A Case Against Hardware Managed DRAM Caches for NVRAM Based Systems

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

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

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

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

UC DAVIS
UNIVERSITY OF CALIFORNIA

Executive Summary

Problem

Understanding performance of a **large** DRAM cache for tera-scale NVRAM based systems.

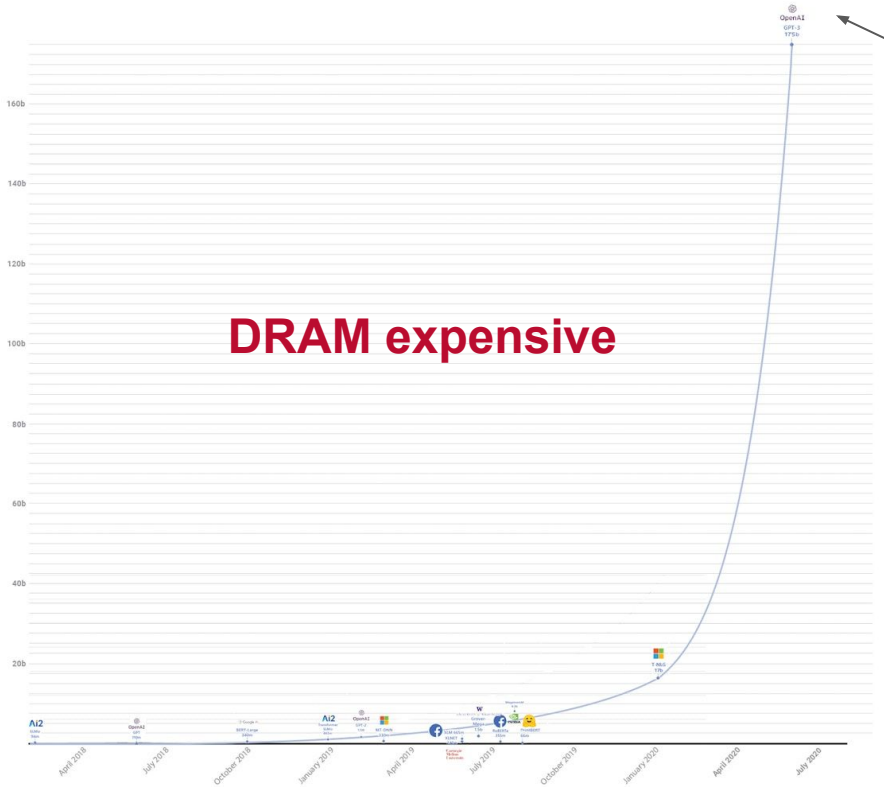
Try to understand these caches

- Microbenchmarks on real hardware
- Case studies with real applications

Show that insights manifest in workloads

- Inflexible direct-mapped policy can lead to a high miss rate
- Significant bandwidth reduction under cache misses
- Potentially many unnecessary dirty writebacks

Emerging Applications Are Growing

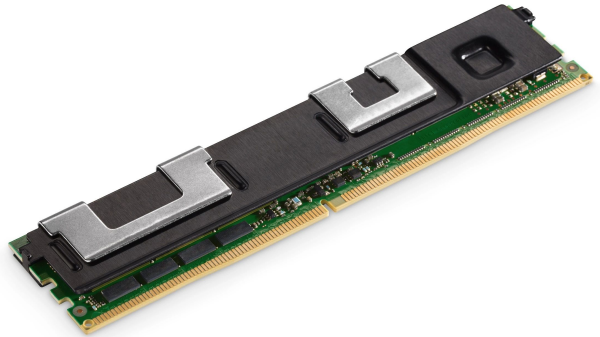


175 Billion Parameters!!

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

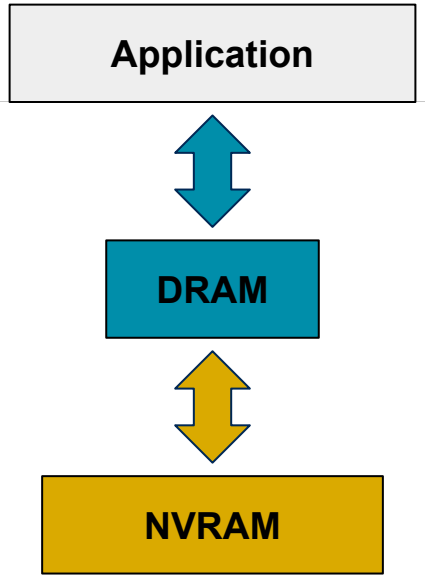
NVRAM Based Servers

- 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**

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)

