Wrong Path Events
and Their Application to
Early Misprediction Detection and Recovery

David N. Armstrong
Hyesoon Kim
Onur Mutlu
Yale N. Patt

University of Texas at Austin
Motivation

• Branch predictors are not perfect.

• Branch misprediction penalty causes significant performance degradation.

• We would like to reduce the misprediction penalty by detecting the misprediction and initiating recovery before the mispredicted branch is resolved.
Branch Misprediction Resolution Latency

![Chart showing cycles from branch issue to recovery for various benchmarks. The x-axis represents different benchmarks: gzip, vpr, gcc, mcf, crafty, parser, eon, perlbench, gap, vortex, bzip2, twolf, and amean. The y-axis represents cycles from branch issue to recovery. The mcf benchmark has the highest latency at 157 cycles, while bzip2 has the lowest at 36 cycles. The average (amean) latency is also shown.]
Performance Potential of Early Misprediction Recovery

![Bar chart showing IPC Improvement Over Baseline (%) for various benchmarks: gzip, vpr, gcc, mcf, crafty, parser, eon, perl, gap, vortex, bzip2, twolf, hmean. The chart indicates varying degrees of improvement with some benchmarks showing significant gains. The highest improvement is 11.7%.](image-url)
An Observation and A Question

• In an out-of-order processor, some instructions are executed on the mispredicted path (wrong-path instructions).

• Is the behavior of wrong-path instructions different from the behavior of correct-path instructions?
  – If so, we can use the difference in behavior for early misprediction detection and recovery.
Talk Outline

- Concept of Wrong Path Events
- Types of Wrong Path Events
- Experimental Evaluation
- Realistic Early Misprediction Recovery
- Shortcomings and Future Research
- Conclusions
What is a Wrong Path Event?

An instance of illegal or unusual behavior that is more likely to occur on the wrong path than on the correct path.

Wrong Path Event = WPE

Probability (wrong path | WPE) \sim 1
Why Does a WPE Occur?

• A wrong-path instruction may be executed before the mispredicted branch is executed.
  – Because the mispredicted branch may be dependent on a long-latency instruction.

• The wrong-path instruction may consume a data value that is not properly initialized.
WPE Example from eon: NULL pointer dereference

1:   for (int i=0 ; i< length(); i++) {
2:       structure *ptr = array[i];
3:       if (ptr->x) {
4:           // . . .
5:       }
6:   }

Beginning of the loop

Array boundary

i = 0

Array of pointers to structs

x8ABCD0  xEFF8B0  x0  x0

1:  for (int i=0; i< length(); i++) {
2:      structure *ptr = array[i];
3:      if (ptr->x) {
4:          // . . .
5:      }
6:  }

1:  for (int i=0; i< length(); i++) {
2:      structure *ptr = array[i];
3:      if (ptr->x) {
4:          // . . .
5:      }
6:  }
First iteration

Array of pointers to structs

\[ i = 0 \]
\[ \text{ptr} = \text{x8ABCD0} \]

Array boundary

1: for (int i=0; i< length(); i++) {
2: structure *ptr = array[i];
3: if (ptr->x) {
4:     // . . .
5: }
6: }

x8ABCD0 | xEFF8B0 | x0 | x0
Array of pointers to structs

```
1 : for (int i=0 ; i< length(); i++) {
2 :     structure *ptr = array[i];
3 :     if (ptr->x) {
4 :         //...
5 :     }
6 : }
```
Loop branch correctly predicted

Array boundary

i = 1

Array of pointers to structs

x8ABCD0  xEFF8B0  x0  x0

1: for (int i=0; i< length(); i++) {
2:     structure *ptr = array[i];
3:     if (ptr->x) {
4:         // . . .
5:     }
6: }

Second iteration

Array boundary

i = 1
ptr = xEFF8B0

Array of pointers to structs

```
1:   for (int i=0; i< length(); i++) {
2:       structure *ptr = array[i];
3:       if (ptr->x) {
4:           // . . .
5:       }
6:   }
```
Second iteration

Array boundary

$i = 1$
$ptr = xEFF8B0$

Array of pointers to structs

```
1 : for (int i=0 ; i< length(); i++) {
2 :     structure *ptr = array[i];
3 :     if (ptr->x) {
4 :         // . . .
5 :     }
6 : }
```
Loop exit branch mispredicted

Array of pointers to structs

```
for (int i=0 ; i< length(); i++) {
    structure *ptr = array[i];
    if (ptr->x) {
        // . . .
    }
}
```
Third iteration on wrong path

Array boundary

Array of pointers to structs

```
1:   for (int i = 0; i < length(); i++) {
2:       structre *ptr = array[i];
3:       if (ptr->x) {
4:           // . . .
5:       }
6:   }
```
Wrong Path Event

Array of pointers to structs

x8ABCD0 xEFF8B0 x0 x0

Array boundary

i = 2
ptr = 0

NULL pointer dereference!

