WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 839: Design the Next-Generation Database

Lecture 10: NVM1

Xiangyao Yu

2/20/2020

Announcements

Upcoming deadlines:

- Form groups: **Feb. 27**
- Proposal due: **Mar. 10**

Send me names of your group members by next Thursday (Feb. 27)

Example Project Ideas

Paper survey:

- Multicore or distributed concurrency control, parallel logging, deterministic database, GPU DB, NVM DB, RDMA-based DB, cloud DB, graph DB, IoT DB, etc.

Evaluation project:

- Pick a distributed application (e.g., spark, graph-DB), compare its performance on a cluster vs. a single server with Optane NVM.
- Compare performance of OLTP or OLAP or HTAP database engines

Example Project Ideas

Research project:

- Explore opportunities in epoch-based concurrency control
- Design deterministic database that does not require knowing read/write sets
- Deploying g-join and evaluate its performance over other join algorithms
- Study the best way to implement Silo in a distributed environment
- Explore how an OLAP engine can benefit from NVM
- Design a OLTP/OLAP engine using AWS Lambda
- Supporting string processing in GPU
- Study how to support UDF in GPU
- **Or any ideas that came out of our in-class discussion**

Discussion Highlights

Key to Q100’s high performance?

- Streaming to reduce data movement, pipeline parallelism, optimized for TPC-H, efficient instructions, fewer memory writes

DPU vs. optimized CPU?

- Q100 is better due to specialization (1)
- DPU may be better than Opt-CPU on TPC-H (2)
- Unclear (1)

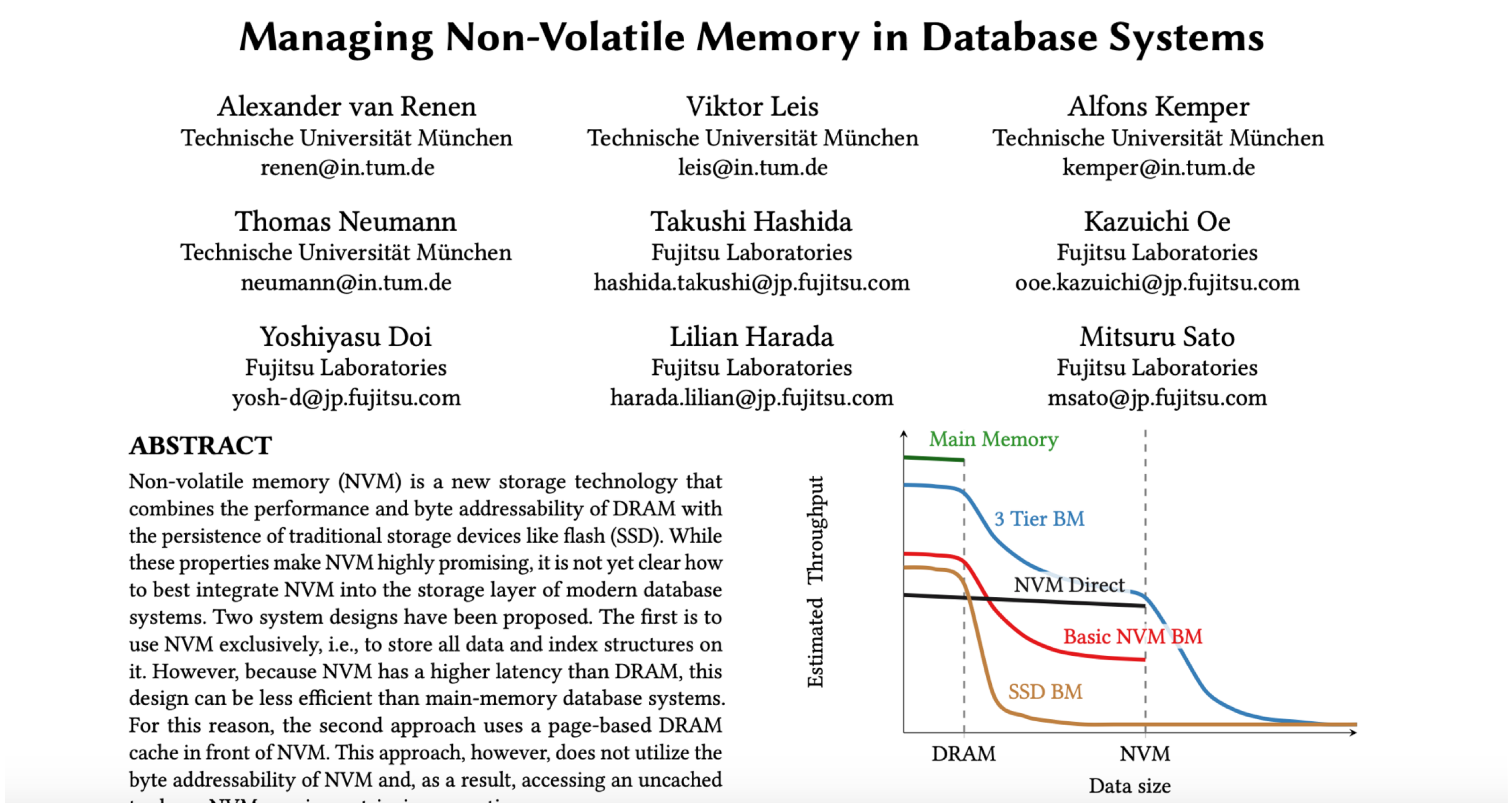
Limitation of Q100?

- Communication (between Q100 and CPU, between tiles) may be a bottleneck
- Large overhead when table is big
- Extra compiler and software development overhead
- Cost of special hardware
- UDF

Optimization goals of database accelerators?

- Minimizing data movement, maximize memory bandwidth. Minimizing PCIe transfer,. Process in memory?
- Data placement, scheduling, concurrency control, energy efficiency

Today’s Paper



Managing Non-Volatile Memory in Database Systems

Alexander van Renen
Technische Universität München
renen@in.tum.de

Viktor Leis
Technische Universität München
leis@in.tum.de

Alfons Kemper
Technische Universität München
kemper@in.tum.de

Thomas Neumann
Technische Universität München
neumann@in.tum.de

Takushi Hashida
Fujitsu Laboratories
hashida.takushi@jp.fujitsu.com

Kazuichi Oe
Fujitsu Laboratories
ooe.kazuichi@jp.fujitsu.com

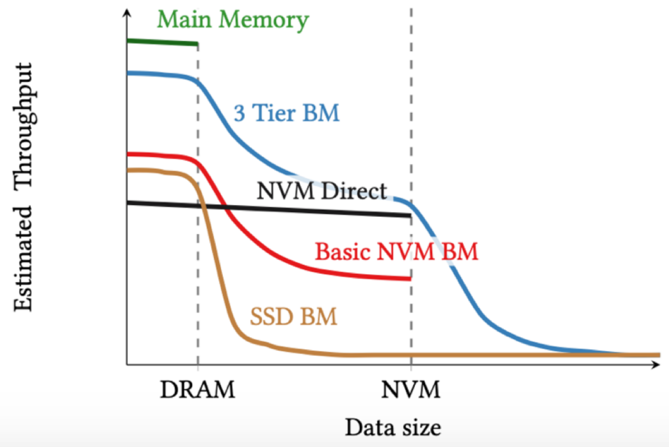
Yoshiyasu Doi
Fujitsu Laboratories
yosh-d@jp.fujitsu.com

