Detecting Global Stride Locality in Value Streams

Huiyang Zhou, Jill Flanagan, Thomas M. Conte

TINKER Research Group
Department of Electrical & Computer Engineering
North Carolina State University
Introduction

• This paper is about predicting values …

• A code example (from the benchmark parser, function is_CON_word)

```assembly
....
00400740 lw $v0[2],0($v1[3])
00400748 sw $v0[2],0($s8[30])
00400750 lw $v0[2],0($s8[30])  // the insn to be predicted
00400758 bne $v0[2],$zero[0],00400768
00400758 bne $v0[2],$zero[0],00400768
....
....
00400798 lw $v0[2],0($s8[30])
004007a0 lw $v1[3],12($v0[2])
004007a8 sw $v1[3],0($s8[30])
004007b0 j 00400750
....
```

local value history from value profile: xx528, xx840, 0, xx792, 0, xx720, 0, xx816, xx768, xx744, xx696, xx624, xx672, …
Hard-to-Predict Values

Hard to predict with (local) value predictors:
- DFCM [Goeman et al.]: 2% accuracy;
- (local) stride [Lipasti et al.]: 4% accuracy.
Global Value Locality

- Another look at the code

```
00400740  lw $v0[2],0($v1[3])  //the correlated load
00400748  sw $v0[2],0($s8[30])
00400750  lw $v0[2],0($s8[30])  //the instruction to be predicted
00400758  bne $v0[2],$zero[0],00400768
00400798  lw $v0[2],0($s8[30])
004007a0  lw $v1[3],12($v0[2])  //the correlated load
004007a8  sw $v1[3],0($s8[30])
004007b0  j 00400750
```
Global Stride Locality

Control flow graph

```
00400740 lw $v0[2],0($v1[3])  // the correlated load
00400748 sw $v0[2],0($s8[30])
004007a0 lw $v1[3],12($v0[2])  // the correlated load
004007a8 sw $v1[3],0($s8[30])
004007b0 j 00400750
```

```
00400770 lw $v0[2],0($s8[30])  // the instruction to be predicted
00400778 bne $v0[2],$zero[0],00400768
```

100% prediction accuracy is possible if we can use the results of correlated instructions.
Global Value History

• Global value history:
  - If the values produced by the dynamic instruction stream are labeled $x_1, x_2, \ldots, x_N$, then the ordered data sequence $(x_1, x_2, \ldots, x_N)$ is the global value history of order (i.e., length) $N$.

• Local value history
  - The value sequence produced by the same instruction to be predicted.
Global Value Locality

• Similar to local value predictability [Sazeides and Smith]

• Global context locality
  – Previous instruction (PI) predictor [Nakra et. al.]
  – Dynamic dataflow-inherited speculative context (DDISC) predictor [Thomas & Franklin]

• Global computational locality

\[ x_N = a_{N-1} x_{N-1} + a_{N-2} x_{N-2} + L + a_1 x_1 + a_0. \]

Similar formulation to “Perceptron branch predictor” [Jimenez & Lin]
Global Stride Locality

- **Global stride locality**
  \[ x_N = x_{N-k} + a_0. \]
  
  \( k \) and \( a_0 \) are run-time variables.

- **The previous example:**  
  \[ x_N = x_{N-1} \]

```assembly
... 00400740 lw $v0[2],0($v1[3])  
     // the correlated load
00400748 sw $v0[2],0($s8[30])
...
  004007a0 lw $v1[3],12($v0[2])  
     //the correlated load
004007a8 sw $v1[3],0($s8[30])
004007b0 j 00400750

00400750 lw $v0[2],0($s8[30])
     //the instruction to be predicted
00400758 bne $v0[2],$zero[0],00400768
```
Global Stride Locality In A Program

• Global stride locality in the code

... Define (e.g., load ra, rb, rc)  // load value is hard to predict
...

Explicit Use (e.g., add rx, ra, #constant) // the dest of add can be
     // predicted using the value of the define instruction results (ra)
...

Explicit Use (e.g., sub rx, ra, rd)  // the dest of sub can be predicted
     // if rd has strong repeating patterns
...

Implicit Use (e.g., load rx, ry, rz) // Implicit use through the
     // memory (e.g., spilling and filling; reloading)
Global Stride Locality In A Program (cont.)