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }

1: for (int i=0; i< length(); i++) {
2:   structure *ptr = array[i];
3:   if (ptr->x) {
4:     // . . .
5:   }
6: }
Types of WPEs

- Due to memory instructions
  - NULL pointer dereference
  - Write to read-only page
  - Unaligned access (illegal in the Alpha ISA)
  - Access to an address out of segment range
  - Data access to code segment
  - Multiple concurrent TLB misses
Types of WPEs (continued)

- Due to control-flow instructions
  - Misprediction under misprediction
    - If three branches are executed and resolved as mispredicts while there are older unresolved branches in the processor, it is almost certain that one of the older unresolved branches is mispredicted.
  - Return address stack underflow
  - Unaligned instruction fetch address (illegal in Alpha)

- Due to arithmetic instructions
  - Some arithmetic exceptions
    - e.g. Divide by zero
Experimental Evaluation
Two Empirical Questions

1. How often do WPEs occur?

2. When do WPEs occur on the wrong path?
Simulation Methodology

- Execution-driven Alpha ISA simulator
  - Faithful modeling of wrong-path execution and wrong-path misprediction recoveries
- 8-wide out-of-order processor
- 256-entry instruction window
- Large and accurate branch predictors
  - 64K-entry gshare and 64K-entry PAs hybrid
  - 64K-entry target cache for indirect branches
  - 64-entry return address stack
- Minimum 30-cycle branch misprediction penalty
- Minimum 500-cycle memory latency, 1 MB L2 cache
- SPEC 2000 integer benchmarks
Mispredictions Resulting in WPEs
Distribution of WPEs

- Misprediction Under Misprediction
- Data Access Out Of Segment Range
- NULL Pointer Dereference
- Unaligned Data Access
- Multiple Concurrent TLB Misses

Graph showing the percentage of wrong path events for various programs.
When do WPEs Occur?
Early Misprediction Recovery Using WPEs
Early Misprediction Recovery

- Once a WPE is detected, there may be:
  - no older unresolved branch in the window
    - Do nothing (false alarm)
  - one older unresolved branch in the window
    - The processor initiates early recovery for that branch, guessing that it is mispredicted.
  - multiple older unresolved branches in the window
    - May be different instances of the same static branch.

- To initiate early misprediction recovery, we need a mechanism that decides which branch is the *oldest mispredicted branch* in the window.
A Realistic Recovery Mechanism

- The **distance in the instruction window** between the WPE-generating instruction and the oldest mispredicted branch is predictable.

- The first time a WPE is encountered, the processor records the distance between the WPE-generating instruction and the mispredicted branch.

- The next time the same instruction generates a WPE, the processor predicts that the branch that has the same distance to the WPE-generating instruction is mispredicted.
Ld C Generates a WPE

<table>
<thead>
<tr>
<th>Instr.</th>
<th>SeqNo</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Br A1</td>
<td>10</td>
<td>PC A</td>
</tr>
<tr>
<td>Br A2</td>
<td>20</td>
<td>PC A</td>
</tr>
<tr>
<td>Ld C</td>
<td>40</td>
<td>PC C</td>
</tr>
</tbody>
</table>

NULL pointer dereference
## WPE is Recorded

<table>
<thead>
<tr>
<th>Instr.</th>
<th>SeqNo</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Br A1</td>
<td>10</td>
<td>PC A</td>
</tr>
<tr>
<td>Br A2</td>
<td>20</td>
<td>PC A</td>
</tr>
<tr>
<td>Ld C</td>
<td>40</td>
<td>PC C</td>
</tr>
</tbody>
</table>

### WPE Record

<table>
<thead>
<tr>
<th>SeqNo</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>PC C</td>
</tr>
</tbody>
</table>
Br A1 Resolves as a Mispredict

<table>
<thead>
<tr>
<th>Instr.</th>
<th>SeqNo</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Br A1</td>
<td>10</td>
<td>PC A</td>
</tr>
<tr>
<td>Br A2</td>
<td>20</td>
<td>PC A</td>
</tr>
<tr>
<td>Ld C</td>
<td>40</td>
<td>PC C</td>
</tr>
</tbody>
</table>

