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

UC DAVIS
UNIVERSITY OF CALIFORNIA

Executive Summary

Problem

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

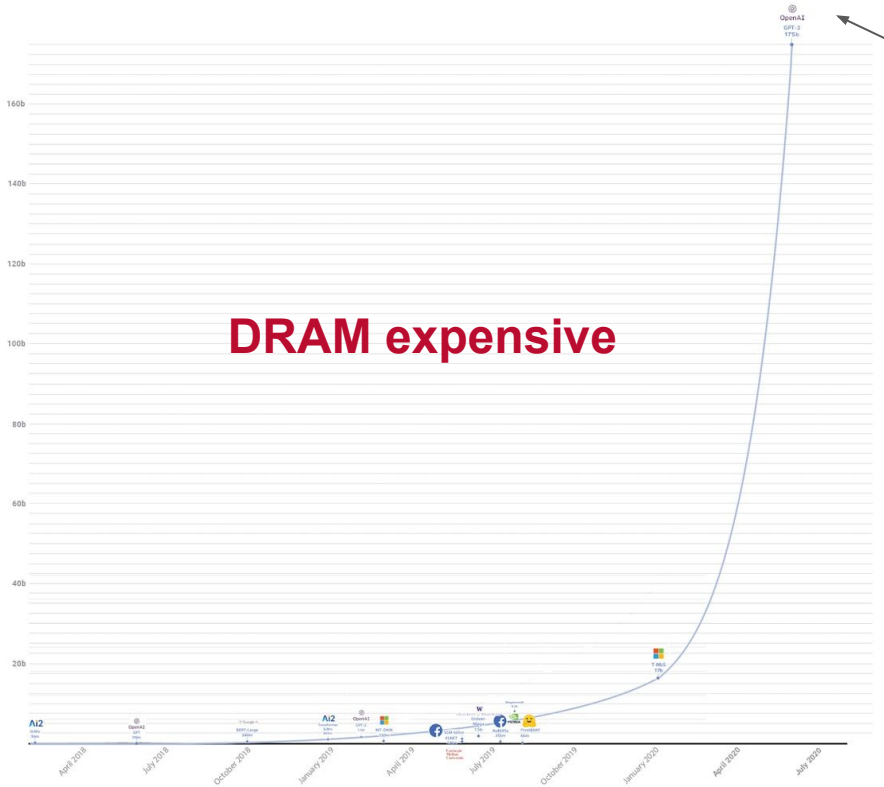
Try to understand these caches

- 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.

Emerging Applications Are Growing

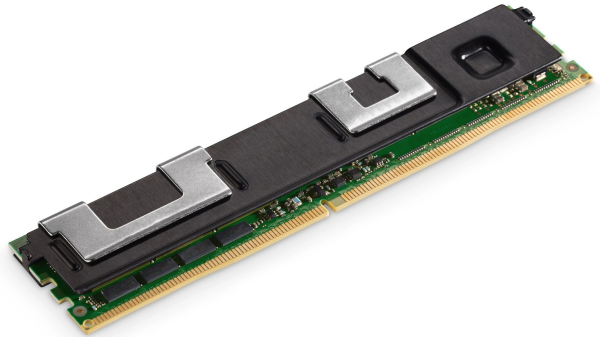


175 Billion Parameters!!

Source: *https://www.productsolving.com/p/openais-gpt-3-will-change-how-we*

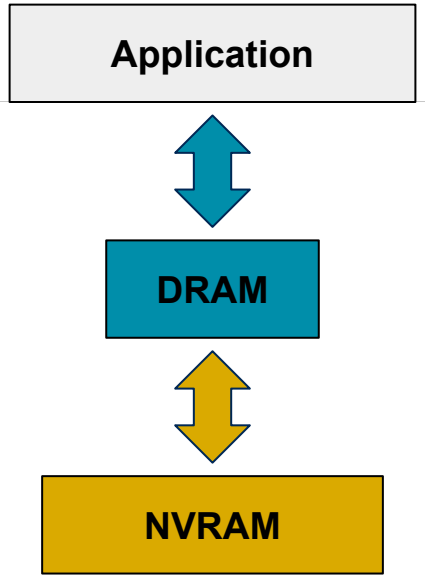
NVRAM Based Servers

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



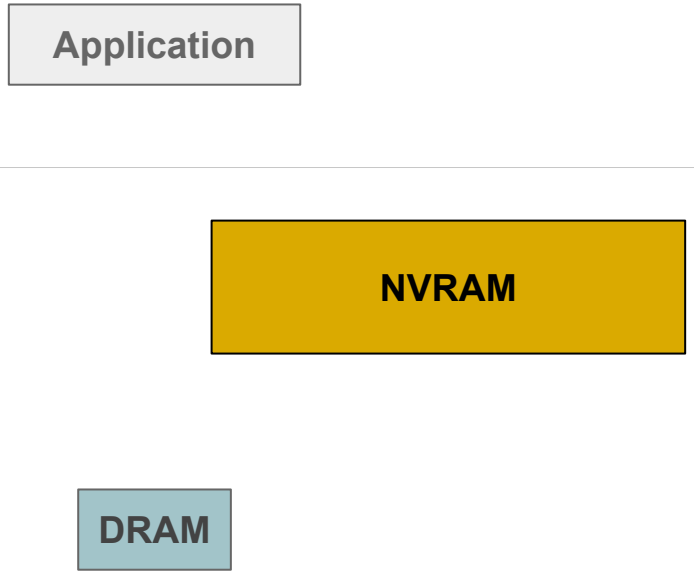
DRAM as Cache

Memory Mode (2LM)



