Speculative Register Promotion
Using Advanced Load Address Table (ALAT)

Jin Lin, Tong Chen, Wei-Chung Hsu, Pen-Chung Yew

http://www.cs.umn.edu/Agassiz

Outline

- Motivation
- Scheme of speculative register promotion using ALAT
- Algorithm of speculative register promotion
- Performance evaluation
- Related work
- Conclusions

Motivation

- Register promotion is usually hampered by imprecise alias information because complete alias analysis is very costly and difficult
- Data speculation can compensate for imprecise alias information
- Can we support aggressive register promotion through data speculation?

Motivation Example

- Register promotion is usually hampered by imprecise alias information because complete alias analysis is very costly and difficult
- Data speculation can compensate for imprecise alias information
- Can we support aggressive register promotion through data speculation?
Our approach

- Use alias profile or compiler heuristics to obtain approximated alias information
- Use data speculation to verify such alias information at run time
- Use Advance Load Address Table (ALAT) in IA64 to provide the necessary support for data speculation

Background of ALAT in IA64

Advanced loads

- Insert advanced loads (ld,c) to prefetch data (store in ALAT)
- Use check data instruction (ld,c) in place of original load
- If memory contents has changed, perform real load
- Advanced loads do not defer exceptions (e.g., page faults)

Regular load:

- ld r1 = [p]  
  add r3 = r1, 1
  ld a r1 = [p]  
  add r4 = r1, 3
  st [r1] = r4

Advanced load:

- ld,a r2 = [p]  
  st [r1] = r2
  ld,c r1 = [p]  
  add r4 = r1, 3
  ld,c.clr r1 = [p]

Examples

a. read after read 
  b. read after write 
  c. multiple redundant loads

Scheme for Speculative Register Promotion

- Use ld,a for the first load
- Check the following loads
  - Scheme a: use ld,c for the following reads.
  - Scheme b: use chk,a for the following reads and the corresponding recovery code.

Examples

- p1:  
  ld a r1 = [p]  
  ld [r1] = r1

- p2:  
  add r3 = r1, 1
  ld a r1 = [p]  
  add r4 = r1, 3
  st [r1] = r4

- p3:  
  ld,c r1 = [p]
  ld,c.clr r1 = [p]
  ld,c.nl r1 = [p]
Compiler Support for Speculative Register Promotion

- Enhanced SSA form with the notion of data speculation
- SSA form for indirect memory reference
  - $\chi$ operator: MayMod
  - $\mu$ operator: MayUse
- Speculative SSA form
  - $\chi_s$ operator: the variable in $\chi_s$ is unlikely to be updated by the corresponding definition statement
  - $\mu_s$ operator: the variable in $\mu_s$ is unlikely to be referenced by the indirect reference

Speculative SSA Form According To Alias Profiling

\[ *p = \ \mu(b_1) \]
\[ b_2 \leftarrow \chi(b_1) \quad \mu(a_1) \]
\[ a_2 \leftarrow \chi(a_1) \quad \mu(v_1) \]
\[ v_2 \leftarrow \chi(v_1) = *p \]

The two examples assume that the points-to set of $p$ generated by compiler is \{a, b\}, the points-to set of $p$ obtained from alias profiling is \{b\}. $v$ is the virtual variable for $*p$. $a_j$ stands for version $j$ of the variable $a$.

Overview of Speculative Register Promotion

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

* Based on SSAPRE [Kennedy, Chan, Liu, Lo, Tu ACM TOLPAS '99]

Enhanced Rename

The target set of $*p$ generated by the compiler is \{a, b\} and $v$ is the virtual variable for $*p$. The target set of $*p$ generated by the alias profiling is \{b\}.

\[ \ldots = a_1 \quad \ldots = a_1 \]
\[ \ldots = a_1 \quad \ldots = a_1 \]
\[ v_2 \leftarrow \chi(v_1) \quad v_2 \leftarrow \chi(v_1) \]
\[ a_2 \leftarrow \chi(a_1) \quad a_2 \leftarrow \chi(a_1) \]
\[ b_2 \leftarrow \chi(b_1) \quad b_2 \leftarrow \chi(b_1) \]
\[ \ldots = a_1 \quad \ldots = a_1 \quad \text{<speculative>} \]

a) Traditional Renaming  (b) Speculative Renaming
Overview of Speculative Register Promotion*

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

* Based on SSAPRE

[Kennedy, Chan, Liu, Lo, Tu ACM TOLPAS '99]

Example of Speculative Code Motion

(a) Before Code Motion

... = a₁
"p₁ = ...
\(v₄ \leftarrow \chi_s(v₃)\)
\(a₂ \leftarrow \chi_s(a₁)\)
\(b₄ \leftarrow \chi_s(b₃)\)
... = a₁ <speculative>

(b) Final Output

... = a₁ (ld.a)
"p₁ = ...
\(v₄ \leftarrow \chi_s(v₃)\)
\(a₂ \leftarrow \chi_s(a₁)\)
\(b₄ \leftarrow \chi_s(b₃)\)
... = a₁ (ld.c)
... = a₁ (ld.c)

Implementation

- Open Research Compiler v1.1
- Benchmark
  - Spec2000 C programs
- Platform
  - HP i2000, 733 MHz Itanium processor, 1GB SDRAM
  - Redhat Linux v7.1
- Pfmon v1.1

Performance Improvement of Speculative Register Promotion

-2.0% 0.0% 2.0% 4.0% 6.0% 8.0% 10.0% 12.0% 14.0%

Improvement percentage

cpu cycle
data access cycle
loads retired
Effectiveness of Speculative Register Promotion

- Cregs
- SLAT
- Run Time Disambiguation
- ALAT
  - No impact on regular loads
  - Easier to enforce correctness
    - No need to check full address
    - ALAT entries can be replaced due to overflow, conflicts or context switches

Conclusions
- Data speculation can effectively support speculative register promotion with imprecise alias information from alias profiling
- This approach can be extended to support other compiler optimizations such as speculative Partial Redundancy Elimination*

* [A Compiler Framework for Speculative Analysis and Optimizations, appear in PLDI 2003]