**WPE Record**

<table>
<thead>
<tr>
<th>SeqNo</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>PC C</td>
</tr>
</tbody>
</table>
Distance is Recorded in a Table

<table>
<thead>
<tr>
<th>Valid</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
</tr>
</tbody>
</table>

PC C → XOR → Global History
Ld C Again Generates a WPE

<table>
<thead>
<tr>
<th>Instr.</th>
<th>SeqNo</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Br A1</td>
<td>25</td>
<td>PC A</td>
</tr>
<tr>
<td>Br A2</td>
<td>35</td>
<td>PC A</td>
</tr>
<tr>
<td>Ld C</td>
<td>55</td>
<td>PC C</td>
</tr>
</tbody>
</table>

- NULL pointer dereference
Distance Table is Accessed

<table>
<thead>
<tr>
<th>Valid</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Br A1 is Predicted To Be a Mispredict

<table>
<thead>
<tr>
<th>Instr.</th>
<th>SeqNo</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Br A1</td>
<td>25</td>
<td>PC A</td>
</tr>
<tr>
<td>Br A2</td>
<td>35</td>
<td>PC A</td>
</tr>
<tr>
<td>Ld C</td>
<td>55</td>
<td>PC C</td>
</tr>
</tbody>
</table>

Distance = 30
Three Issues in Realistic Recovery

- Recovery may be initiated for a correctly-predicted correct-path branch.
  - Happens for 3.8% of WPE detections.

- A WPE may occur on the correct path!
  - Happens rarely (0.05% of all WPEs).
  - We see no recovery initiated in this case.

- Early recovery for indirect branches
  - The processor needs to predict a new target address.
  - Target addresses can also be recorded in the distance prediction table.
IPC Improvement

![Graph showing the percent IPC improvement over baseline for various benchmarks, with bars for Realistic Early Recovery and Optimal Early Recovery.]
Shortcomings

1. WPEs do not occur frequently. Their coverage of mispredicted branches is low.

2. WPEs do not occur early enough.

3. Staying on the wrong path a few more cycles is sometimes more beneficial for performance than recovering early.
Future Research Directions

• Finding/generating new types of WPEs
  – Can the compiler help?
  – Better understanding of the differences between wrong path and correct path needed.

• Identifying what makes a particular wrong-path episode beneficial for performance
  – Understanding the trade-offs related to the wrong path would be useful in deciding when to initiate early recovery.

• What else can WPEs be used for?
  – Other kinds of speculation?
  – Energy savings?
Conclusions

- Wrong-path instructions sometimes exhibit illegal and unusual behavior, called *wrong path events (WPEs)*.

- WPEs can be used to guess that the processor is on the wrong path and to initiate *early misprediction recovery*.

- A realistic and accurate early misprediction recovery mechanism is described.

- Shortcomings of WPEs are discussed.

- Future research should focus on finding new WPEs.
Related Work

• Glew proposed the use of *bad memory addresses* and illegal instructions as strong indicators of branch mispredictions [Wisc 00].

• Jacobsen et al. explored branch confidence estimation to predict the likelihood that a branch is mispredicted [ISCA 96].

• Manne et al. used branch confidence to gate the pipeline when there is a high likelihood that the processor is on the wrong path [ISCA 98].

• Jiménez et al. proposed an overriding predictor scheme, where a more accurate slow predictor overrides the prediction made by a less accurate fast predictor [MICRO 00].

• Falcón et al. proposed the use of predictions made for younger branches to re-evaluate the prediction made for an older branch [ISCA 04].
Contributions

1. A new idea in branch prediction.

2. We observe that wrong-path instructions exhibit illegal and unusual behavior, called wrong path events (WPEs).

3. We describe a novel mechanism to trigger early misprediction detection and recovery using WPEs.

4. We analyze and discuss the shortcomings of WPEs and propose new research areas to address these shortcomings.
Distribution of Cycles Between WPE and Branch Resolution
Distance Predictor Accuracy

![Bar chart showing distance predictor accuracy across various applications. The chart includes categories such as Incorrect (Recovery on correct path), Incorrect (Recovery on wrong path), Incorrect (No recovery), No prediction (Invalid distance), Correct prediction (Multiple unresolved branches), and Correct prediction (Single unresolved branch). Each category is represented by different colors and patterns, and the percentage of distance predictor outcomes is shown for each application.]
WPE and Misprediction Rate

![Graph showing WPEs and Branch Mispredictions per 1000 Instructions for various benchmarks such as gzip, vpr, gcc, mcf, crafty, parser, eon, perl, gap, vortex, bzip2, twolf, and amean.]