Lilian Harada
Fujitsu Laboratories
harada.lilian@jp.fujitsu.com

Mitsuru Sato
Fujitsu Laboratories
msato@jp.fujitsu.com

ABSTRACT

Non-volatile memory (NVM) is a new storage technology that combines the performance and byte addressability of DRAM with the persistence of traditional storage devices like flash (SSD). While these properties make NVM highly promising, it is not yet clear how to best integrate NVM into the storage layer of modern database systems. Two system designs have been proposed. The first is to use NVM exclusively, i.e., to store all data and index structures on it. However, because NVM has a higher latency than DRAM, this design can be less efficient than main-memory database systems. For this reason, the second approach uses a page-based DRAM cache in front of NVM. This approach, however, does not utilize the byte addressability of NVM and, as a result, accessing an uncached



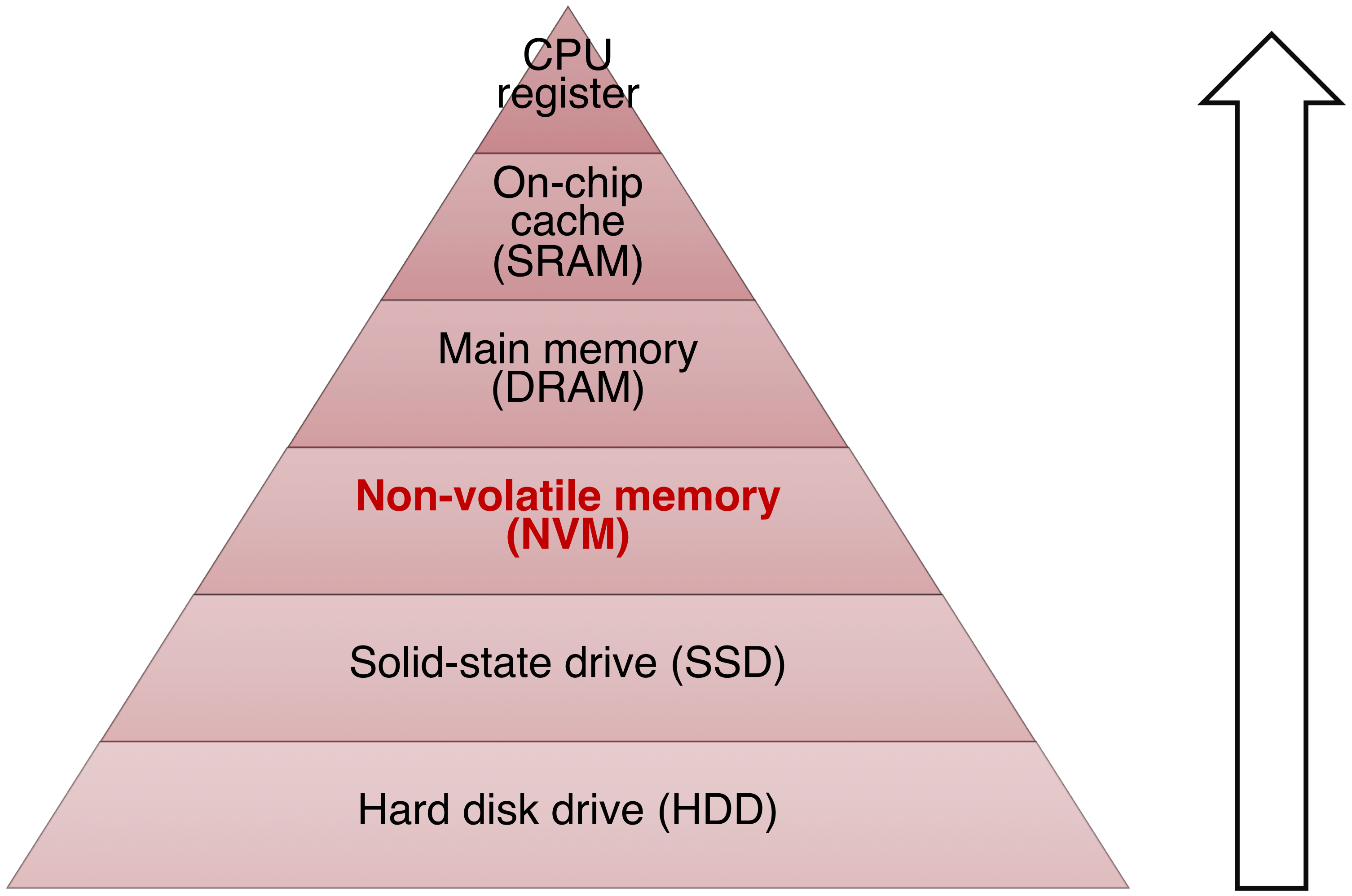
SIGMOD 2018

Today’s Agenda

