Investigating Hardware Caches for Terabyte-scale NVDIMMs

Julian T. Angeles  
CS Dept., UC Davis  
jtangeles@ucdavis.edu

Mark Hildebrand  
ECE Dept., UC Davis  
mhildebrand@ucdavis.edu

Venkatesh Akella  
ECE Dept., UC Davis  
akella@ucdavis.edu

Jason Lowe-Power  
CS Dept., UC Davis  
jlowepower@ucdavis.edu
Executive Summary

Problem
How do Terabyte-scale applications perform with hardware managed DRAM caches?

Try to understand these caches
- Microbenchmarks on real hardware

Show that insights manifest in graph workloads
- We find a 50% reduction in bandwidth utilization
- We find 3-5x more data being moved
- Terabyte-scale applications perform poorly with hardware managed DRAM caches.
Outline

- Background
  - Motivation
  - Related Work
- DRAM cache analysis
- Performance in real workload
- Discussion
Emerging Applications Are Growing

Source: https://www.product-solving.com/p/openais-gpt-3-will-change-how-we...
NVRAM Based Servers

- Cheaper per byte than DRAM
- Orders of magnitude more capacity
- Persistent and byte-addressable
- Increased latency and reduced bandwidth
DRAM as Cache

Memory Mode (2LM)

Application

Cache

Smaller DRAM used to mask higher latency

Main Memory

DRAM

NVRAM
Work Understanding NVRAM

- Izraelevitz et al. study Intel Optane DC characteristics\(^1\)
- Wang et al. analyze the microarchitecture with simulator\(^2\)

---

\(^1\) Joseph Izraelevitz, Jian Yang, Lu Zhang, Juno Kim, Xiao Liu, Amirsaman Memaripour, Yun Joon Soh, Zixuan Wang, Yi Xu, Subramanya R. Dulloor, Jishen Zhao, Steven Swanson - Basic Performance Measurements of the Intel Optane DC Persistent Memory Module (MICRO '19)

\(^2\) Zixuan Wang, Xiao Liu, Jian Yang, Theodore Michailidis, Steven Swanson, Jishen Zhao - Characterizing and Modeling Non-Volatile Memory Systems (MICRO '20)
Work Done to Bridge Performance Gap

Software Level

- AutoTM optimizes movement of live ML tensors¹
- Sage optimizes performance for graphs²

Motivation For Our Work

DRAM Cache Level

- Alloy, Loh-Hill explore designs and tradeoffs\(^1,2\)
- BEAR reduces bandwidth consumption of DRAM cache\(^3\)

Limitations and performance of DRAM caches in NVRAM systems?

---

[1] Gabriel H. Loh; Mark D. Hill - Efficiently enabling conventional block sizes for very large die-stacked DRAM caches (MICRO '11)
Outline

● Background
● DRAM cache analysis
  ○ General characteristics
  ○ Microbenchmarks
● Performance in real workload
● Discussion
DRAM Cache Characteristics

Intel System
- 384GB DRAM
- 6TB NVRAM

DRAM Cache
- Direct Mapped policy
- Experiences Access Amplification
Access Amplification: Write Dirty Miss

3 accesses to DRAM and NVRAM per demand request
DRAM Cache Evaluation

Goal
Understand access amplification and bandwidth performance

How

- Custom open source benchmark generator\(^1\)
- Hardware performance counters to capture traffic

# DRAM Cache: Higher Access Amplification

## Microbenchmark Memory Accesses

<table>
<thead>
<tr>
<th></th>
<th>LLC Read</th>
<th></th>
<th>LLC Write</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hit</td>
<td>Miss</td>
<td>Hit</td>
<td>Miss</td>
<td>DDO</td>
</tr>
<tr>
<td></td>
<td>Clean</td>
<td>Dirty</td>
<td>Clean</td>
<td>Dirty</td>
<td></td>
</tr>
<tr>
<td>DRAM Read</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>DRAM Write</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>NVRAM Read</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NVRAM Write</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Amplification</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

5x observed amplification compared to 3x
DRAM Cache: Higher Access Amplification

[Diagram showing mean bandwidth (GB/s) across different random and sequential accesses, highlighting 100% write dirty miss to DRAM and 1x DRAM Read, DRAM Write, and NVRAM Read, 2x DRAM Write.]
**DRAM Cache: Poor Bandwidth Utilization**

<table>
<thead>
<tr>
<th></th>
<th>Achievable</th>
<th>Observed with \nClean Misses</th>
<th>Observed with \nDirty Misses</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NVRAM Read Bandwidth</strong></td>
<td>30 GB/s$^1$</td>
<td>23 GB/s</td>
<td>23 GB/s</td>
</tr>
<tr>
<td><strong>NVRAM Write Bandwidth</strong></td>
<td>11 GB/s</td>
<td>8 GB/s</td>
<td>8 GB/s</td>
</tr>
</tbody>
</table>

