Dancing in the Dark: Profiling for Tiered Memory

Jinyoung Choi, Sergey Blagodurov* and Hung-Wei Tseng
University of California, Riverside and Advanced Micro Devices (AMD)*
Expected to see in this talk

- Challenges in tiered memory architectures
- Comparison between existing HW monitors for profiling memory-access
- A hybrid profiling method combining multiple HW monitors together
Memory heterogeneity

With emergence of new memory technologies, memory-access became heterogeneous.
Tiered Memory Architecture (TMA)

Byte-addressable memory

Low latency/high bandwidth

- HBM
- DRAM

High latency/low bandwidth

- PMEM

Low capacity

- Cold data
- Hot data

High capacity

load/store instruction

Process

CPU

Profiling memory behavior is critical part in TMA
Challenge #1 — Poor visibility in SW side

SW

No SW interface

Page fault exceptions

Compiler

Machine instruction

load/store instruction

HW monitors

L1/L2/LLC/TLB misses

Memory-access
- Cache/TLB lookup
- Page table walk
- DRAM access
**Challenge #2 — Diversity of HW monitors**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTE Accessed bit</td>
</tr>
<tr>
<td>Performance counters</td>
</tr>
<tr>
<td>IBS (Instruction Based Sampling)</td>
</tr>
<tr>
<td>PEBS (Precise Event Based Sampling)</td>
</tr>
</tbody>
</table>

Q. What are pros/cons for each method?
Challenge #3 — Profiling metrics?

Q. How to combine different information from different methods?

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTE Accessed bit</td>
</tr>
<tr>
<td>Performance counters</td>
</tr>
<tr>
<td>IBS (Instruction Based Sampling)</td>
</tr>
<tr>
<td>PEBS (Precise Event Based Sampling)</td>
</tr>
</tbody>
</table>
Following sections

• Background
• Tiered-Memory Profiler (TMP)
• Evaluation
• Conclusion
## Existing HW monitors

<table>
<thead>
<tr>
<th>Method</th>
<th>Supported architecture</th>
<th>Interesting information</th>
<th>Granularity</th>
<th>Tracking target address?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTE Accessed bit</td>
<td>Universal (Page table)</td>
<td>Touched by MMU (TLB misses)</td>
<td>Coarse (polling by SW)</td>
<td>Yes</td>
</tr>
<tr>
<td>Performance counters</td>
<td>Vendor specific</td>
<td>HW events (cache/TLB misses)</td>
<td>Fine (register counter)</td>
<td>N/A</td>
</tr>
<tr>
<td>IBS (Instruction Based Sampling)</td>
<td>AMD</td>
<td>Cache/TLB misses</td>
<td>Fine (trace-based)</td>
<td>Yes</td>
</tr>
<tr>
<td>PEBS (Precise Event Based Sampling)</td>
<td>Intel</td>
<td>HW events (e.g., cache misses)</td>
<td>Fine (trace-based)</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Tiered-Memory Profiler (TMP)

- TMP kernel module combines A-bit, IBS/PEBS, and performance counter drivers.
- API provides comprehensible memory-access information to the application.
Tiered-Memory Profiler (TMP)

1. Get PIDs from top command
2. Filter PIDs based on usage of CPU and memory
3. Request A-bit checking for selected PIDs to A-bit driver
4. Traverse all valid PTEs
5. In each PTE, check A-bit and reset for the next epoch
6. If a page is accessed, increment A-bit count in page descriptor

Diagram:
- User-Space:
  - Process id, sampling rate, and etc...
  - Top
  - TMP Daemon
- Kernel-Space:
  - A-bit driver
  - IBS/PEBS driver
  - Performance Counter module
  - Page Descriptor
  - Performance Counters
  - IBS/PEBS

Operations:
- Check A-bit
- A-bit count
- IBS/PEBS count
- Sample
- Read/Reset

Main Memory
- PTE
- Page Descriptor

Processor
Tiered-Memory Profiler (TMP)

API

User-Space

TMP Daemon

Process id, sampling rate, and etc...

Kernel-Space

TMP Kernel Module

A-bit driver

IBS/PEBS driver

Performance Counter module

1. Enable IBS/PEBS
2. Receive periodic interrupt based on sampling rate
3. Copy samples in interrupt handler
4. Increment IBS/PEBS count in page descriptor
5. Disable IBS/PEBS

Check A-bit

PTE

Page Descriptor

A-bit count

IBS/PEBS count

Sample

Read/Reset

Performance Counters

Processor

Main Memory

A-bit

Enable IBS/PEBS

Receive periodic interrupt based on sampling rate

Copy samples in interrupt handler

Increment IBS/PEBS count in page descriptor

Disable IBS/PEBS

UC RIVERSIDE
Tiered-Memory Profiler (TMP)

User-Space

API

Process id, sampling rate, and etc...