NVM Basics

Managing Non-Volatile Memory in Database Systems

Storage Hierarchy



Lower latency

Higher bandwidth

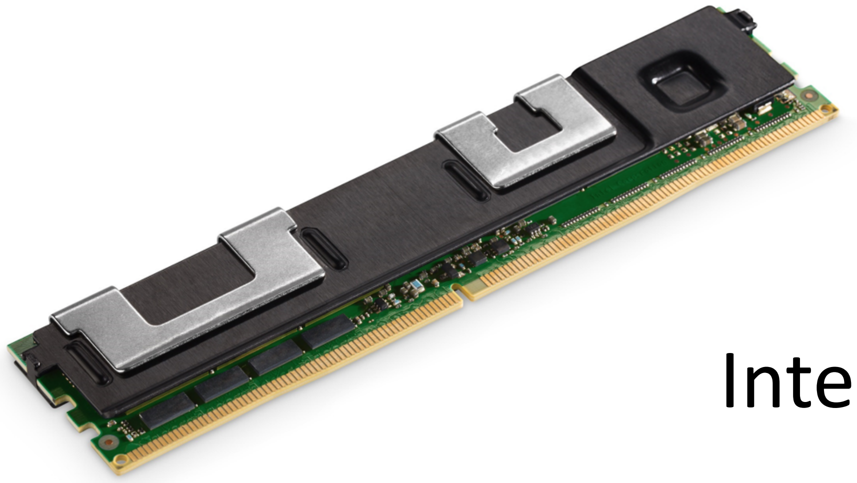
Smaller capacity

Higher cost

More energy

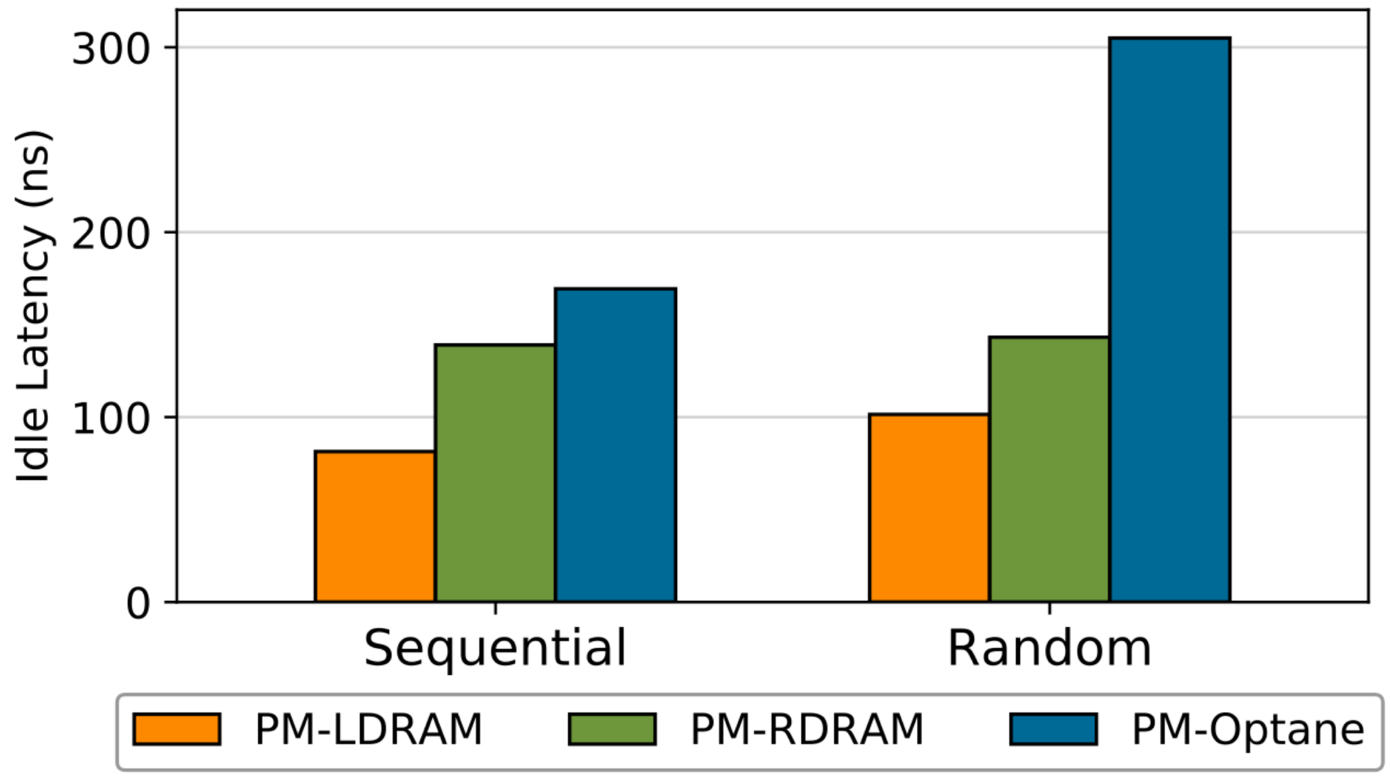
DRAM vs. NVM vs. SSD

	DRAM	NVM	SSD/HDD
Access granularity	**Byte addressable**	**Byte addressable**	Block storage
Durability	Volatile	**Non-volatile**	**Non-volatile**



Intel® Optane™ Memory

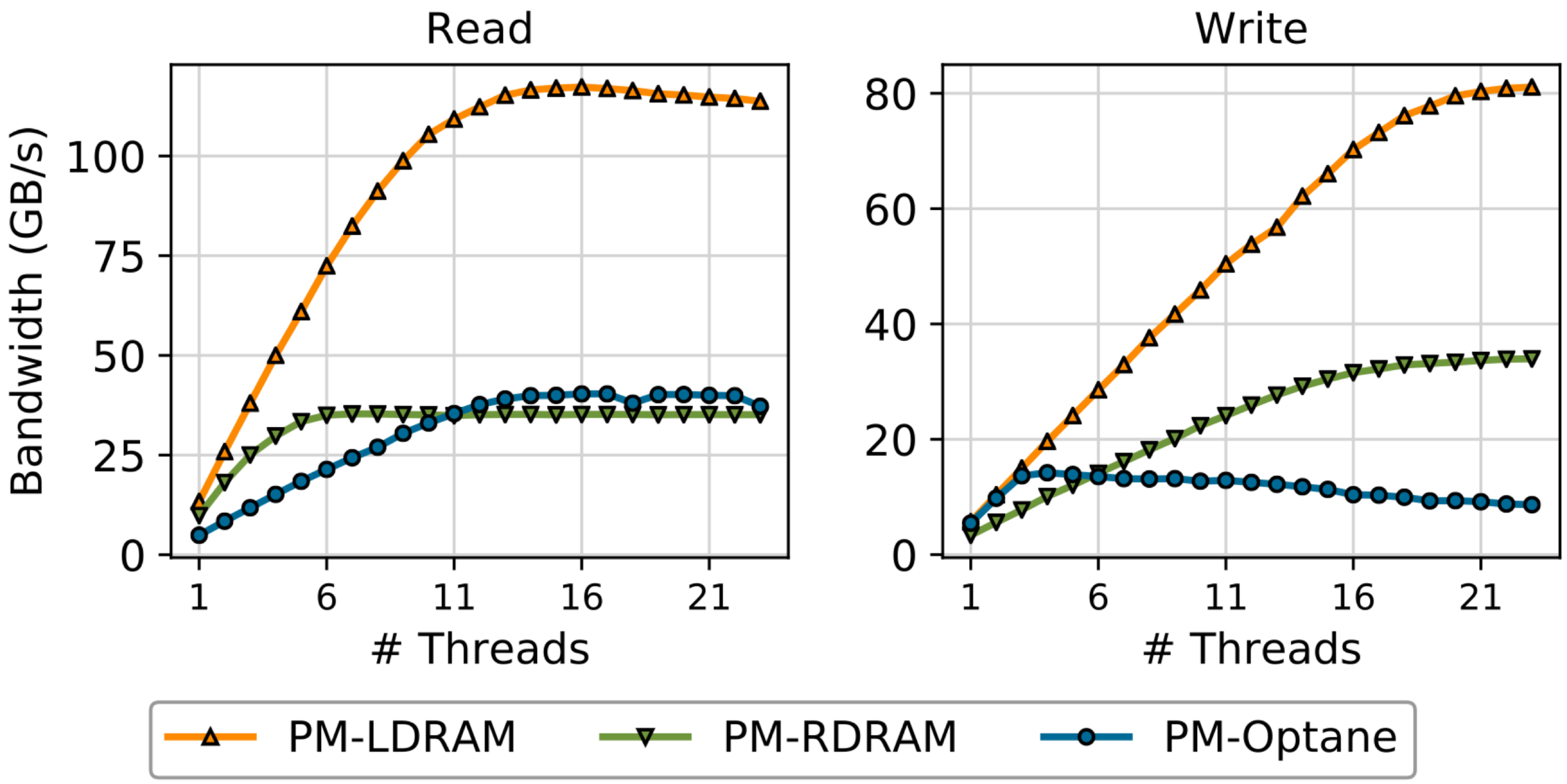
NVM Performance – Read Latency



Random read latency is 305 ns which is 3x slower than local DRAM
Sequential read latency is 2x better than random read latency

* Figure from *Basic Performance Measurements of the Intel Optane DC Persistent Memory Module*

NVM Performance – Bandwidth



Per-DIMM max read bandwidth: 6.6 GB/s, max write bandwidth: 2.3 GB/s
Read/write bandwidth gap is 2.9x for NVM (1.3x for DRAM)

* Figure from *Basic Performance Measurements of the Intel Optane DC Persistent Memory Module*