- Global stride locality embedded in the data structure

```
struct string_list {
    struct string_list * next;
    char * string;
}
```

Near constant stride between the two addresses (*next, *string) if the two fields (next, string) are allocated and referenced in the same order. [Serrano & Wu]
Outline

• Introduction of global stride locality
• GDiff predictor
• Value Delay
• Speculative Value Queue
• Hybrid Value Queue
• Potential uses of gDiff predictor
• Summary
The GDiff Value Predictor

Two major structures

- Global value queue (gvQ)
  
  completing instructions’ results

- Prediction table

- PC

 Stored differences and distance
Example

• A code example

...  
a: load r1, r2, #20  
b:  
c:  
d: add r3, r1, #4  
...

Instruction \( a \) produces \((1, 8, 3, 2, \ldots)\)
Instruction \( d \) generates \((5, 12, 7, 6, \ldots)\)
How GDiff Works

Instruction $a$ produces (1, 8, 3, 2, …)
Instruction $d$ generates (5, 12, 7, 6, …)
The GDiff Value Predictor

\[ x_N = x_{N-k} + a_0. \]
Results: Exploiting Global Stride Value Locality

Value prediction accuracy

- stridge
- DFCM
- gDiff (queue size = 8)

Based on trace simulation (sim-profile) and predicting all value producing insns
Stride, gDiff, DFCM L1: 8k entry pc-indexed tagless table; DFCM L2: 64K entry table
Outline

• Introduction of global stride locality
• GDiff predictor
• Value Delay
• Speculative Value Queue
• Hybrid Value Queue
• Potential uses of GDiff predictor
• Summary
Value Delay

- When the prediction of an instruction is being made, the correlated instruction has not finished

\[\text{....} \]
A. `lw $v0[2],0($v1[3])` //the correlated load
B. `sw $v0[2],0($s8[30])`
C. `lw $v0[2],0($s8[30])` //the instruction to be predicted
D. `bne $v0[2],$zero[0],00400768`

\[\text{....} \]
Value Delay Impact

Model value delay as T values: a prediction can not use the previous T insns’ results.

Trace simulation using sim-profile
Outline

• Introduction of global stride locality
• GDiff predictor
• Value Delay
• Speculative Value Queue
• Hybrid Value Queue
• Potential uses of GDiff predictor
• Summary
Reducing the Value Delay

Gdiff with speculative value queue (SGVQ)
Results of SGVQ

Prediction accuracy and coverage of the gDiff predictor with SGVQ and the local stride predictor

Worse than the simple local stride predictor. Why?
Problems with Speculative GDiff

- Though value delay is reduced.
- The results are out-of-order

Completion order

Cache miss

Cache hit
Outline

- Introduction of global stride locality
- GDiff predictor
- Value Delay
- Speculative Value Queue
- Hybrid Value Queue
- Potential uses of gDiff predictor
- Summary
The Hybrid GDIf**f Value Predictor

```
<table>
<thead>
<tr>
<th>Fetch</th>
<th>Dispatch</th>
<th>Issue</th>
<th>Reg Read</th>
<th>Execution</th>
<th>Write Back</th>
<th>Retire</th>
</tr>
</thead>
</table>
```

Execution pipeline

- GDIf**f prediction Table
- Global value queue
- GDIf**f predictor
- Local stride predictor
- local stride prediction

Update (out-of-order)

Key: construct the global value history *in-order (fetch/dispatch order)*; combine with a different value predictor.
Results of Hybrid GDiff

Prediction accuracy and coverage of gDiff and local predictors

(Local context uses DFCM)
Potential Uses of GDiff

- Predicting load addresses.
  - Forming the global value history only with load addresses.
  - Predict next address based on previous addresses.

- Predicting addresses of missing loads only
  - Forming the global value history only with addresses of missing loads.
  - Predict next address based on previous addresses.

- Predicting values to break true data dependence chain.

- And ...
Predicting Addresses Using Hybrid GDiff

Predictability of load address stream

4-way 256k entry Markov address predictor uses tag for confidence
Predicting Address of Missing Loads

Predictability of addresses of missing loads

- ls_m_accu
- gs_m_accu
- markov_m_accu
- ls_m_cov
- gs_m_cov
- markov_m_cov

Bar chart showing the predictability of addresses for different missing loads using various models with different coverage metrics.
Using GDiff to Break Data Dependencies

![Speedups of gDiff and local value prediction](chart.png)

4/64 issue out-of-order model
Summary

- Global stride locality
- The Gdiff value predictor
- Value delay issue
- Reduce the value delay
- In-order global value history

Potential Uses:
- What to put into the global value queue (i.e., global value history)
- How to utilize the prediction
Thank you and Questions?