Inferred Call Path Profiling (ICPP)

Todd Mytkowicz, Devin Coughlin and Amer Diwan
University of Colorado at Boulder
Call Paths vs Calling Context

**Call Paths:** Sequence of calls
main → A → … → foo

**Calling Context:** Value that represents sequence of calls
Where we want To be!

Hardware Counters

Software Instrumentation
Bond and McKinley's Probabilistic Calling Context

void A() {
    ...
    foo();
    ...
}

main
A
B
foo
Bond and McKinley's Probabilistic Calling Context

```
void A() {
    int temp = V;
    ...
    V = f(temp, cs_1);
    cs_1: foo();
    ...
}
```

\( V \) at foo provides calling context
Context without computation?

**Stack Height:** Number of bytes from main to foo

foo with stack height of 20 bytes: main → A → foo

foo with stack height of 16 bytes: main → B → foo

*SP* at foo provides calling context
Inferring call paths

What is the call path?

- program-counter = foo
- stack height = 48 bytes
PC/SP is precise 66% of the time
Problem: Ambiguity

- PC/SP maps to more than one call path

Remove ambiguity by changing height of one call path
Activation Record Resizing

1. B +8
2. A +8
3. D +8
   (D, 24)

1. B +8
2. E +8
3. D +8
   (D, 24)

ARR

1. B +8
2. A +8
3. D +8
   (D, 24)

1. B +8
2. E +12
3. D +8
   (D, 28)
ARR via the Frame Pointer

Before ARR:

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>402de8:</td>
<td>push %rbp</td>
</tr>
<tr>
<td>402de9:</td>
<td>mov %rsp, %rbp</td>
</tr>
<tr>
<td>402dec:</td>
<td>sub $0x30, %rsp</td>
</tr>
</tbody>
</table>

After ARR:

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>402de8:</td>
<td>push %rbp</td>
</tr>
<tr>
<td>402de9:</td>
<td>mov %rsp, %rbp</td>
</tr>
<tr>
<td>402dec:</td>
<td>sub $0x40, %rsp</td>
</tr>
</tbody>
</table>
Disambiguation: Random Search

- Produce PC/SP to call path map
- Find random conflict
- Affect height of one path
  - accept if % ambiguity goes down
- Repeat until done
Experimental Methodology

- Intel ICC compiler on 2.4GHz workstation
- SPEC C/C++ CPU 2006
- Offline: train and evaluate on *same* data
  - SPEC train input set
- Online: train and evaluate on *different* data
  - SPEC train and ref input sets
Results: Offline Scenario

C++: 88% precise

C: 96% precise
Across SPEC: 5-precise or less = 99%
Results: Online Scenario

• Unable to find PC/SP in train map
  • 23% for C++
  • 32% for C

• Of those PC/SP pairs in our train map:
  • 80% precise for the C++ programs
  • 91% precise for the C programs
  • 92% PC/SP pairs are 5-precise or less
  • 2% of the time: incorrect
Negligible overhead (geomean = 0.2%)
Some Space overhead but we can fix
Low Cost
Low Amount of Context

High Amount of Context
High Cost

ICPP
Hardware Counters

Software Instrumentation