ESTABLISHING A BASE OF TRUST WITH PERFORMANCE COUNTERS FOR ENTERPRISE WORKLOADS

Andrzej Nowak (CERN / EPFL), Ahmad Yasin, Avi Mendelson and Willy Zwaenepoel

USENIX 2015, July 10th
Can you get accurate basic block level profiles with Event Based Sampling?
Event Based Sampling - EBS

Instruction Pointer

Executed program

Caused event

Counter (+1)

Performance monitoring interrupt

Resume

PMI service routine

Logged IP sample

HARDWARE

USERLAND

OS KERNEL
- Overflow causes PMI
- This takes N cycles to propagate
- Sample is taken elsewhere

Modelled after S. Eranian / D. Levinthal
OPEN ISSUES

- Capabilities of different architectures
- Characteristics of different methods
- Can software improvements help?
- “known effects” - a problem in practice? Kernels vs. apps
- Remedies – quantitative improvements?

- Can you get accurate basic block level profiles with EBS?
METHODS SURVEYED

Classic
- Widely used
- 1 BB / sample

Precise
- Newer hardware
  - PEBS/PDIR/IBS
- Reduces “known effects”
- 1 BB / sample

Last Branch Records
- Leverage LBR
- Sample on “taken branches”
- Multiple BBs / sample
METHODS SURVEYED

Classic
- WSM: INST_RETIRED
- AMD: RETIRED_INSTRUCTIONS
- IVB: INST_RETIRED

Precise
- WSM: INST_RETIRED (precise)
- AMD: IBS
- IVB: INST_RETIRED:PREC_DIST

Last Branch Records
- WSM: BR_INST_EXEC:TAKEN
- AMD: N/A
- IVB: BR_INST_RETIRED:NEAR_TAKEN

10 Jul 2015
## How Last Branch Records Work

### Last Branch Record (Intel)

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

```assembly
400982: mov  rsi,r15
400985: mov  rdi,r12
400988: movsd QWORD PTR [r15],xmm0
40098d: movsd QWORD PTR [r15+0x8],xmm0
400993: movsd QWORD PTR [r15+0x10],xmm0
400999: movsd QWORD PTR [rsp],xmm0

Source 14 40099e: call 400830
4009a3: test eax,eax
4009a5: movsd xmm0,QWORD PTR [rsp]

Target 14 400830: movsd xmm1,QWORD PTR [rip+0x0dafa8]
400837: 00
400838: movsd xmm3,QWORD PTR [rsi+0x10]

[...]

40087a: jb 400898
40087c: movsd xmm5,QWORD PTR [rdi+0x18]
400881: mov eax,0x2
400886: subsd xmm5,xmm1
40088a: ucomisd xmm5,xmm3

40088e: jb 400898

Source 15 400890: repz ret

Target 15 4009a3: test eax,eax
4009a5: movsd xmm0,QWORD PTR [rsp]
4009aa: je 400970
4009ac: cmp eax,0x1
```
• Intel “Westmere”: 2x 6 core X5650
• AMD “Magny Cours”: 4x 12 core 6164-HE
• Intel “Ivy Bridge”: 1x E3-1265L

• Perf in Linux 3.6.6, RHEL6
• All bells and whistles disabled
  – HW/SW frequency scaling, NMI watchdog, etc
• Latency biased
  
  ```cpp
  while (n--)
  ((n%2) ? x /= y : x += y);
  ```

• G4Box
  
  – Heavier version of the above
  – Executes two functions depending on condition

• Call chain
  
  – A loop, and a 10-deep call chain inside it

• Geant4 test40
  
  – Heavier version of the above
  – OO physics workload: small, fragmented methods
**Accuracy Metric**

Accuracy Error ($x$) = \[
\frac{\sum_{i \in BB} |(BB_x[i] - BB_{REF}[i])|}{\text{net_instruction_count}}
\]

- $x$ is the method
- Total # instructions measured by PIN
- # instructions measured in a BB
- # instructions measured by PIN

“wrongly attributed instructions”
Accuracy Error per method/benchmark, IVB (lower is better)

- Classic is always inferior
- Precise events are better
- LBR is best
## RAW RESULTS - KERNELS