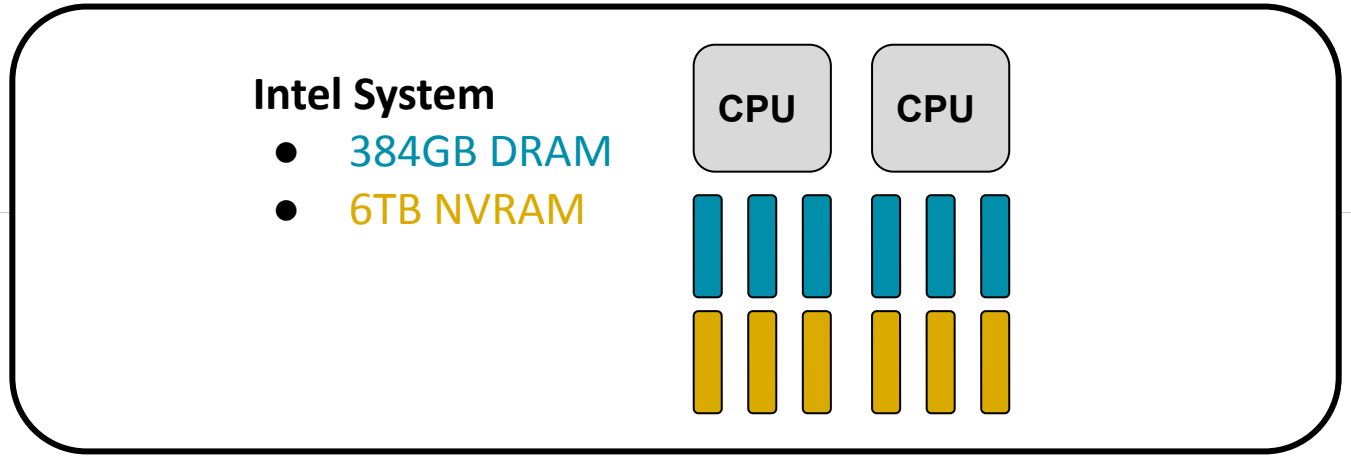
[1] Mark Hildebrand, Jawad Khan, Sanjeev Trika, Jason Lowe-Power, Venkatesh Akella - *AutoTM: Automatic Tensor Movement in Heterogeneous Memory Systems using Integer Linear Programming* (ASPLOS '20)

[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)

Outline

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

DRAM Cache Characteristics



DRAM Cache

- **Direct Mapped** policy

Micro Benchmarks: Cache Study

Goal: Understand access amplification and bandwidth capability.