Smaller DRAM used to **mask higher latency**

Work Understanding NVRAM



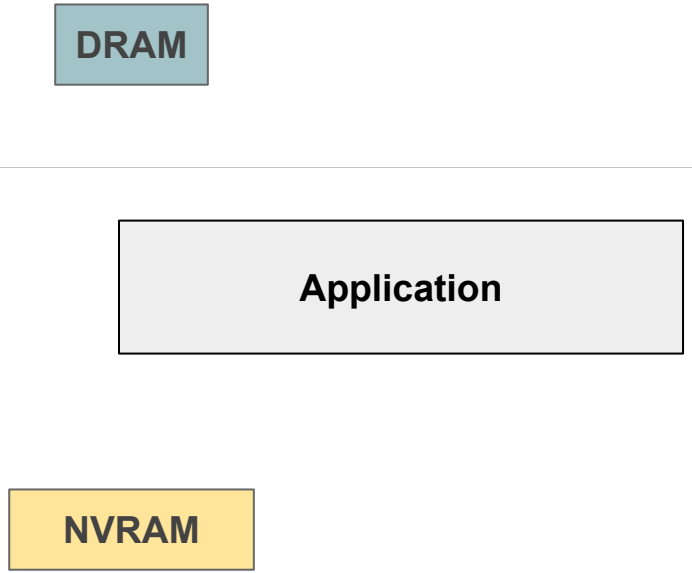
System Level

- 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)

UCDAVIS

Work Done to Bridge Performance Gap



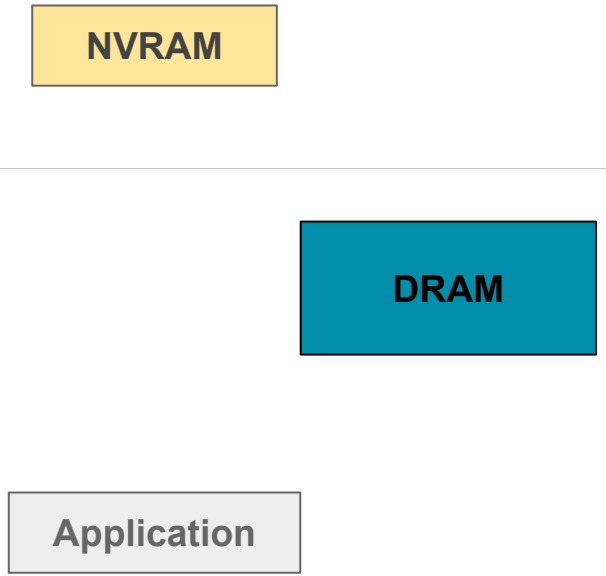
Software Level

- AutoTM optimizes movement of live ML tensors[1]
- Sage optimizes performance for graphs[2]

[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)

UCDAVIS

Motivation For Our Work



DRAM Cache Level

- Alloy, Loh-Hill explore designs and tradeoffs[1,2]
- BEAR reduces bandwidth consumption of DRAM cache[3]

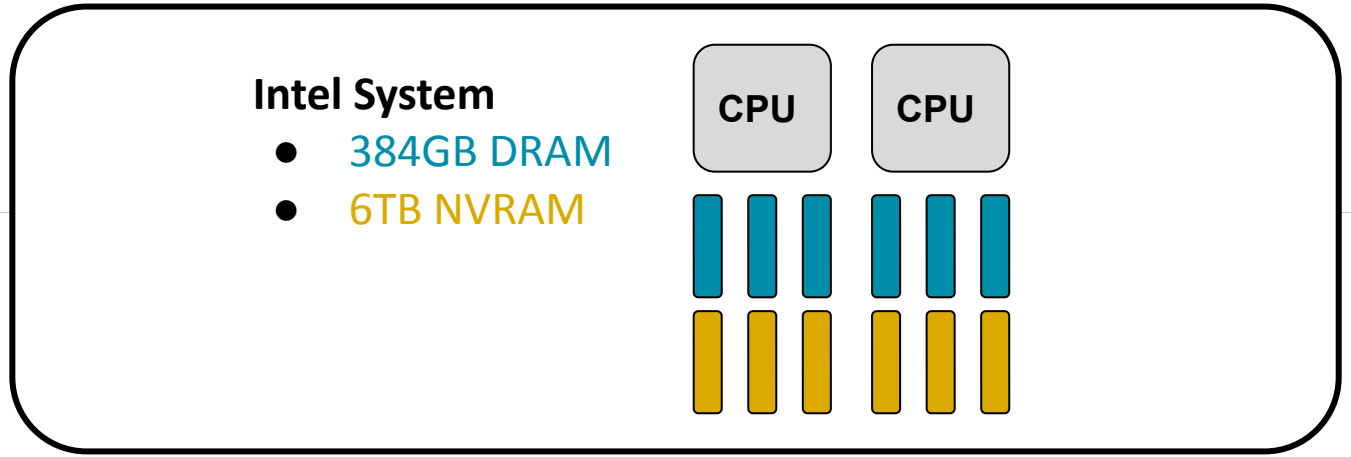
Limitations and **performance** of DRAM caches in NVRAM systems?