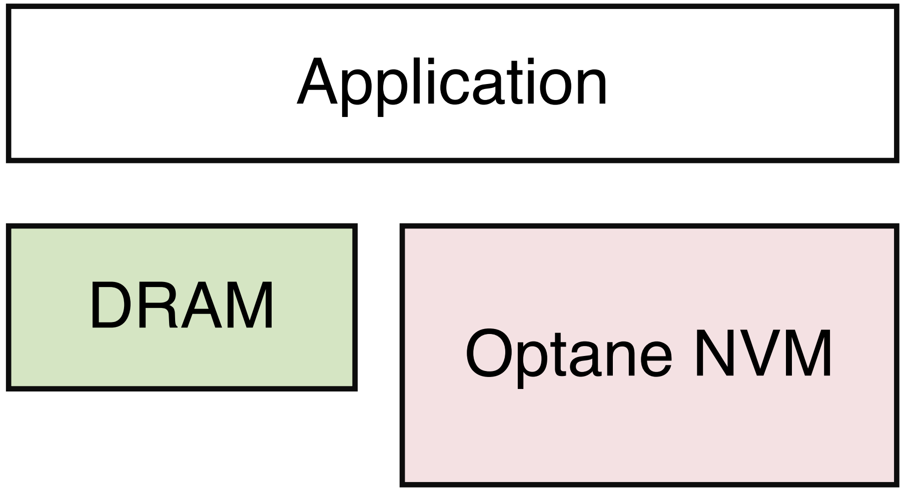
NVM Performance – Cost

Type	Density	Price	$/GB
DRAM	32GB	$374.71	$11.71
DRAM	64GB	$708.25	$11.07
DRAM	128GB	$1,913.21	$14.95
DRAM	256GB	$5,952.00	$23.25
Optane	**128GB**	**$577.00**	**$4.51**
Optane	**256GB**	**$2,125.00**	**$8.30**
Optane	**512GB**	**$6,751.00**	**$13.19**

Data from https://thememoryguy.com/intels-optane-dimm-price-model/

Operation Modes

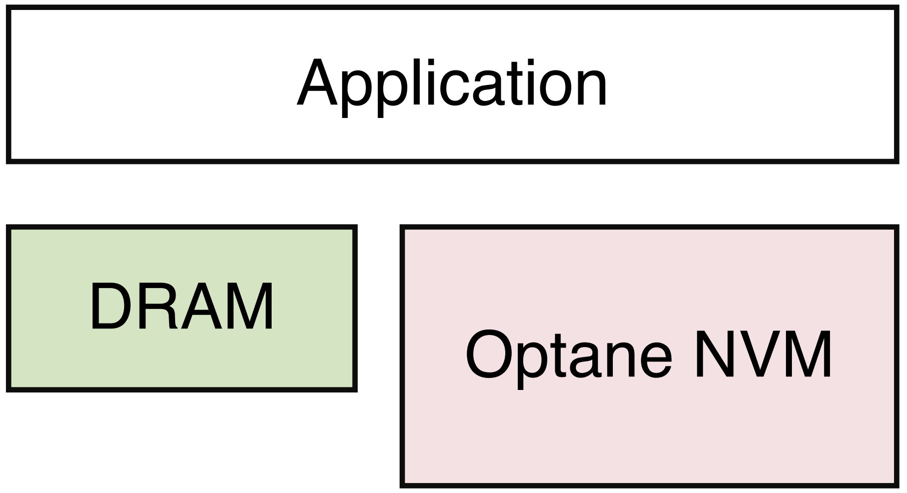
App direct mode



DRAM and NVM in different address space

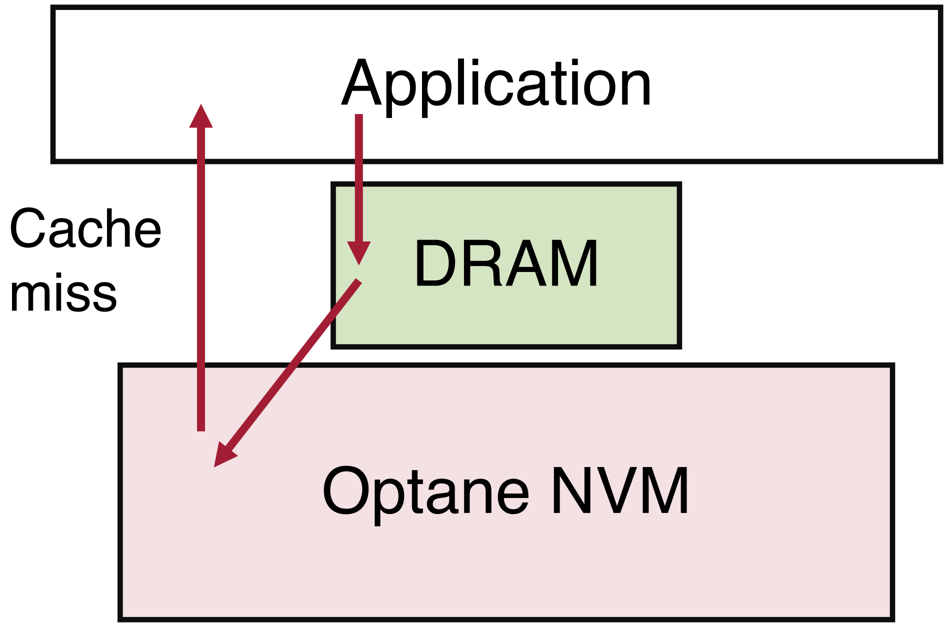
Operation Modes

App direct mode



DRAM and NVM in different address space

Memory Mode



DRAM as a cache, managed by hardware

How to Use NVM in a Database?