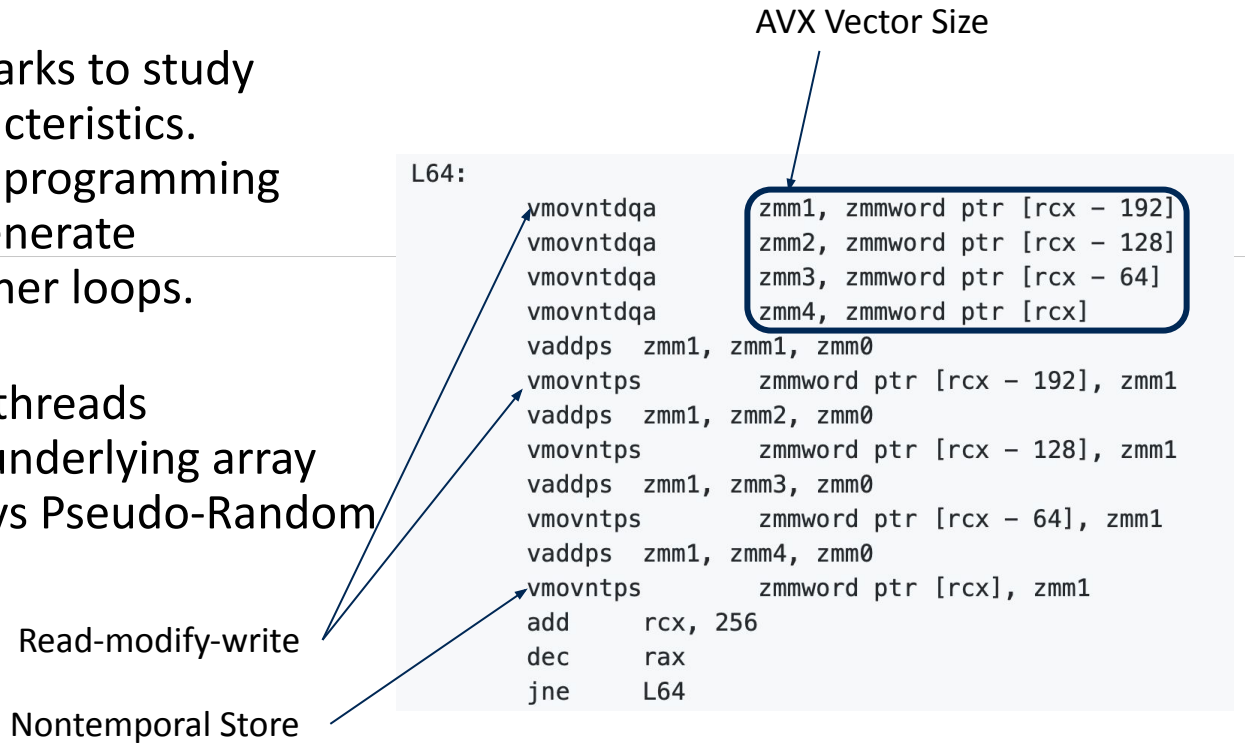
Why: Tracking metadata for a 192GB cache is **hard**.

Strategy

- High performance kernels to generate custom read/write traffic.
- For misses: Large Array (over 2x size of the DRAM cache)
- Hardware performance counters.

Micro Benchmarks

- Suite of benchmarks to study bandwidth characteristics.
- Uses Julia's metaprogramming capabilities to generate low-overhead inner loops.
- Other knobs
  - Number of threads
  - Size of the underlying array
  - Sequential vs Pseudo-Random



Repo: https://github.com/darchr/KernelBenchmarks.jl

DRAM Cache: High 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 access amplification

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**

Dirty misses are **very** expensive!

[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

It’s possible that future implementations can fix these issues, but there are other pitfalls as well.