[1] Gabriel H. Loh; Mark D. Hill - *Efficiently enabling conventional block sizes for very large die-stacked DRAM caches* (MICRO '11)
[2] Moinuddin Qureshi, Gabriel H. Loh - *Fundamental Latency Trade-off in Architecting DRAM Caches: Outperforming Impractical SRAM-Tags with a Simple and Practical Design* (MICRO '12)
[3] Chiachen Chou, Aamer Jaleel, Moinuddin K. Qureshi - *BEAR: Techniques for mitigating bandwidth bloat in gigascale DRAM caches* (ISCA '15)

Outline

- Background
- DRAM cache analysis
  - General characteristics
  - Microbenchmarks
- Performance in real workload
- Discussion

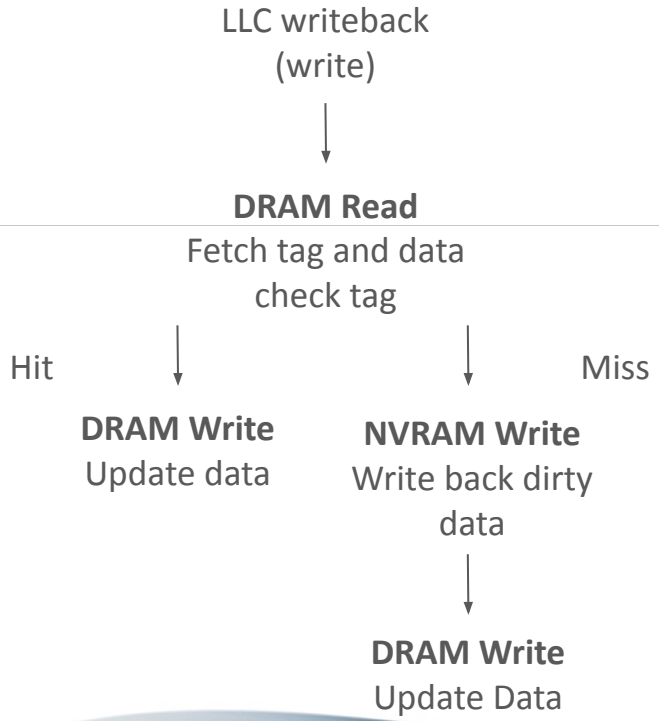
DRAM Cache Characteristics



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

[1] https://github.com/darchr/KernelBenchmarks.jl

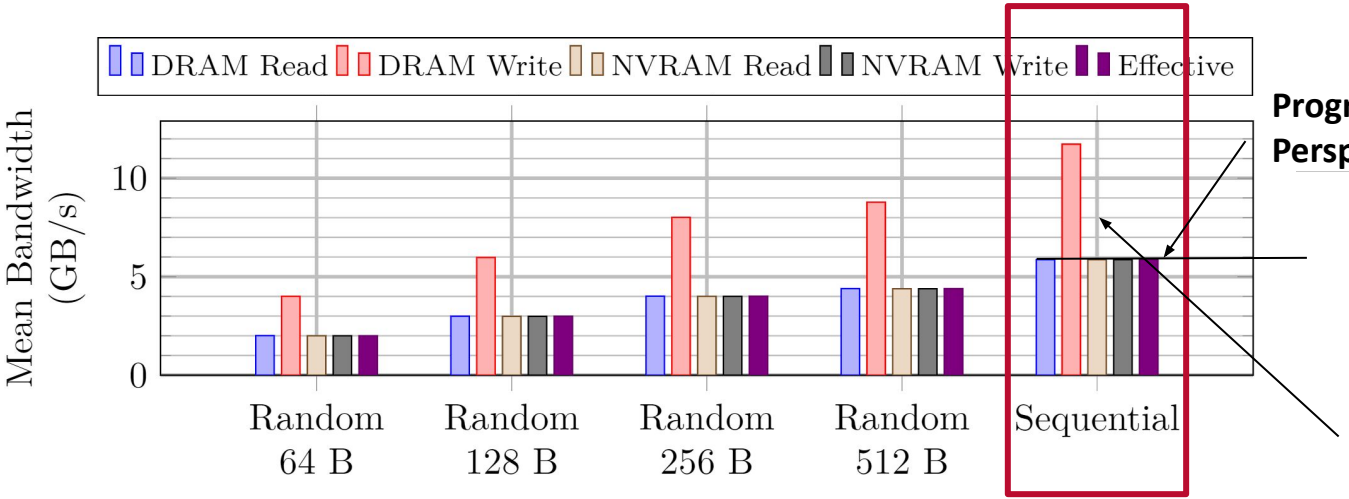
DRAM Cache: Higher Access Amplification

Microbenchmark Memory Accesses

	LLC Read			LLC Write			
	Hit	Miss		Hit	Miss		DDO
		Clean	Dirty		Clean	Dirty	
DRAM Read	1	1	1	1	1	1	
DRAM Write		1	1	1	2	2	1
NVRAM Read		1	1		1	1	
NVRAM Write			1			1	
Amplification	**1**	**3**	**4**	**2**	**4**	**5**	**1**