<table>
<thead>
<tr>
<th>INST RETIRED default</th>
<th>Precise event or uops</th>
<th>Taken Branches Retired</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Even period</td>
<td>Prime period</td>
</tr>
<tr>
<td><strong>AMD</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>latencybias</td>
<td>0.84</td>
<td>0.88</td>
</tr>
<tr>
<td>callchain</td>
<td>0.61</td>
<td>0.80</td>
</tr>
<tr>
<td>testG4Box</td>
<td>0.28</td>
<td>0.29</td>
</tr>
<tr>
<td>test40</td>
<td>0.60</td>
<td>0.91</td>
</tr>
<tr>
<td><strong>WSM</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>latencybias</td>
<td>1.57</td>
<td>1.57</td>
</tr>
<tr>
<td>callchain</td>
<td>0.48</td>
<td>0.53</td>
</tr>
<tr>
<td>testG4Box</td>
<td>0.32</td>
<td>0.36</td>
</tr>
<tr>
<td>test40</td>
<td>0.58</td>
<td>0.57</td>
</tr>
<tr>
<td><strong>IVB</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>latencybias</td>
<td>1.21</td>
<td>0.28</td>
</tr>
<tr>
<td>callchain</td>
<td>0.59</td>
<td>0.48</td>
</tr>
<tr>
<td>testG4Box</td>
<td>0.18</td>
<td>0.17</td>
</tr>
<tr>
<td>test40</td>
<td>0.60</td>
<td>0.20</td>
</tr>
</tbody>
</table>
• Non-HPC subset from SPEC2006 CPU
  – 429.mcf,
  – 453.povray,
  – 471.omnetpp,
  – 483.xalancbmk

• CERN’s “FullCMS”
  – Monte-Carlo simulation
  – Several MLOC
  – Runs on ~300’000 cores
SAMPLE OF RAW RESULTS — APPS ON IVB

Accuracy Error per method/app, IVB
(lower is better)

- mcf
- povray
- omnetpp
- xalancbmk
- FullCMS

- Classic
- Precise
- LBR
## RAW RESULTS — MACHINE/APP

<table>
<thead>
<tr>
<th></th>
<th>INST RETIRED Default</th>
<th>Precise; Fixed period</th>
<th>Precise; Prime period</th>
<th>LBR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AMD</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mcf</td>
<td>0.66</td>
<td>1.225</td>
<td>1.212</td>
<td></td>
</tr>
<tr>
<td>povray</td>
<td>0.48</td>
<td>0.545</td>
<td>0.557</td>
<td></td>
</tr>
<tr>
<td>omnetpp</td>
<td>0.80</td>
<td>1.018</td>
<td>1.013</td>
<td></td>
</tr>
<tr>
<td>xalancbmk</td>
<td>0.53</td>
<td>0.606</td>
<td>0.606</td>
<td></td>
</tr>
<tr>
<td>FullCMS</td>
<td>0.53</td>
<td>0.611</td>
<td>0.606</td>
<td></td>
</tr>
<tr>
<td><strong>WSM</strong></td>
<td></td>
<td></td>
<td></td>
<td>N/A</td>
</tr>
<tr>
<td>mcf</td>
<td>0.44</td>
<td>0.432</td>
<td>0.446</td>
<td>0.194</td>
</tr>
<tr>
<td>povray</td>
<td>0.50</td>
<td>0.511</td>
<td>0.514</td>
<td>0.177</td>
</tr>
<tr>
<td>omnetpp</td>
<td>0.67</td>
<td>0.682</td>
<td>0.679</td>
<td>0.302</td>
</tr>
<tr>
<td>xalancbmk</td>
<td>0.46</td>
<td>0.485</td>
<td>0.484</td>
<td>0.087</td>
</tr>
<tr>
<td>FullCMS</td>
<td>0.55</td>
<td>0.547</td>
<td>0.547</td>
<td>0.177</td>
</tr>
<tr>
<td><strong>IVB</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mcf</td>
<td>0.50</td>
<td>0.306</td>
<td>0.305</td>
<td>0.034</td>
</tr>
<tr>
<td>povray</td>
<td>0.49</td>
<td>0.164</td>
<td>0.161</td>
<td>0.123</td>
</tr>
<tr>
<td>omnetpp</td>
<td>0.65</td>
<td>0.246</td>
<td>0.249</td>
<td>0.116</td>
</tr>
<tr>
<td>xalancbmk</td>
<td>0.56</td>
<td>0.341</td>
<td>0.340</td>
<td>0.112</td>
</tr>
<tr>
<td>FullCMS</td>
<td>0.55</td>
<td>0.179</td>
<td>0.177</td>
<td>0.120</td>
</tr>
</tbody>
</table>

Errors
Green is better
Lower is better
RECOMMENDATIONS

• Full list in paper
• Control over events and distinctions matter
• Use modern platforms
• LBR and precise sampling are good and can be improved in hardware
• LBR based methods offer enhanced degrees of accuracy
• “Ivy Bridge” (IVB) offers good features
• LBR and IVB “precisely distributed” (“PDIR”) work best
• Software improvements can help a lot
• “Known effects” are less pronounced on apps overall
• 5x improvement on apps, 20x on kernels

• Decent basic-block level accuracy is possible with EBS
CONCLUSIONS

- Use precise events, they’re “for free”
- Use LBR whenever available, large gains
- Decent basic-block level accuracy is possible with EBS

Thank you.
Andrzej.Nowak@cern.ch