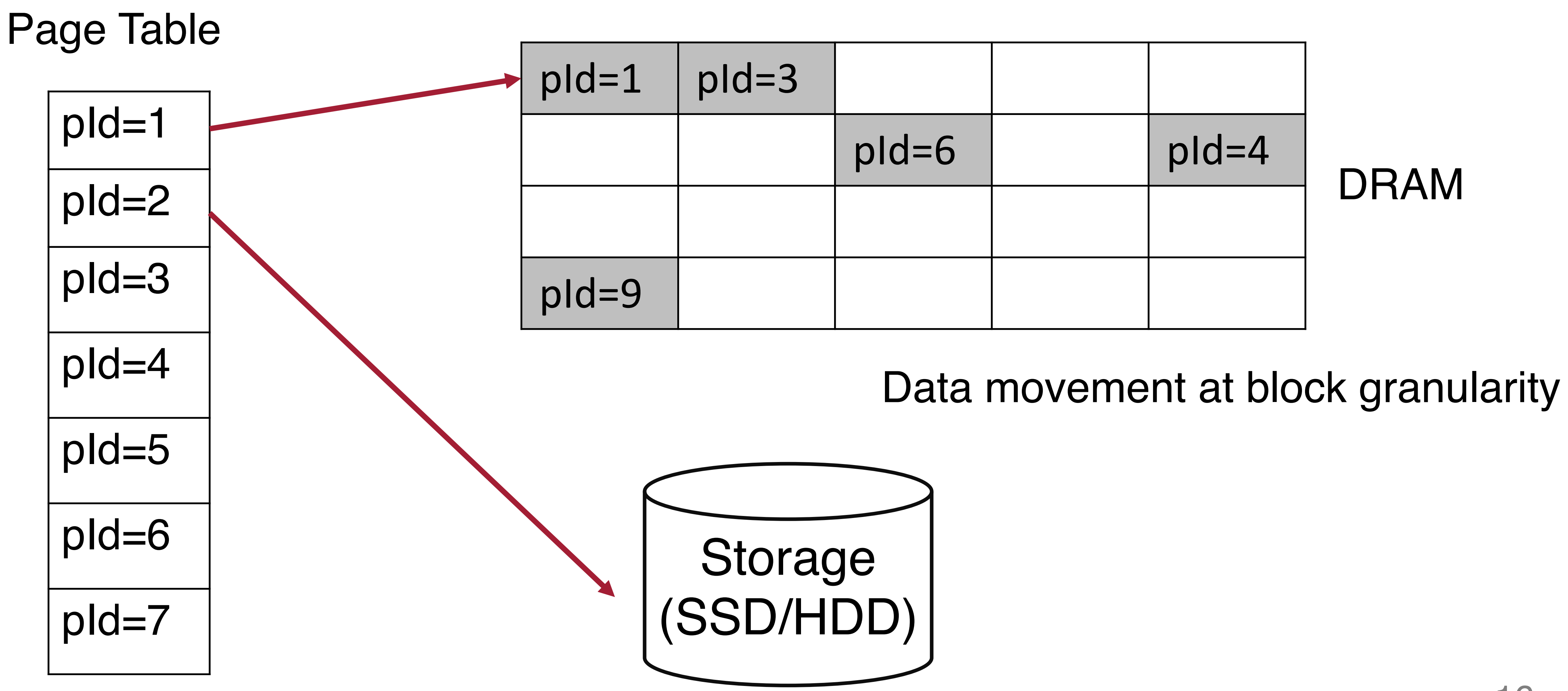
NVM Direct Mode

- NVM as the primary storage
- All changes persistent to NVM when a transaction commits
- Logging can be simplified
- **Will discuss in details next lecture**

DRAM as a Cache

- NVM write throughput is lower than DRAM
- Buffer management (BM) for DRAM data
- Write-ahead logging
- **This paper was written before memory-mode was available**

Buffer Management



Data movement at block granularity

Buffer Management in SSD/HDD vs. NVM

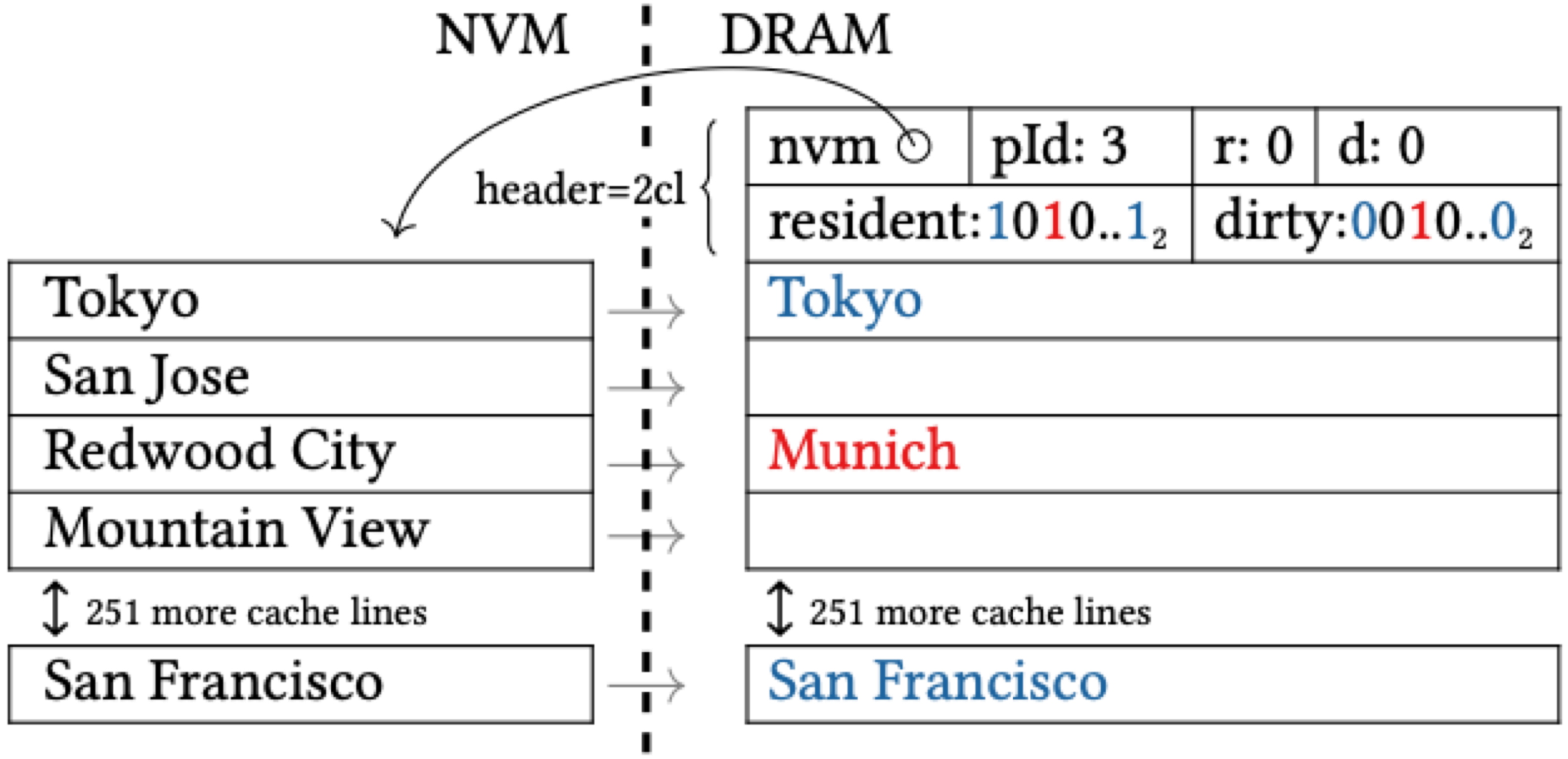
Buffer management in SSD/HDD

- Block storage
- Load a full page at a time (e.g., 16 KB)