5x observed amplification
compared to **3x**

DRAM Cache: Higher Access Amplification



100% Write Dirty Miss to DRAM

Program Perspective

1x DRAM Read, DRAM Write and NVRAM Read

2x DRAM Write

DRAM Cache: Poor Bandwidth Utilization

	Achievable	Observed with Clean Misses
NVRAM Read Bandwidth	30 GB/s[1]	23 GB/s

60% Utilization

	Achievable	Observed with Dirty Misses
NVRAM Write Bandwidth	11 GB/s	8 GB/s

72% Utilization

Microbenchmark Bandwidth Performance

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

[1] *Different from reported values due to larger DIMM size*

UCDAVIS

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

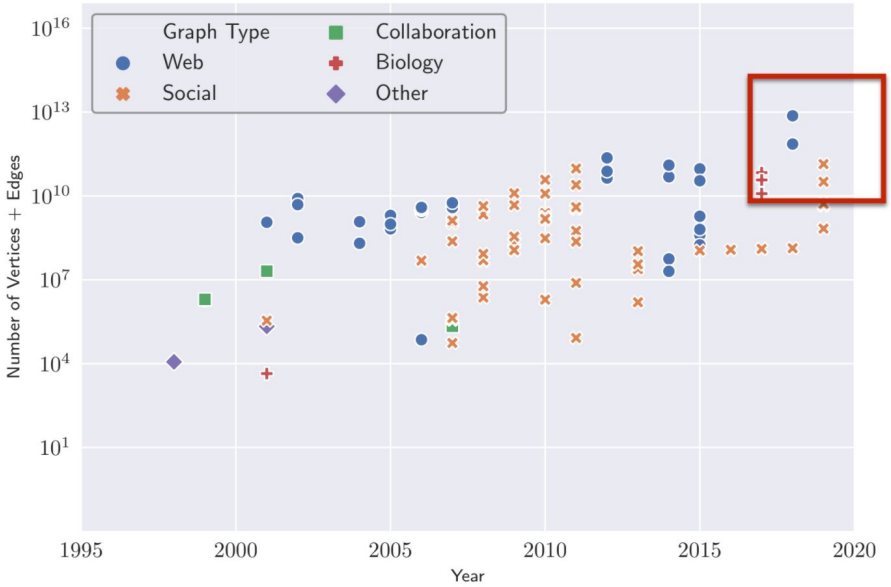
Case Study - Graphs

Why

- Growing workload
- Irregular Access Pattern

Galois (Gill et al.)

- Gill et al. study performance of Galois framework in 2LM[1]



Graphs are getting huge![2]

[1] Gurbinder Gill, Roshan Dathathri, Loc Hoang, Ramesh Peri, Keshav Pingali - *Single machine graph analytics on massive datasets using Intel optane DC persistent memory* (VLDB '20)

[2] Dhulipala, Laxman. *Sage: Parallel Semi-Asymmetric Algorithms for NVRAMs*. (VLDB '20)

UCDAVIS

Galois - Graph Inputs

Web Data Commons - Hyperlink2012 (*wdc12*)

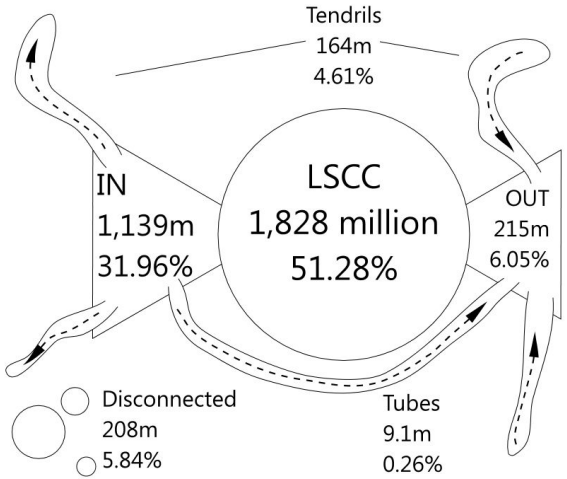
- Largest publicly available graph
- **3.5 billion web pages and 128 billion hyperlinks**

Randomized Scale Free Graph (*kron30*)