TMP Daemon

2

1
Give TMP Daemon register information

Kernel-Space

TMP Kernel Module

A-bit driver

IBS/PEBS driver

Performance Counter module

Check A-bit

A-bit count

IBS/PEBS count

Sample

Read/Reset

Main Memory

Processor

Page Descriptor

IBS/PEBS

Performance Counters

1
Read performance counter register and reset

PTE

PTE

PTE

PTE

Process id, sampling rate, and etc…

1
# Testbed and workload setup

## Testbed

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>AMD Ryzen5 3600X w/ 6 cores</td>
</tr>
<tr>
<td>LLC</td>
<td>32MB</td>
</tr>
<tr>
<td>Memory</td>
<td>64 GB DRAM</td>
</tr>
<tr>
<td>Storage</td>
<td>1 TB SSD</td>
</tr>
<tr>
<td>OS</td>
<td>Linux kernel version 4.15.18</td>
</tr>
</tbody>
</table>

## Workload

<table>
<thead>
<tr>
<th>Workload</th>
<th>Input</th>
<th>Type</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Analytics</td>
<td>Wiki dataset / Size: 0.6GB</td>
<td>CloudSuite</td>
<td>1 master / 32 workers</td>
</tr>
<tr>
<td>Data Caching</td>
<td>Twitter dataset / Size: 36GB</td>
<td>CloudSuite</td>
<td>4 memcached / 8 clients</td>
</tr>
<tr>
<td>Graph 500</td>
<td>Input size: 1GB</td>
<td>HPC</td>
<td>8 processes</td>
</tr>
<tr>
<td>Graph Analytics</td>
<td>Twitter dataset / Size: 1.4GB</td>
<td>CloudSuite</td>
<td>1 master / 16 workers</td>
</tr>
<tr>
<td>GUPS</td>
<td>Input size: 4GB</td>
<td>HPC</td>
<td>8 processes</td>
</tr>
<tr>
<td>LULESH</td>
<td>Input size: 21GB</td>
<td>HPC</td>
<td>8 processes</td>
</tr>
<tr>
<td>Web Serving</td>
<td>Faban workload generator</td>
<td>CloudSuite</td>
<td>3 servers / 100 clients</td>
</tr>
<tr>
<td>XSBench</td>
<td>Input size: 120GB</td>
<td>HPC</td>
<td>8 processes</td>
</tr>
</tbody>
</table>
Heatmaps: Data Analytics

IBS profiling

A-bit profiling

Two heatmaps are not identical!

Hot pages

Hot pages

Hot pages
## Count of pages captured with A-bit and IBS

<table>
<thead>
<tr>
<th>Workload</th>
<th>A-bit</th>
<th>IBS</th>
<th>Both (from each method)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Analytics</td>
<td>111175</td>
<td>58740</td>
<td>1425</td>
</tr>
<tr>
<td>Data Caching</td>
<td>14817</td>
<td>11835</td>
<td>30</td>
</tr>
<tr>
<td>Graph 500</td>
<td>5458</td>
<td>4896</td>
<td>16</td>
</tr>
<tr>
<td>Graph Analytics</td>
<td>28323</td>
<td>28260</td>
<td>104</td>
</tr>
<tr>
<td>GUPS</td>
<td>5587</td>
<td>76009</td>
<td>16</td>
</tr>
<tr>
<td>LULESH</td>
<td>5735</td>
<td>6940</td>
<td>3</td>
</tr>
<tr>
<td>Web Serving</td>
<td>25220</td>
<td>3002</td>
<td>71</td>
</tr>
<tr>
<td>XSBench</td>
<td>5301</td>
<td>199609</td>
<td>8</td>
</tr>
</tbody>
</table>
We evaluate two different page placement policies.

- **Oracle**: Assumes knowledge of how many times each page will be accessed in the coming epoch and brings in the hottest (most frequently accessed) pages into the tier 1 memory at the start of the epoch. Oracle represents the upper limit for policy design.

- **History**: A simple yet practical policy that, at the start of each epoch, brings the previous epoch’s hottest pages into tier 1 memory.
Hitrate in first tier of memory

In oracle, combined profiling is always outperforming or matching others.

In history, combined profiling is not always outperforming because history policy cannot predict complicated memory access.
Hitrate in first tier of memory

In oracle, combined profiling is always outperforming or matching others.

In history, combined profiling is not always outperforming because history policy cannot predict complicated memory access.

---

In oracle, combined profiling is always outperforming or matching others.

In history, combined profiling is not always outperforming because history policy cannot predict complicated memory access.
Conclusion

- Tiered memory architectures require careful resource allocation to maximize their efficiencies — none of existing profiling mechanisms can provide a **complete picture of the application behavior**.
- Tiered-Memory Profiler (TMP) provides **comprehensible memory-access information** by combining multiple low-overhead profiling mechanisms together.
- Leveraging TMP the memory-access information to achieve up to 70% better performance.
Thank you

CNS-1940048
CNS-2007124

https://www.escalab.org/
https://www.github.com/ESCALAB
References

This file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license.
Disclaimer:
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.’ AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Attribution:
© 2021 Advanced Micro Devices, Inc. All rights reserved.
AMD, the AMD Arrow logo, Ryzen™ and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.
Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.