Buffer management in NVM

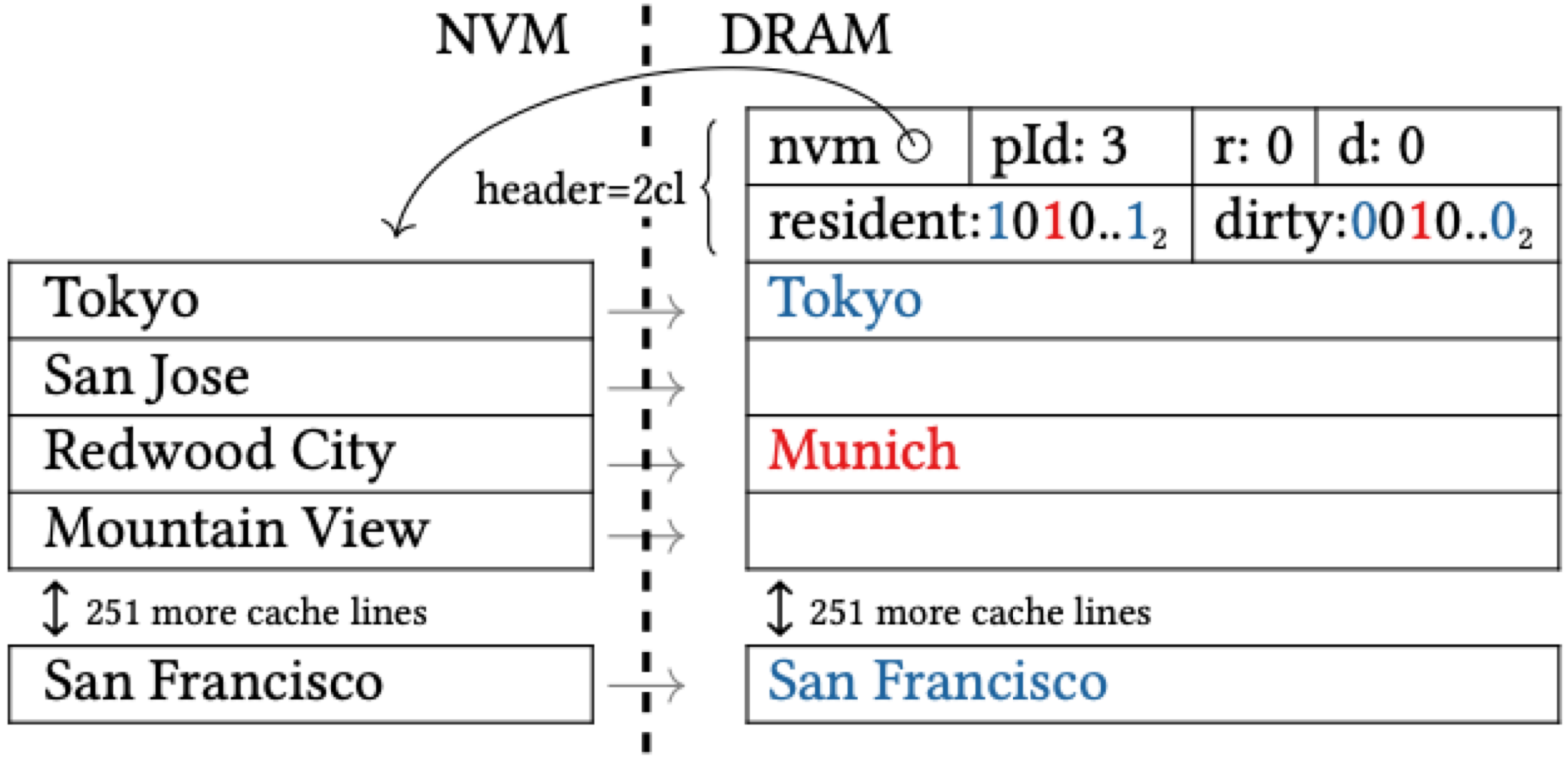
- Byte addressable
- Waste of bandwidth if full pages are loaded
- Loading a cacheline at a time (64 B)

Cache-Line-Grained Pages



Page initially empty, cachelines loaded as they are accessed

Cache-Line-Grained Pages



Page initially empty, cachelines loaded as they are accessed

Overhead: each access checks/updates `resident/dirty` bits

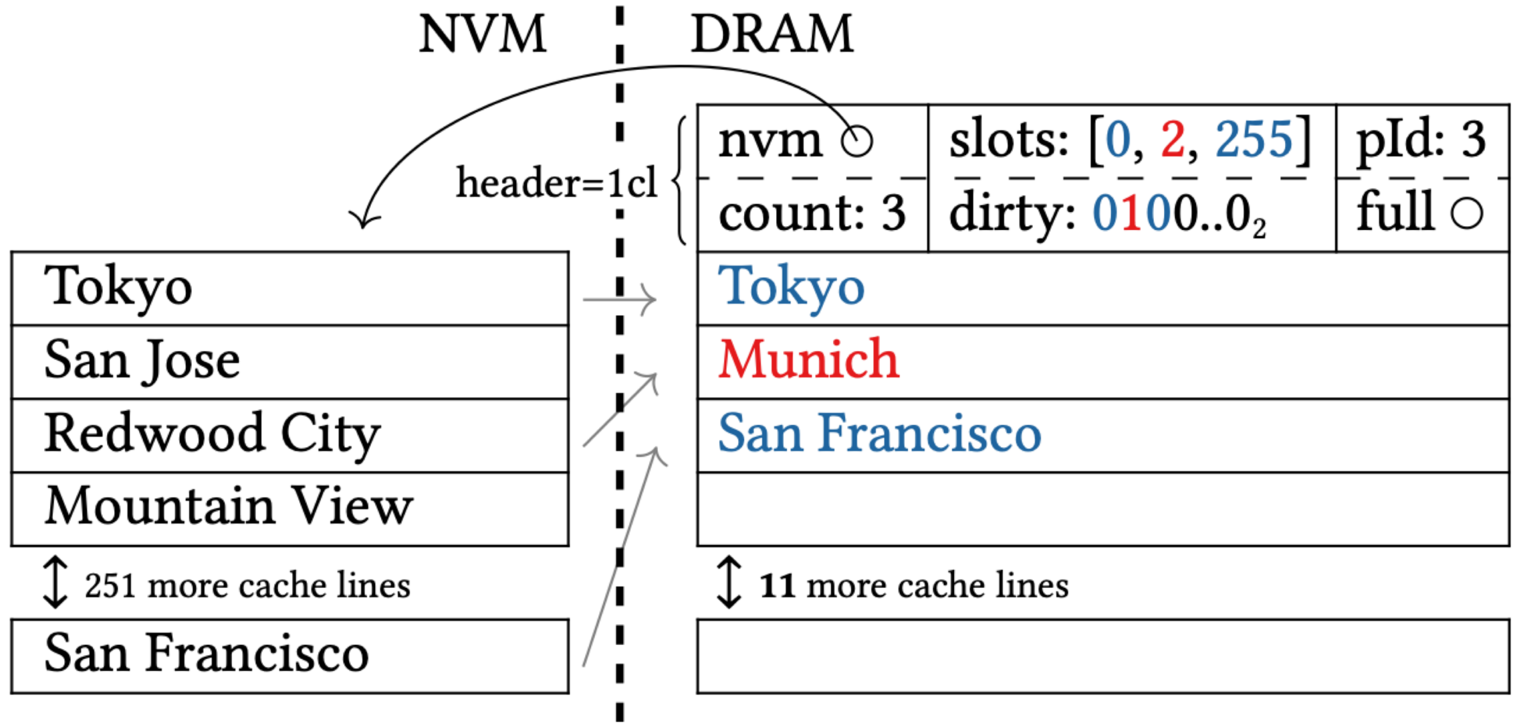
- Does memory-mode solve this problem?