- Graph500 based kronecker generator
- Fits in DRAM

Data collection

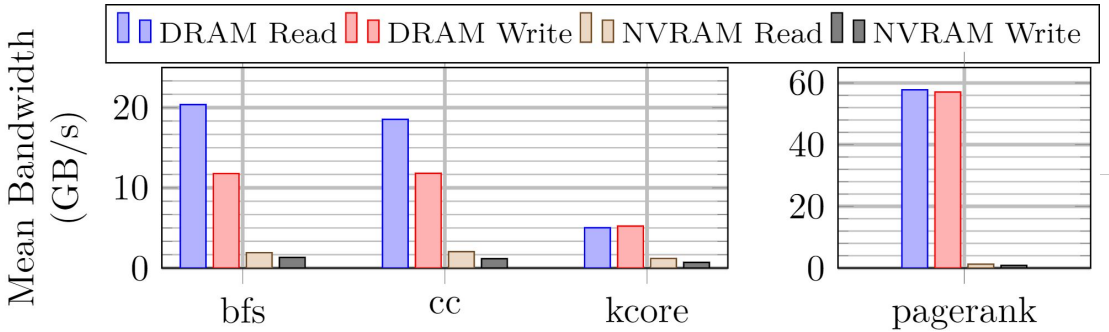
- HW performance counters



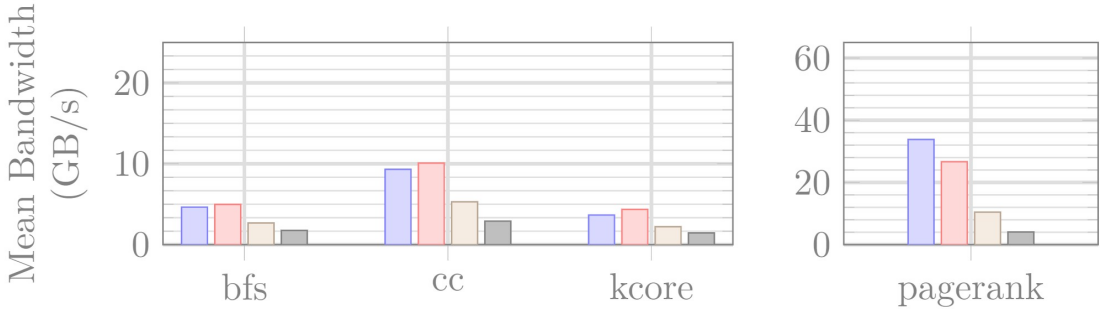
wdc12 characteristics[1]

[1]: Topology of the 2012 WDC Hyperlink Graph

Large Graphs Suffer Poor Bandwidth Utilization



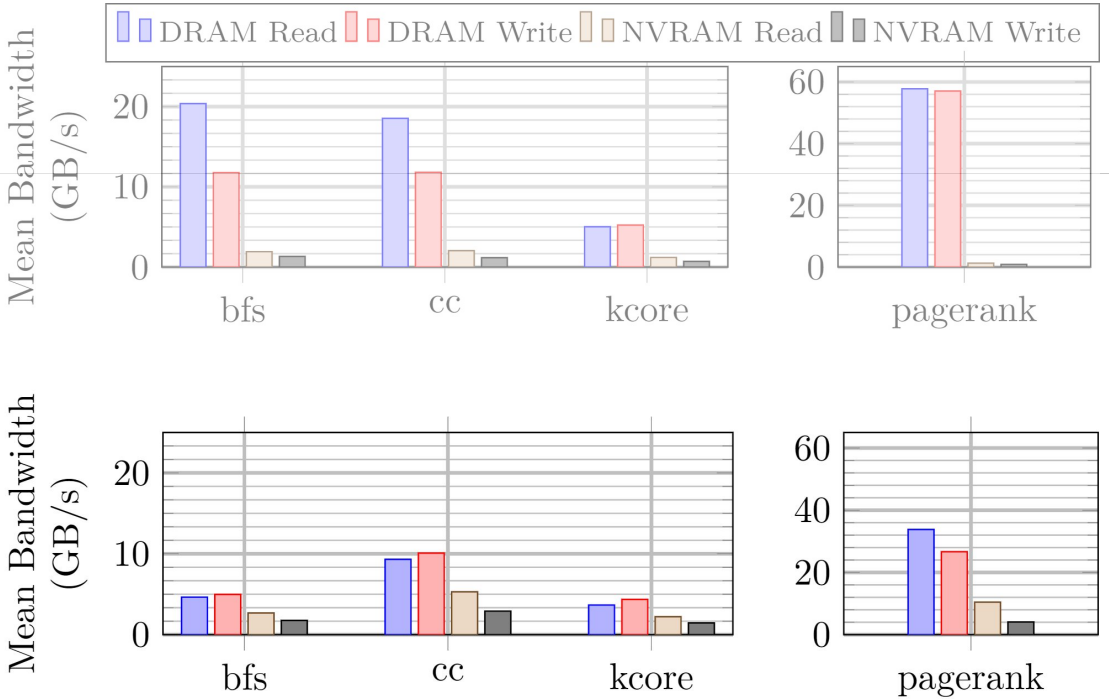
Bandwidth performance of kron30, which fits in DRAM cache