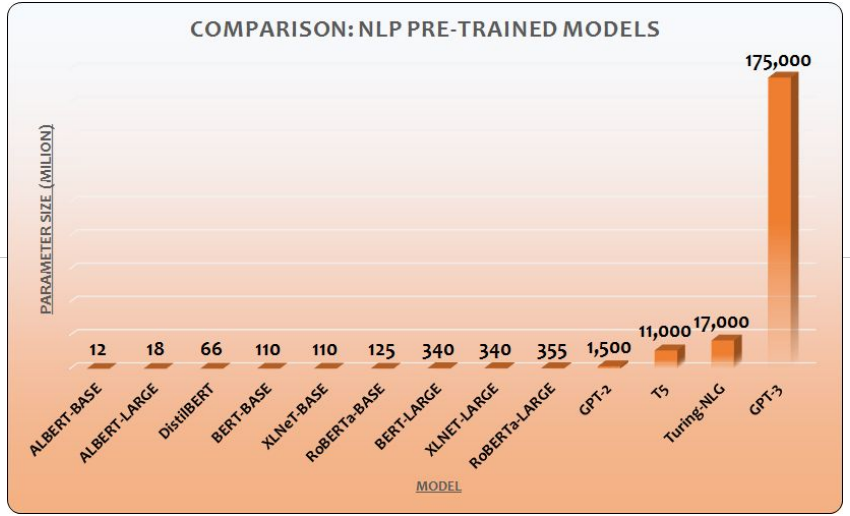
Outline

- Background
- DRAM cache analysis
- Performance in real workload
- Discussion

Case Study - DNN Training

Why

- DNN models keep getting larger
- Predictable memory access pattern to study



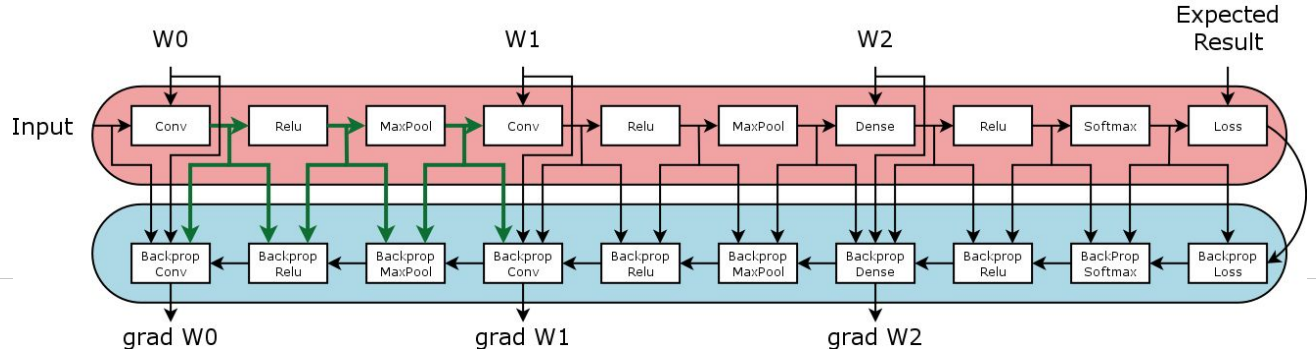
AutoTM[1]

- Previous work optimizing tensor location and movement during DNN training
- Modifications to the ngraph[2] compiler for data analysis.
- AutoTM up to **3x faster** than 2LM

[1] https://github.com/darchr/AutoTM
[2] https://github.com/openvinotoolkit/openvino
Image from: https://medium.com/analytics-vidhya/openai-gpt-3-language-models-are-few-shot-learners-82531b3d3122

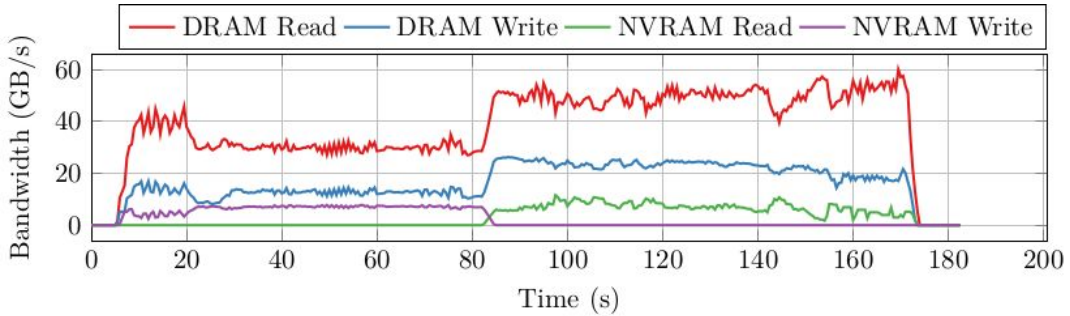
UCDAVIS

DNN Training

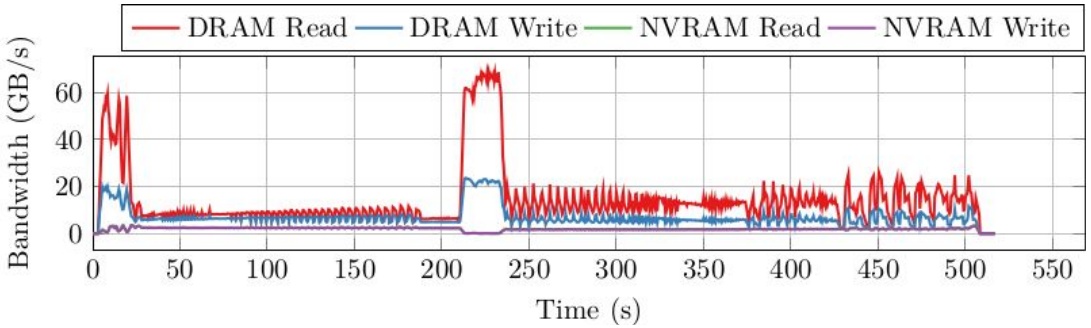


- Workload
  - DNN training for 4 popular networks: Vgg16, Inception-v4, Resnet-200, Densenet-264.
  - Scaled batchsize so maximum memory footprint exceeded 600 GB.
- Data Collection
  - Runtime, HW performance counters, ngraph telemetry.

