Efficient Performance Monitoring in the Cloud with Virtual Performance Monitoring Units (PMUs)

Contributors
Wei Wang, Kan Liang, Like Xu, Andi Kleen, Guang Zeng, Danmei Wei

Presenter
Sean Christopherson
Agenda

Project Goals
Background
Our Solutions
Test Results
Current Status
Future Works
Part 1: Project Goals
Project Goals

Virtual PMUs are usually disabled in today’s clouds

```
root@instance-4: /opt/perf - Google Chrome

https://ssh.cloud.google.com/projects/ubuntu-12-09-2018/zones/us-west2:

root@instance-4:/opt/perf# perf record -e branch-misses ./ftest
Error:
The branch-misses event is not supported.
root@instance-4:/opt/perf#  
```
Project Goals

Virtual PMUs are usually disabled in today’s clouds

- inaccurate profiling results
- lack of advanced PMU features (e.g., LBR and PEBS)

Many cloud vendors (e.g., Google*, Alibaba*, Tencent*, Huawei*, Baidu*) have a strong interest in making PMUs usable in their cloud productions

What we did

- Reduced PMU virtualization overhead to generate more accurate profiling results
- Added support for LBR and PEBS virtualization in KVM

*Other names and brands may be claimed as the property of others.
Part 2: Background
Each **Fixed Function Counter** counts a specific event:
- Fixed counter 0: Instruction retired
- Fixed counter 1: Unhalted core cycles
- Fixed counter 2: Reference cycles

**General Purpose Counters** can be configured to count any supported event:
- Unhalted core cycles
- Instruction retired
- Branch instruction retired
- …

**Last Branch Records**
- Stack of MSRs that records branch sources and destinations
- Enabled via DEBUGCTRL MSR
- Usually takes a PMU counter to do branch sampling

PMU can be configured to generate **Performance Monitoring Interrupts** after N events:
- Sampling
- Histograms
- …
PMU Usage in Native Linux*

$ perf record -e branch-misses ./test_program

Perf Event:
- A scheduling entity from PMU’s point of view
- Stores the config and state data
- Usually associated with one or more PMU counters

*Other names and brands may be claimed as the property of others.
PMUs Usage in Linux* KVM Guest

$ perf record -e branch-misses ./test_program

Test Program

Approximately 2.6 ms

vPMU

Function calls

vPMI

Callback

PMI

MSR read/write

PMU

CPU

*Other names and brands may be claimed as the property of others.
Part 3: Our Solutions
vPMU Working Model Optimization

Guest Linux*

$: perf record -e branch-misses ./test_program

Linux Userspace Perf Utility

Linux Kernel Perf Subsystem

Perf Event

Test_program Thread

Callbars

Host Linux Perf Subsystem

vPMU

KVM vCPU

Function calls

PMU

CPU

vMSR read/write

vPMI

<700ns

MSR read/write

*Other names and brands may be claimed as the property of others.
Last Branch Records (LBR) Virtualization

$ perf record -b ./test_program

Linux Userspace Perf Utility

Linux Kernel Perf Subsystem

Test Event

Thread

Guest Linux*

vMSR read/write

vLBR Stack

vPMU

vLBR Enable

KVM vCPU

Host Linux Perf Subsystem

Perf Event

vCPU Thread

vCPU running

LBR Stack

PMU

CPU

LBR Enable

vLBR Enable

MSR read/write

vMSR read/write

*Other names and brands may be claimed as the property of others.
Part 4: Test Results
Test Environment

- CPU: Intel® Xeon® Processor E5-2699 v4 @ 2.20GHz
- Host and Guest Kernel: Linux* 4.19.0, booted with “nowatchdog”
- VM Configuration: 4 vCPUs, 8G memory

*Other names and brands may be claimed as the property of others.
Latency Comparison (Logarithmic)

Latency: ns

- MSR Update Latency: 2641807 ns (Non-optimization) vs 688 ns (Optimization) - 3500x reduction
- Guest PMI Latency: 8114988 ns (Non-optimization) vs 9073 ns (Optimization) - 900x reduction
$ perf record -e branch-misses ./test_program

