{align*}
t_1 &= a_1 * c_1 \\
... &= t_1 \\
... &= t_1 \\
\pi_1 &= ...
\end{align*}
\]

\[
\begin{align*}
v_1 &\leftarrow \chi(v_1), a_2 &\leftarrow \chi(a_1) \\
b_2 &\leftarrow \chi(b_1) \\
... &= c_1 \\
\end{align*}
\]

(c) after code motion step

Speculative SSAPRE

Major issues:
- How to identify speculative redundant expressions?
- How to generate corresponding checks?
- How to remove unnecessary phi nodes for expressions?

Overview of Speculative PRE
- Phi insertion
- Rename
- Down_safety
- Will_be_available
- Finalize
- Code motion
Enhanced Rename Step to Allow Data Speculation

\[ \begin{align*}
    a_1 &= \ldots \\
    \ldots &= a_1 \ast c_1 \\
    \ldots &= a_1 \ast c_1 \quad \text{[redundant]} \\
    *p_1 &= \ldots \\
    v_2 &= \chi(v_1) \\
    a_2 &= \chi(a_1) \\
    b_2 &= \chi(b_1) \\
    \ldots &= a_2 \ast c_1
\end{align*} \]

(a) traditional renaming

\[ \begin{align*}
    \hat{a}_1 &= \ldots \\
    \ldots &= \hat{a}_1 \ast c_1 \\
    \ldots &= \hat{a}_1 \ast c_1 \\
    \hat{p}_1 &= \ldots \\
    \hat{v}_2 &= \chi(v_1) \\
    \hat{a}_2 &= \chi(a_1) \\
    \hat{b}_2 &= \chi(b_1) \\
    \ldots &= \hat{a}_2 \ast c_1
\end{align*} \]

(b) speculative renaming

Enhanced Rename Step to Allow Data Speculation (Cont’d)

\[ \begin{align*}
    a_1 &= \ldots \\
    \ldots &= a_1 \ast c_1 \\
    \ldots &= a_1 \ast c_1 \quad \text{[redundant]} \\
    *p_1 &= \ldots \\
    v_2 &= \chi(v_1) \\
    a_2 &= \chi(a_1) \\
    b_2 &= \chi(b_1) \\
    \ldots &= a_2 \ast c_1
\end{align*} \]

(a) traditional renaming

\[ \begin{align*}
    \hat{a}_1 &= \ldots \\
    \ldots &= \hat{a}_1 \ast c_1 \\
    \ldots &= \hat{a}_1 \ast c_1 \\
    \hat{p}_1 &= \ldots \\
    \hat{v}_2 &= \chi(v_1) \\
    \hat{a}_2 &= \chi(a_1) \\
    \hat{b}_2 &= \chi(b_1) \\
    \ldots &= \hat{a}_2 \ast c_1 \quad \text{[speculative redundant]}
\end{align*} \]

(b) speculative renaming

Overview of Speculative PRE

- **Phi insertion**
- **Rename**
- **Down_safety**
- **Will_be_available**
- **Finalize**
- **Code motion**

Enhanced Code Motion Step

\[ \begin{align*}
    \ldots &= a_1 \ast c_1 \\
    \ldots &= a_1 \ast c_1 \quad \text{[redundant]} \\
    *p_1 &= \ldots \\
    v_2 &= \chi(v_1) \\
    a_2 &= \chi(a_1) \\
    b_2 &= \chi(b_1) \\
    \ldots &= a_2 \ast c_1 \quad \text{[speculative redundant]}
\end{align*} \]

(a) Before Code Motion

\[ \begin{align*}
    t_1 &= a_1 \ast c_1 \quad \text{(advance load for a)} \\
    \ldots &= t_1 \\
    \ldots &= \hat{a}_1 \ast c_1 \quad \text{[speculative redundant]} \\
    \hat{p}_1 &= \ldots \\
    \hat{v}_2 &= \chi(v_1) \\
    \hat{a}_2 &= \chi(a_1) \\
    \hat{b}_2 &= \chi(b_1) \\
    \ldots &= \hat{a}_2 \ast c_1 \quad \text{[check for a]}
\end{align*} \]

(b) Final Output
Experimental Environment

- Open Research Compiler v2.0
- Benchmark: Spec2000 C programs
- Platform:
  - HP i2000, 700MHz Itanium processor, 1GB SDRAM
  - Redhat Linux 7.1
- Pfmon v1.1

Experimental Data

- Speculative SSAPRE based register promotion in our speculative compiler analysis framework
- Measured performance
- Effectiveness
- Potential performance

Performance Improvement of Speculative Register Promotion Based on Alias Profile

<table>
<thead>
<tr>
<th></th>
<th>ammp</th>
<th>art</th>
<th>equake</th>
<th>bzip2</th>
<th>gzip</th>
<th>mcf</th>
<th>parser</th>
<th>twolf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improvement percentage</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU cycle</td>
<td>16%</td>
<td>14%</td>
<td>12%</td>
<td>10%</td>
<td>8%</td>
<td>6%</td>
<td>4%</td>
<td>2%</td>
</tr>
<tr>
<td>Data access cycle</td>
<td>2%</td>
<td>4%</td>
<td>6%</td>
<td>8%</td>
<td>10%</td>
<td>12%</td>
<td>14%</td>
<td>16%</td>
</tr>
<tr>
<td>Loads retired</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Effectiveness of Speculative Register Promotion

- ldc_chk as a percentage of retired loads
- mis-speculation

aggressively compiled using --O3 with type based alias analysis
Potential of Speculative Register Promotion

Related Work

- Control and data speculation used in instruction scheduling [Ju. et al. 2000]
- Partial redundancy elimination (PRE)
  - Standard SSAPRE [Chow. et al. 1997]
  - Control Speculation in PRE [Lo et al. 1998, Bodik et al. 1998]

Conclusions

- A general compiler analysis framework is proposed to incorporate speculative information for both data and control speculation
- A speculative SSA form is proposed to represent probabilistic aliasing information
- Speculative optimizations such as speculative SSAPRE can be implemented with good performance in this framework

END