Bandwidth performance of wdc12, which exceeds DRAM cache

UCDAVIS

Large Graphs Suffer Poor Bandwidth Utilization

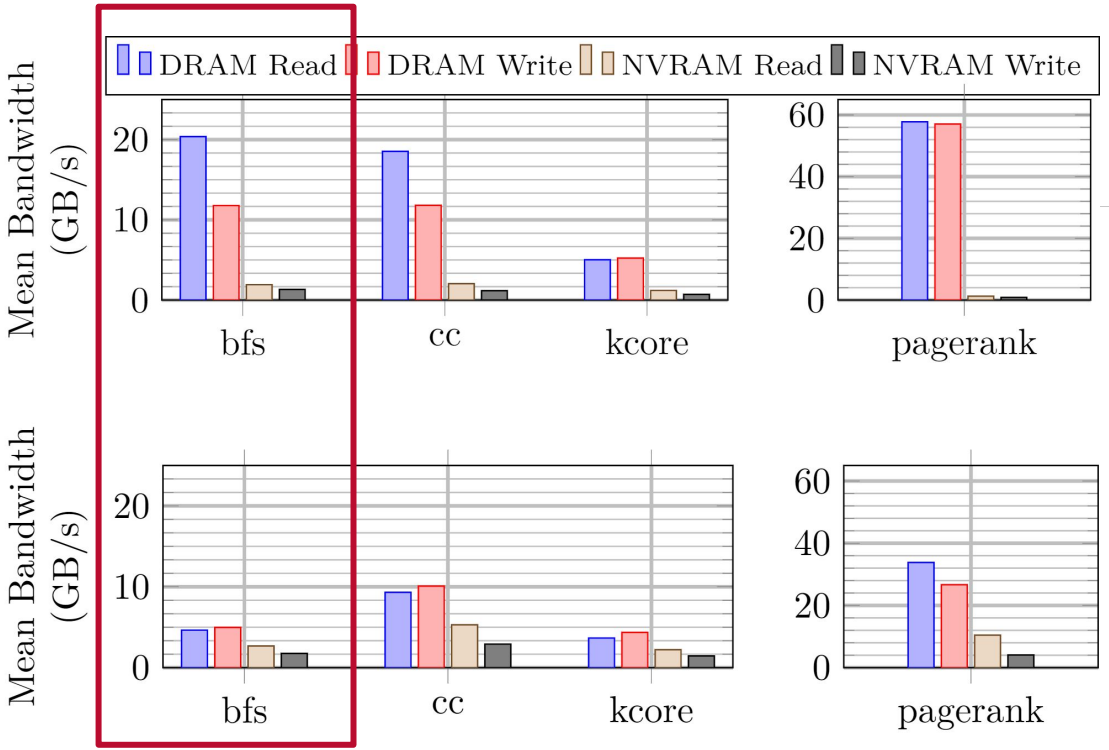


Bandwidth performance of kron30, which fits in DRAM cache

Bandwidth performance of wdc12, which exceeds DRAM cache

UCDAVIS

Large Graphs Suffer Poor Bandwidth Utilization



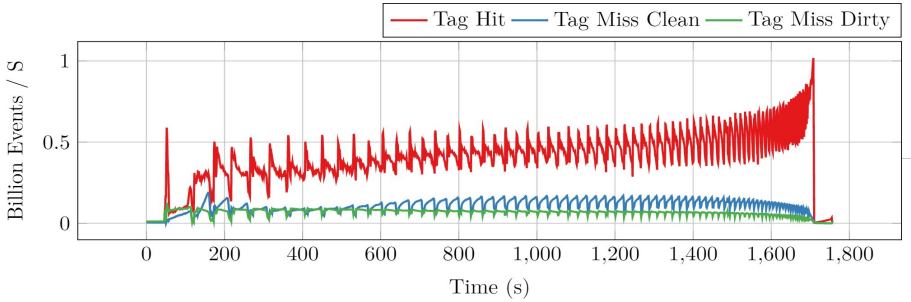
Bandwidth performance of kron30, which fits in DRAM cache

~50% Reduction

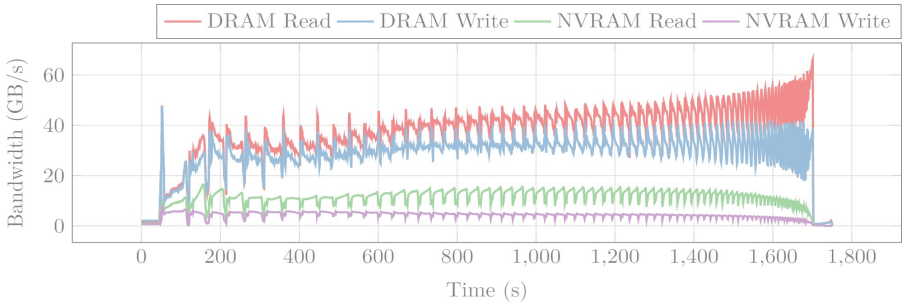
Bandwidth performance of wdc12, which exceeds DRAM cache

