High Frequency Performance Monitoring via Architectural Event Measurement

Chen Liu
cliu@clarkson.edu
Clarkson University, Potsdam, New York, USA
Motivation

- There is significant demand to improve various performance metrics such as speed and energy efficiency within modern computing systems.

- The finer-grained performance details are commonly gathered using hardware performance counters that are built into modern processors.

Applications of performance counter profiling:
- Optimization
- Malware Detection
- Scheduling techniques
- Dynamic power estimation
Motivation

Many tools have been developed to provide a high-level API to control the low-level performance counters.

<table>
<thead>
<tr>
<th>Tools</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perf</td>
<td>Timer granularity</td>
</tr>
<tr>
<td>PAPI</td>
<td>Performance overhead</td>
</tr>
<tr>
<td>LiMiT</td>
<td>Require Source code instrumentation</td>
</tr>
<tr>
<td></td>
<td>Require kernel patch</td>
</tr>
</tbody>
</table>
Kernel - Lineage of Event Behavior (K-LEB)

A performance counter-based profiling tool that utilizes a kernel space collection system to produce *precise, non-intrusive, low overhead, high periodicity* performance counter data.
K-LEB System Model

- **Controller process**
  - Control the kernel module from user space.

- **Kernel module**
  - Access PMU to collect performance counter data.

- **PMU**
  - Special hardware registers use to monitor the hardware events.

- **Monitored process**
  - Process being monitored.
Process Flows

5 phases
1) Module initialization
2) Start monitoring
3) Stop monitoring
4) Module de-initialization
5) Logging
Process Interaction

Interaction between K-LEB and the monitored process.

5 phases

1) Module initialization
2) Start monitoring
3) Stop monitoring
4) Module de-initialization
5) Logging
Experiment setup

Intel Core i7-920 @ 2.67GHz Nehalem running Ubuntu 16.04 with Linux kernel version 4.13.0-15.

Intel Xeon Platinum 8259CL @ 2.50GHz Cascade Lake running Ubuntu 16.04 with Linux kernel version 4.4.0-1112-aws.

- LINPACK
  - Capture phase behavior
  - Non-intrusive monitoring
  - Performance overhead

- Docker
  - Workload characterization
  - Non-intrusive monitoring

- Meltdown
  - Anomaly detection
  - High timer granularity
Case study 1 LINPACK

- Capture phase behavior.
- K-LEB has a very small FLOPS loss of 0.64% in comparison with 7.08% from Perf.
- No source code required.
Case study 2 Docker

- Workload characterization.
- Computation/Memory intensive.
- Non-intrusive to a running program.
Case study 2 Docker (continued)

- Running on AWS machine.
- The programs still follow the same trend in terms of their MPKI from low to high.
Case study 3 Meltdown

- Anomaly detection.
- High frequency timer.
- Monitor program with short execution time.

![Graph showing LLC Miss/Reference Ratio for Meltdown vs Non-Meltdown cases]

Sample Number

0.0 0.2 0.4 0.6 0.8 1.0 1.2

Non-Meltdown Meltdown

Meltdown vs Non-Meltdown LLC Miss/Reference Ratio

BID 2022 April 2nd, 2022
Performance overhead comparison

- Test on matrix multiplication program.

- Percentage performance overhead for each profiling tools.

<table>
<thead>
<tr>
<th>Sample Rate</th>
<th>K-LEB</th>
<th>Perf stat</th>
<th>Perf record</th>
<th>Number of Samples</th>
<th>PAPI</th>
<th>LiMiT</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 ms</td>
<td>0.68</td>
<td>6.01</td>
<td>1.66</td>
<td>250</td>
<td>6.43</td>
<td>4.08</td>
</tr>
<tr>
<td>1 ms</td>
<td>0.8</td>
<td>N/A</td>
<td>2.15</td>
<td>2500</td>
<td>7.78</td>
<td>4.47</td>
</tr>
<tr>
<td>0.1 ms</td>
<td>1.48</td>
<td>N/A</td>
<td>6.55</td>
<td>25000</td>
<td>16.53</td>
<td>10.01</td>
</tr>
</tbody>
</table>
Normalized Execution Time

- Test on matrix multiplication program.
- K-LEB consistently has less spread in execution time across all comparable tool.
Hardware events count difference

- Percentage difference of hardware events count of K-LEB in comparison to other profiling tools.

<table>
<thead>
<tr>
<th></th>
<th>Branch</th>
<th>Load</th>
<th>Store</th>
<th>Instruction retired</th>
<th>Clock cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perf stat</td>
<td>-7.95E-04</td>
<td>-6.29E-05</td>
<td>-3.90E-04</td>
<td>-5.23E-05</td>
<td>-0.30</td>
</tr>
<tr>
<td>Perf record</td>
<td>7.38E-03</td>
<td>-4.55E-03</td>
<td>-0.15</td>
<td>5.42E-03</td>
<td>0.03</td>
</tr>
<tr>
<td>PAPI</td>
<td>0.01</td>
<td>0.03</td>
<td>0.24</td>
<td>0.01</td>
<td>0.02</td>
</tr>
</tbody>
</table>
Conclusion

In this work we introduce K-LEB, a kernel module-based approach for performance counter collection with following features.

- Being non-intrusive to the program being monitored.
- Can provide high frequency sampling rate up to 100µs, which is 100 finer granularity than current available tools.
- Have very low overhead.

This new approach allows users to better measure performance and behavioral characteristics of programs.

As a result, many other subject areas that benefit from using performance data, such as program analysis, malware detection and scheduling techniques, could advance as well.
Acknowledgement

- Thanks Mr. Chutitep Woralert, Mr. James Bruska, and Dr. Lok Yan for working on the K-Leb project.

- https://github.com/camel-clarkson/k-leb

- https://camel.clarkson.edu/