Densenet-264 Training - AutoTM vs 2LM



Bandwidth trace for a single iteration of training using AutoTM.



Bandwidth trace for a single iteration of training using 2LM.

1LM - Bandwidth looks reasonable:

- Write to NVRAM on the forward pass
- Read from NVRAM on the backward pass

2LM - Odd behavior

- NVRAM writes on both forward and backward pass.
- Periods of high bandwidth at beginning of forward and backward pass.
- **Much** lower average bandwidth.

DRAM cache generates significant unnecessary dirty writebacks.

UCDAVIS

Outline

- Background
- DRAM cache analysis
- Performance in real workload
- Discussion

Takeaways

- DRAM caches allow for quick adoption of the large capacity offered by persistent memory.

But ...

- Bandwidth utilization of 2LM is low, especially for applications with low locality.
- Particularly - dirty writebacks are very expensive.
- Hardware only doesn’t know if memory is free or not, so must always be conservative.

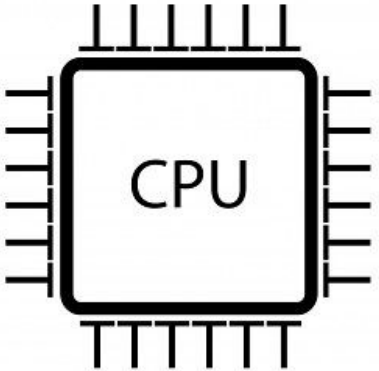
Limitations of Software Data Management

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)?

Acknowledgements

- Intel Corporation
- NSF - Grant No. CNS-1850566
- Members of the Davis Computer Architecture Research Group

UCDAVIS

A Case Against Hardware Managed DRAM Caches for NVRAM Based Systems

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

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

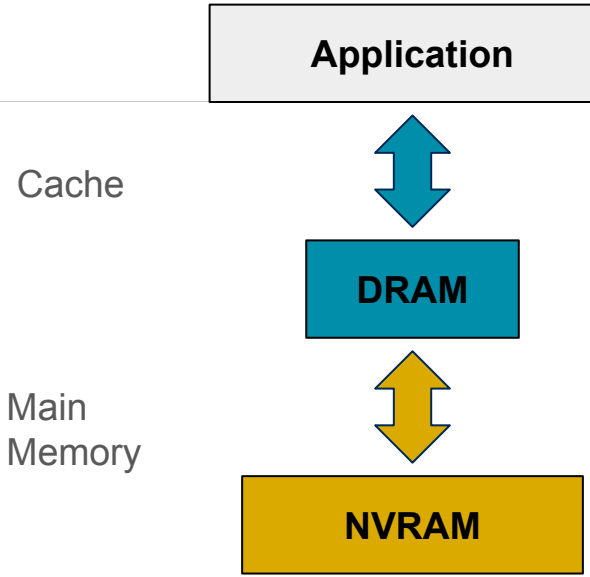
Jason Lowe-Power
CS Dept., UC Davis
jlowepower@ucdavis.edu

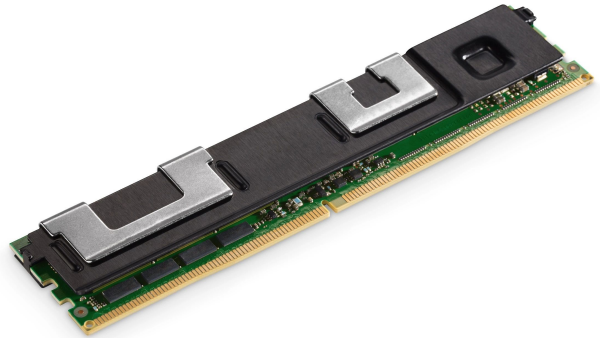
Venkatesh Akella
ECE Dept., UC Davis
akella@ucdavis.edu

UC DAVIS
UNIVERSITY OF CALIFORNIA

DRAM as Cache

Memory Mode (2LM)





Smaller DRAM used to **mask higher latency**

Summary

What

- Microbenchmarks to understand cache performance.
- Two case studies showing superiority of software managed memory over hardware managed.

Conclusions

- Bandwidth is not used efficiently with a high miss rate.
  - Metadata Tracking
  - Small cache granularity
- Without application insight, cache has to be conservative.