UCDAVIS

Large Graphs: Miss-rate and Bandwidth



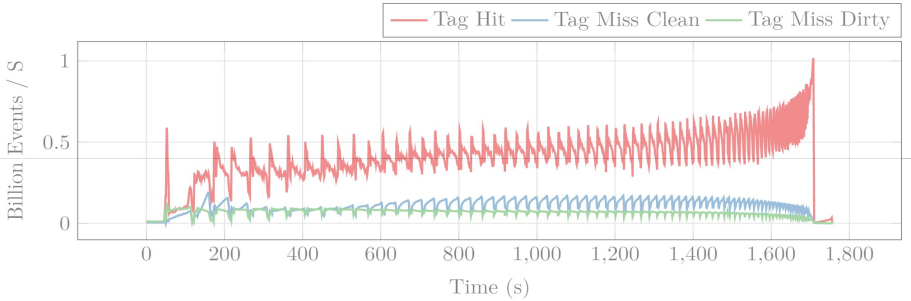
wdc12 Tag Statistics of PageRank



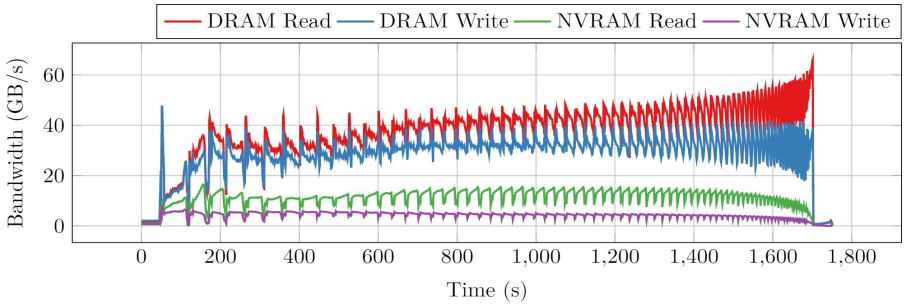
wdc12 Bandwidth Trace of PageRank

UCDAVIS

Large Graphs: Miss-rate and Bandwidth



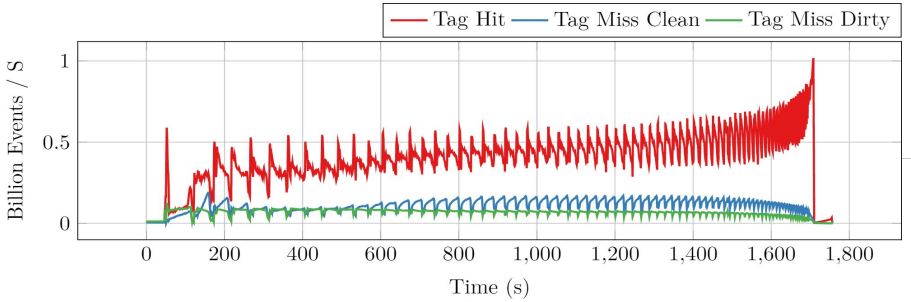
wdc12 Tag Statistics of PageRank



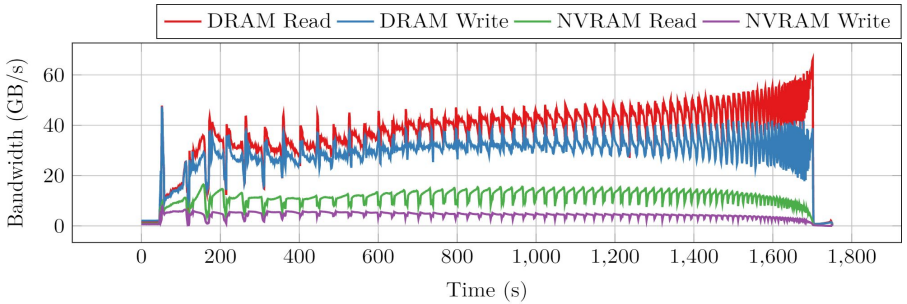
wdc12 Bandwidth Trace of PageRank

UCDAVIS

Large Graphs: Miss-rate and Bandwidth



wdc12 Tag Statistics of PageRank



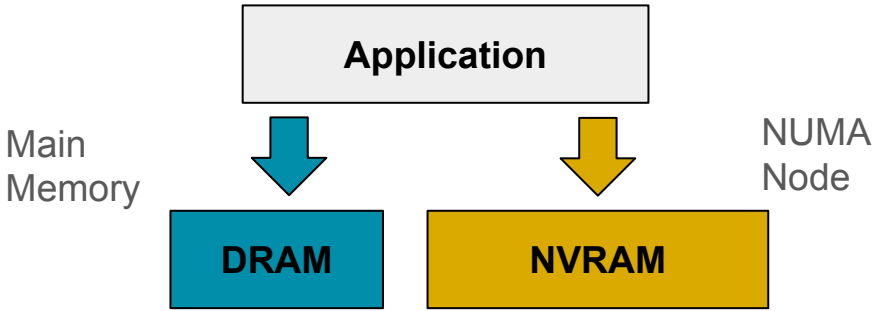
wdc12 Bandwidth Trace of PageRank

Modest miss-rate in graphs but **severe bandwidth drop**

UCDAVIS

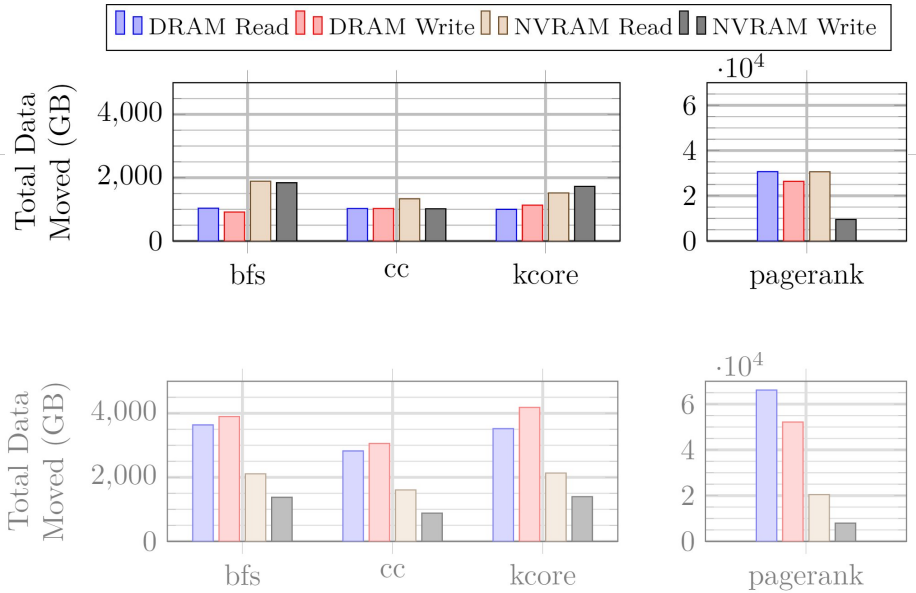
How Much Data is Moved?