### Test on Host:

- 48.19% ftest ftest [.] main
- 21.21% ftest ftest [.] bar
- 18.44% ftest ftest [.] foo
- 11.64% ftest ftest [.] qux
- 0.40% ftest libc-2.23.so [.] __random
- 0.02% ftest libc-2.23.so [.] __random_r

### Test in Guest without optimization:

- 82.64% ftest [kernel.kallsyms]
- 6.15% ftest [kernel.kallsyms]
- 4.62% ftest [kernel.kallsyms]
- 2.20% ftest [kernel.kallsyms]
- 1.32% ftest [kernel.kallsyms]
- 0.66% ftest [kernel.kallsyms]

### Test in Guest with optimization:

- 46.18% ftest ftest [.] main
- 22.29% ftest ftest [.] bar
- 20.29% ftest ftest [.] foo
- 10.47% ftest ftest [.] qux
- 0.36% ftest libc-2.23.so [.] __random
- 0.24% ftest libc-2.23.so [.] __random_r

Perf run doesn’t complete due to the large vPMU overhead

Results gathered via stopping the run via “ctrl-c”
## Last Branch Recording (LBR) Tests

$ perf record --call-graph lbr ./ftest

### Host Results

<table>
<thead>
<tr>
<th>Children</th>
<th>Self</th>
<th>Command</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 99.99%</td>
<td>0.00%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] _start</td>
</tr>
<tr>
<td>+ 99.99%</td>
<td>0.00%</td>
<td>ftest</td>
<td>libc-2.23.so</td>
<td>[.] __libc_start_main</td>
</tr>
<tr>
<td>+ 99.99%</td>
<td>13.27%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] main</td>
</tr>
<tr>
<td>+ 39.73%</td>
<td>36.65%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] qux</td>
</tr>
<tr>
<td>+ 38.72%</td>
<td>16.57%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] bar</td>
</tr>
<tr>
<td>+ 29.44%</td>
<td>10.73%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] foo</td>
</tr>
<tr>
<td>+ 20.71%</td>
<td>8.37%</td>
<td>ftest</td>
<td>libc-2.23.so</td>
<td>[.] __random</td>
</tr>
<tr>
<td>+ 13.39%</td>
<td>12.97%</td>
<td>ftest</td>
<td>libc-2.23.so</td>
<td>[.] __random_r</td>
</tr>
<tr>
<td>+ 9.23%</td>
<td>1.32%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] random@plt</td>
</tr>
</tbody>
</table>

### Guest Results

<table>
<thead>
<tr>
<th>Children</th>
<th>Self</th>
<th>Command</th>
<th>Shared Object</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 100.00%</td>
<td>0.00%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] _start</td>
</tr>
<tr>
<td>+ 99.99%</td>
<td>0.00%</td>
<td>ftest</td>
<td>libc-2.23.so</td>
<td>[.] __libc_start_main</td>
</tr>
<tr>
<td>+ 99.99%</td>
<td>13.23%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] main</td>
</tr>
<tr>
<td>+ 38.51%</td>
<td>16.57%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] foo</td>
</tr>
<tr>
<td>+ 37.32%</td>
<td>37.26%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] qux</td>
</tr>
<tr>
<td>+ 27.69%</td>
<td>12.28%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] bar</td>
</tr>
<tr>
<td>+ 19.09%</td>
<td>9.43%</td>
<td>ftest</td>
<td>libc-2.23.so</td>
<td>[.] random</td>
</tr>
<tr>
<td>+ 10.88%</td>
<td>1.44%</td>
<td>ftest</td>
<td>ftest</td>
<td>[.] random@plt</td>
</tr>
<tr>
<td>+ 9.66%</td>
<td>9.65%</td>
<td>ftest</td>
<td>libc-2.23.so</td>
<td>[.] random_r</td>
</tr>
</tbody>
</table>
Part 5: Current Status
Current Status

- vPMU optimization
  - https://lkml.org/lkml/2018/11/1/937 (full optimization, NAK’d)
  - https://lkml.org/lkml/2019/10/27/834 (intermediate step)
- LBR
- PEBS
  - https://lkml.org/lkml/2019/10/27/53
Part 6: Future Works
Future Works

- Continue to upstream the patches
- Support arch v5 PMU features
Thank You!
Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request. No product or component can be absolutely secure.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm

Intel, the Intel logo, and Xeon are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

© Intel Corporation.