Microbenchmark Bandwidth Performance

- **Achievable Bandwidth**: 30 GB/s
- **Observed with Clean Misses**: 23 GB/s
- **Observed with Dirty Misses**: 11 GB/s

**60% Utilization**

- **High DRAM cache miss rate experience severe bandwidth bottleneck**

**72% Utilization**

---

$^1$ Different from reported values due to larger DIMM size
Microbenchmark Insights

- **5x** Access Amplification per demand access
- High Miss-rate, **poor bandwidth utilization**
  - Does not account for latency

**Conclusion**
DRAM caches work poorly with working sets that exceed it
Outline

- Background
- DRAM cache analysis
- Performance in real workload
  - Miss-rate and bandwidth
  - Access Amplification
- Discussion
Case Study - Graphs

Why

- Growing workload
- Irregular Access Pattern

Galois (Gill et al.)

- Gill et al. study performance of Galois framework in 2LM

Graphs are getting huge!


Galois - Graph Inputs

Web Data Commons - Hyperlink2012 (wdc12)
- Largest publicly available graph
- 3.5 billion web pages and 128 billion hyperlinks

Randomized Scale Free Graph (kron30)
- Graph500 based kronecker generator
- Fits in DRAM

Data collection
- HW performance counters

[1]: Topology of the 2012 WDC Hyperlink Graph
Large Graphs Suffer Poor Bandwidth Utilization

Bandwidth performance of kron30, which fits in DRAM cache

Bandwidth performance of wdc12, which exceeds DRAM cache
Large Graphs Suffer Poor Bandwidth Utilization

Bandwidth performance of kron30, which fits in DRAM cache

Bandwidth performance of wdc12, which exceeds DRAM cache
Large Graphs Suffer Poor Bandwidth Utilization

Bandwidth performance of kron30, which fits in DRAM cache

~50% Reduction

Bandwidth performance of wdc12, which exceeds DRAM cache
Large Graphs: Miss-rate and Bandwidth

wdc12 Tag Statistics of PageRank

wdc12 Bandwidth Trace of PageRank
Large Graphs: Miss-rate and Bandwidth

**wdc12 Tag Statistics of PageRank**

**wdc12 Bandwidth Trace of PageRank**
Large Graphs: Miss-rate and Bandwidth

Modest miss-rate in graphs but severe bandwidth drop

wdc12 Tag Statistics of PageRank

wdc12 Bandwidth Trace of PageRank
How Much Data is Moved?

App Direct Mode (1LM)

Application

DRAM

NVRAM

Main Memory

NUMA Node

Observe baseline data movement of kernels
Large Graphs: Significant Data Movement

**wdc12** - NVRAM as extra NUMA nodes

**wdc12** - NVRAM as system memory, DRAM as cache
Large Graphs: Significant Data Movement

**wdc12** - NVRAM as extra NUMA nodes

**wdc12** - NVRAM as system memory, DRAM as cache
Large Graphs: Significant Data Movement

Excessive movement of data due to access amplification

wdc12 - NVRAM as extra NUMA nodes

wdc12 - NVRAM as system memory, DRAM as cache
Takeaways

- Modest amount of dirty misses causes poor bandwidth utilization
- Current implementation has high access amplification
- Terabyte-scale applications perform poorly with hardware managed DRAM caches.
Outline

- Background
- DRAM cache analysis
- Performance in real workload

Discussion
- Software Solutions in 1LM
- Future work
## Software Managed Memory

<table>
<thead>
<tr>
<th><strong>AutoTM</strong> (Our Group)(^1)</th>
<th><strong>Sage</strong> (Dhulipala et al.)(^2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Method</strong></td>
<td><strong>Method</strong></td>
</tr>
<tr>
<td>Used static information to optimize tensor movement</td>
<td>Graph algorithms that mutate in DRAM only</td>
</tr>
<tr>
<td><strong>Results</strong></td>
<td><strong>Results</strong></td>
</tr>
<tr>
<td>3x performance improvement over hardware caches</td>
<td>1.94x speedup over Galois(2LM)</td>
</tr>
</tbody>
</table>

---

\(^{1}\) Mark Hildebrand, Jawad Khan, Sanjeev Trika, Jason Lowe-Power, Venkatesh Akella - *AutoTM: Automatic Tensor Movement in Heterogeneous Memory Systems using Integer Linear Programming* (ASPLOS'20)

\(^{2}\) Laxman Dhulipala, Charles McGuffey, Hongbo Kang, Yan Gu, Guy E. Blelloch, Phillip B. Gibbons, Julian Shun - *Sage: parallel semi-asymmetric graph algorithms for NVRAMs* (VLDB'20)
Limitations of Software Data Management

CPU cores
- Using to move data
- Difficult to transfer data asynchronously

Specialization
- Requires application specific knowledge
- Not general enough for other use cases
Future Work

What We Want

- High level knowledge of data access patterns of software
- Hardware acceleration benefits

How do we design a solution that is both implicit (like hardware caches) and high performance (like explicit data movement)?