App Direct Mode (1LM)



Observe **baseline** data movement of kernels

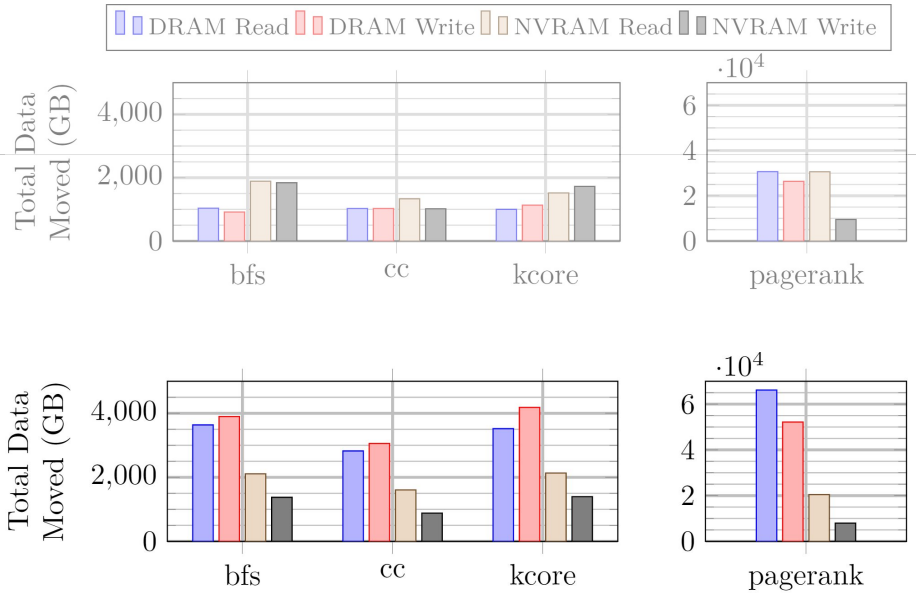
Large Graphs: Significant Data Movement



wdc12 - NVRAM as extra NUMA nodes

wdc12 - NVRAM as system memory, DRAM as cache

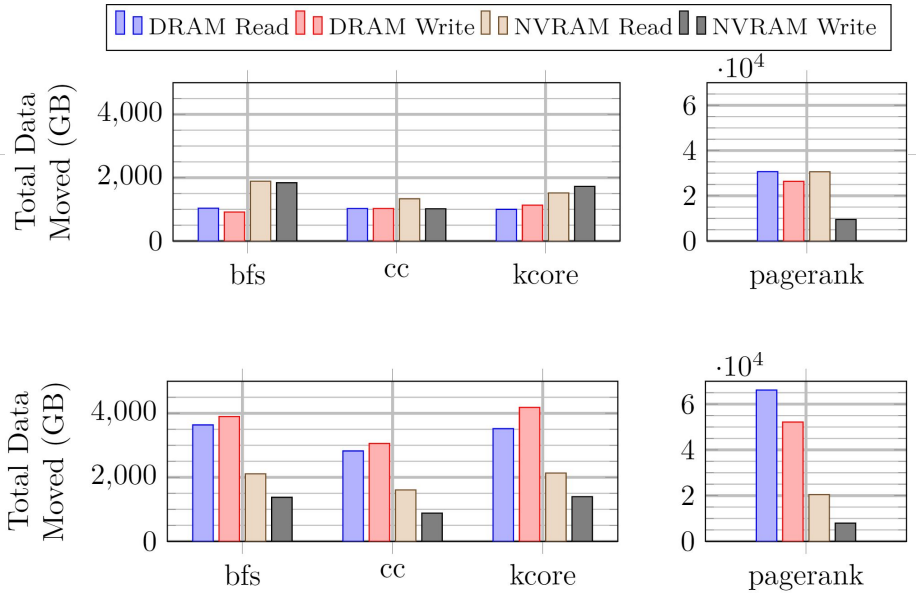
Large Graphs: Significant Data Movement



wdc12 - NVRAM as extra NUMA nodes

wdc12 - NVRAM as system memory, DRAM as cache

Large Graphs: Significant Data Movement



wdc12 - NVRAM as extra NUMA nodes

Excessive movement of data due to **access amplification**

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

AutoTM (Our Group)[1]

CNN Training

Method

Used static information to optimize tensor movement

Results

3x performance improvement over **hardware caches**

Sage (Dhulipala et al.)[2]

Graph Analytics

Method

Graph algorithms that mutate in DRAM only

Results

1.94x speedup over Galois(2LM)

[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)

UCDAVIS

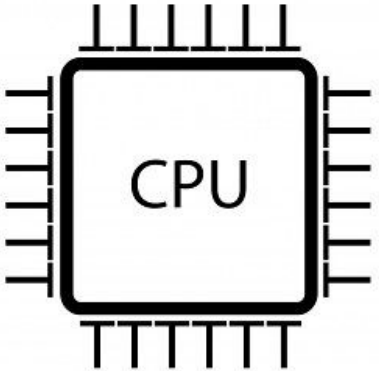
Limitations of Software Data Management

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)?