Mini Pages

Page layout consumes more DRAM space than necessary

Mini Pages

Page layout consumes more DRAM space than necessary



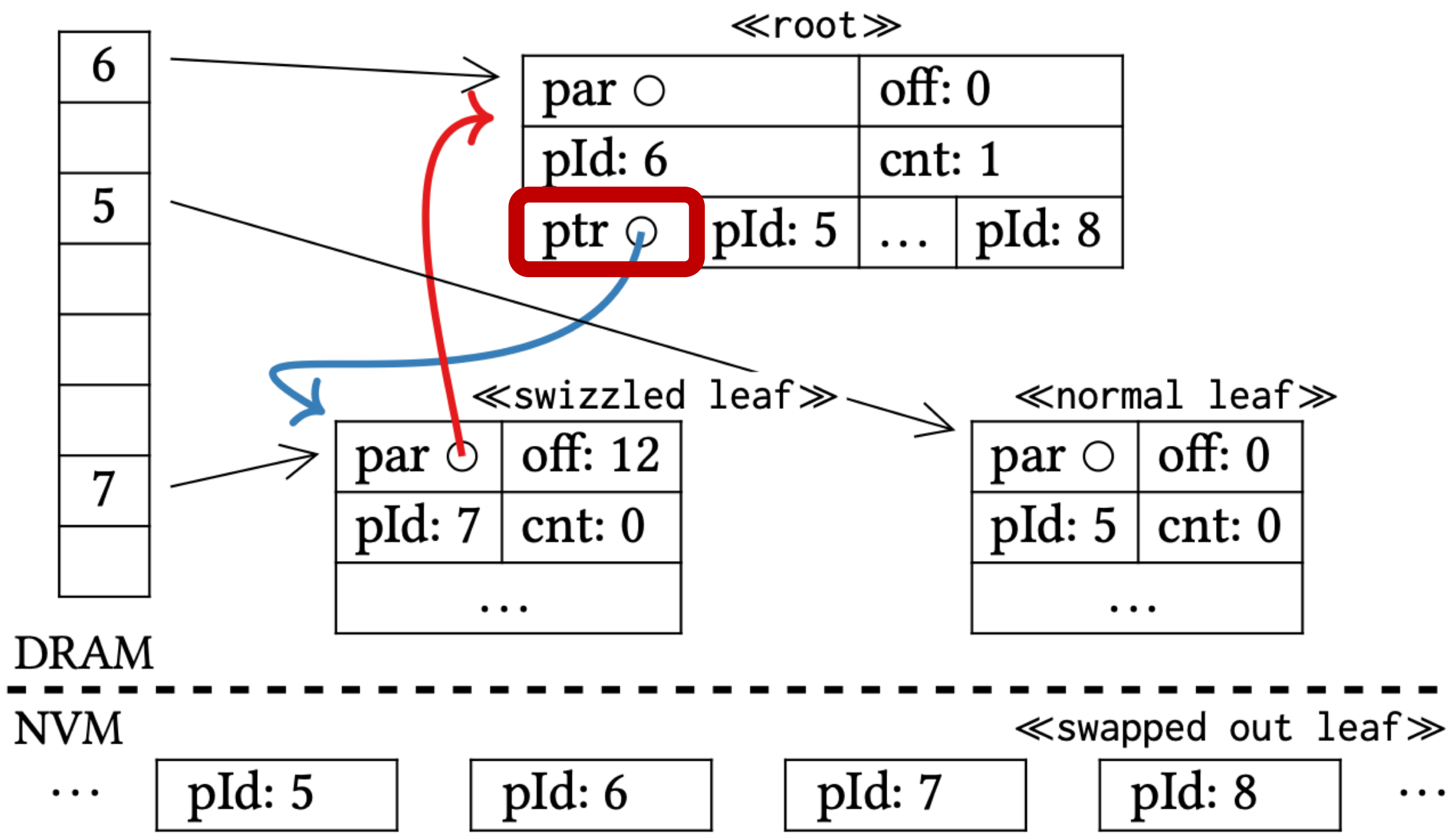
Mini page: a sparse representation of a page

- Cachelines are sorted
- Promote to full page when a mini-page is full

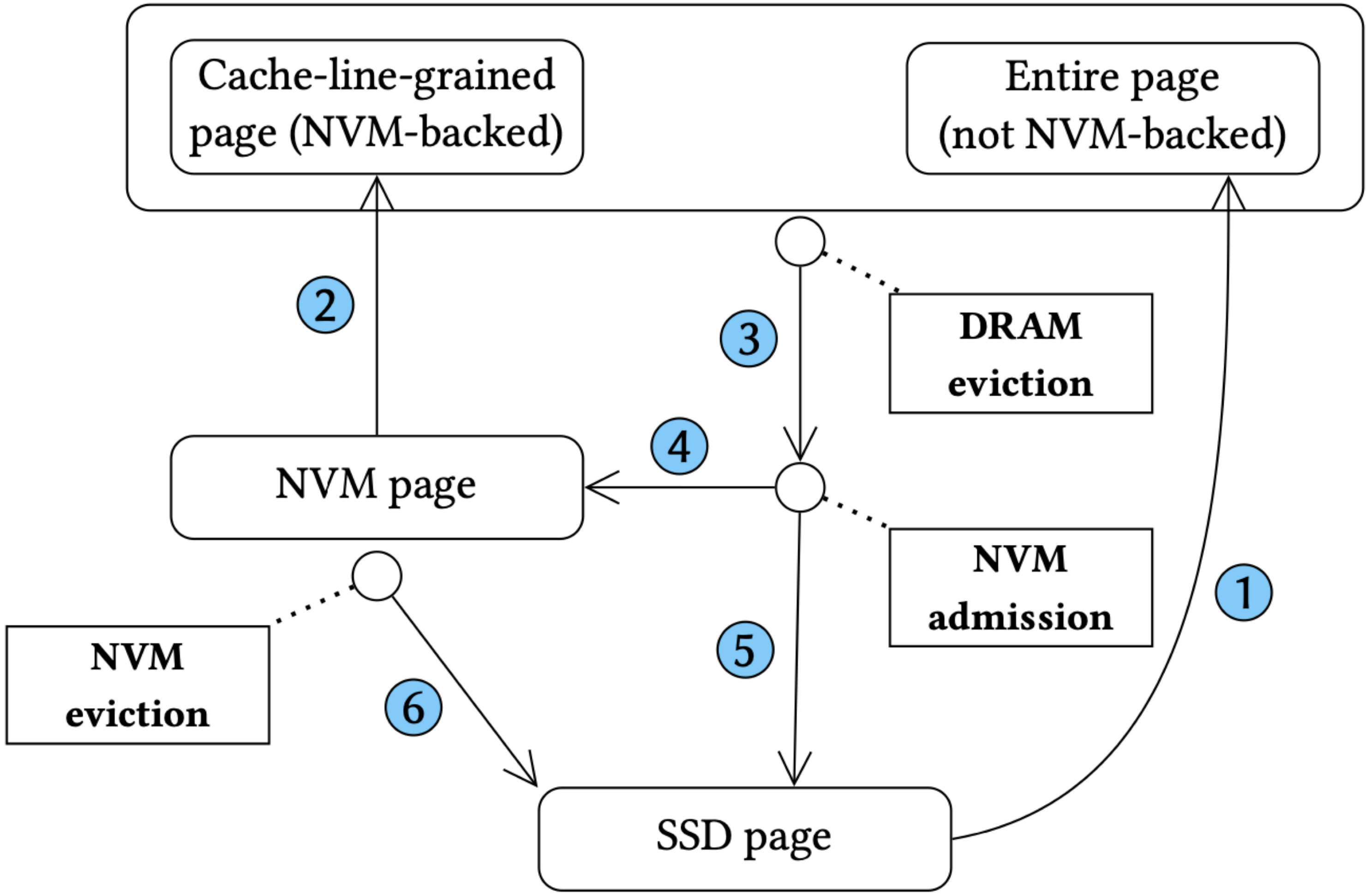
Pointer Swizzling

Reduce overhead of page table lookup

- Store pointer rather than pageID if page is in main memory
- Cannot swap out a page before its children



Three-Tier Buffer Management

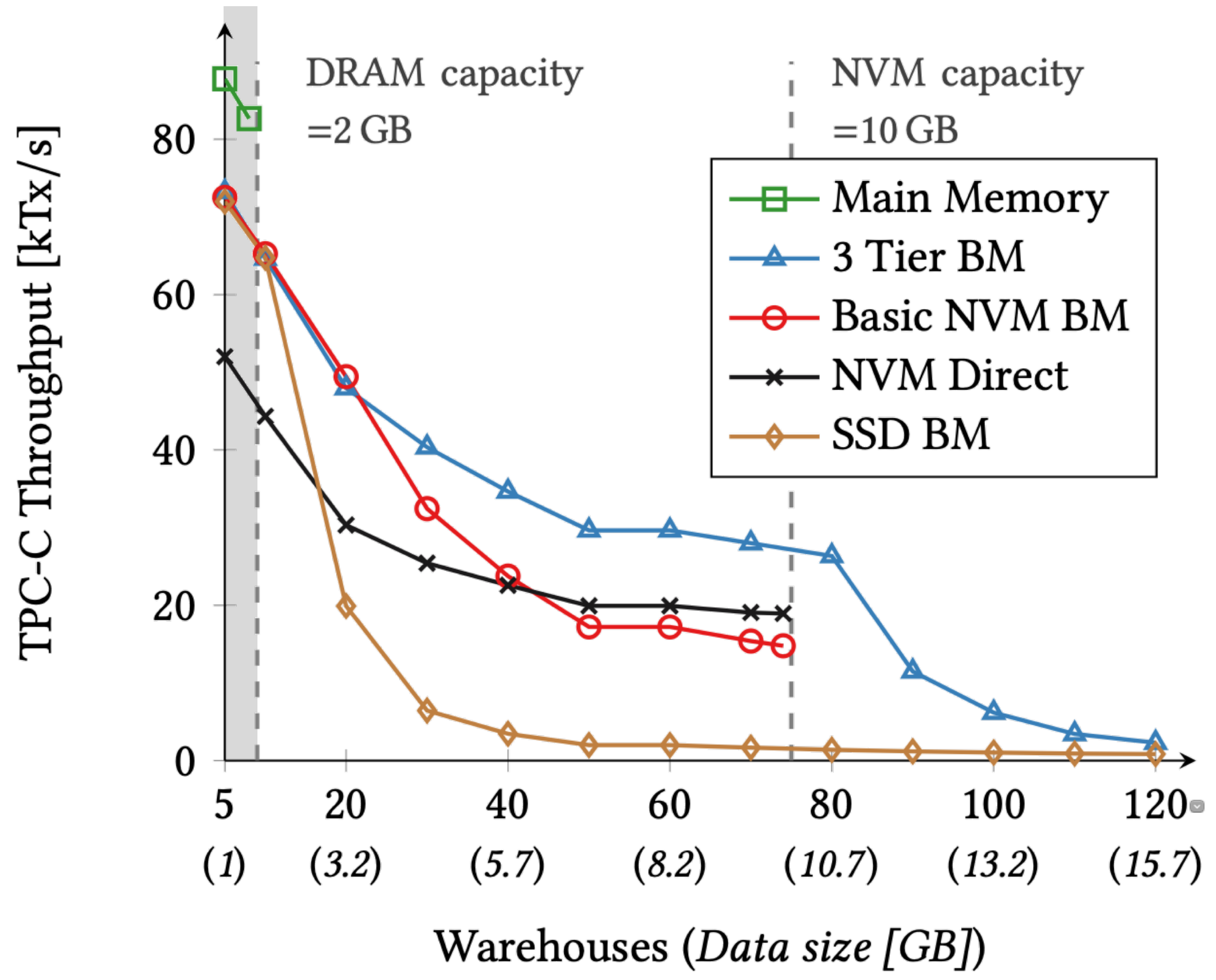

Cache-line-grained page (NVM-backed)
Entire page (not NVM-backed)
DRAM eviction
NVM admission
NVM eviction
NVM page
SSD page
1
2
3
4
5
6

1. DRAM & NVM miss
   - Directly load to DRAM
2. NVM hit
   - Load cache-line-granted page to DRAM
3. DRAM eviction
   - Clock (second-chance)

4&5. NVM admission
   - Admission set (second-chance)

6. NVM eviction
   - Clock

Evaluation – TPC-C



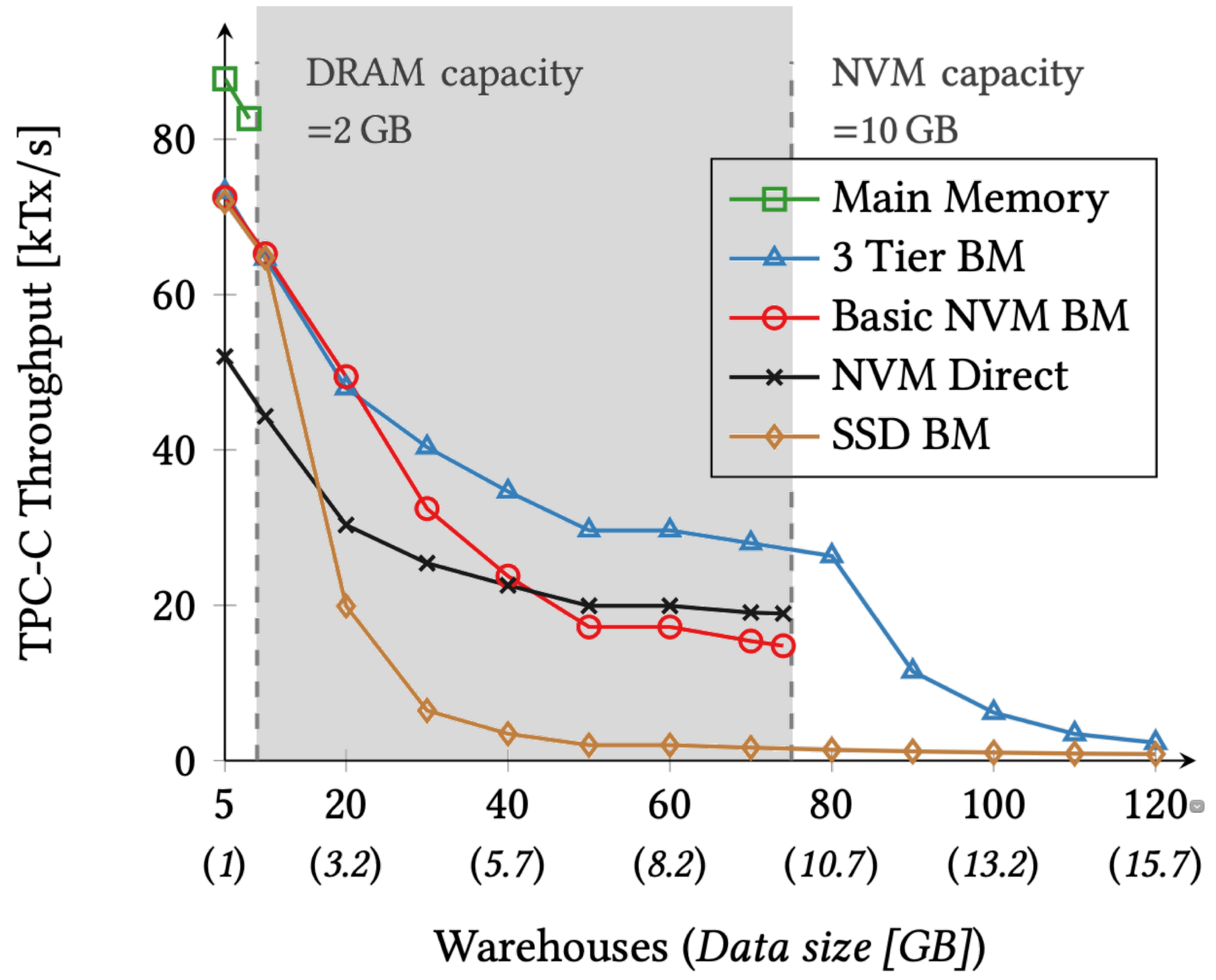
Data fit in DRAM

Main memory has the best performance

3-Tier, Basic NVM BM, and SSD BM have similar performance

NVM direct has worse performance

Evaluation – TPC-C



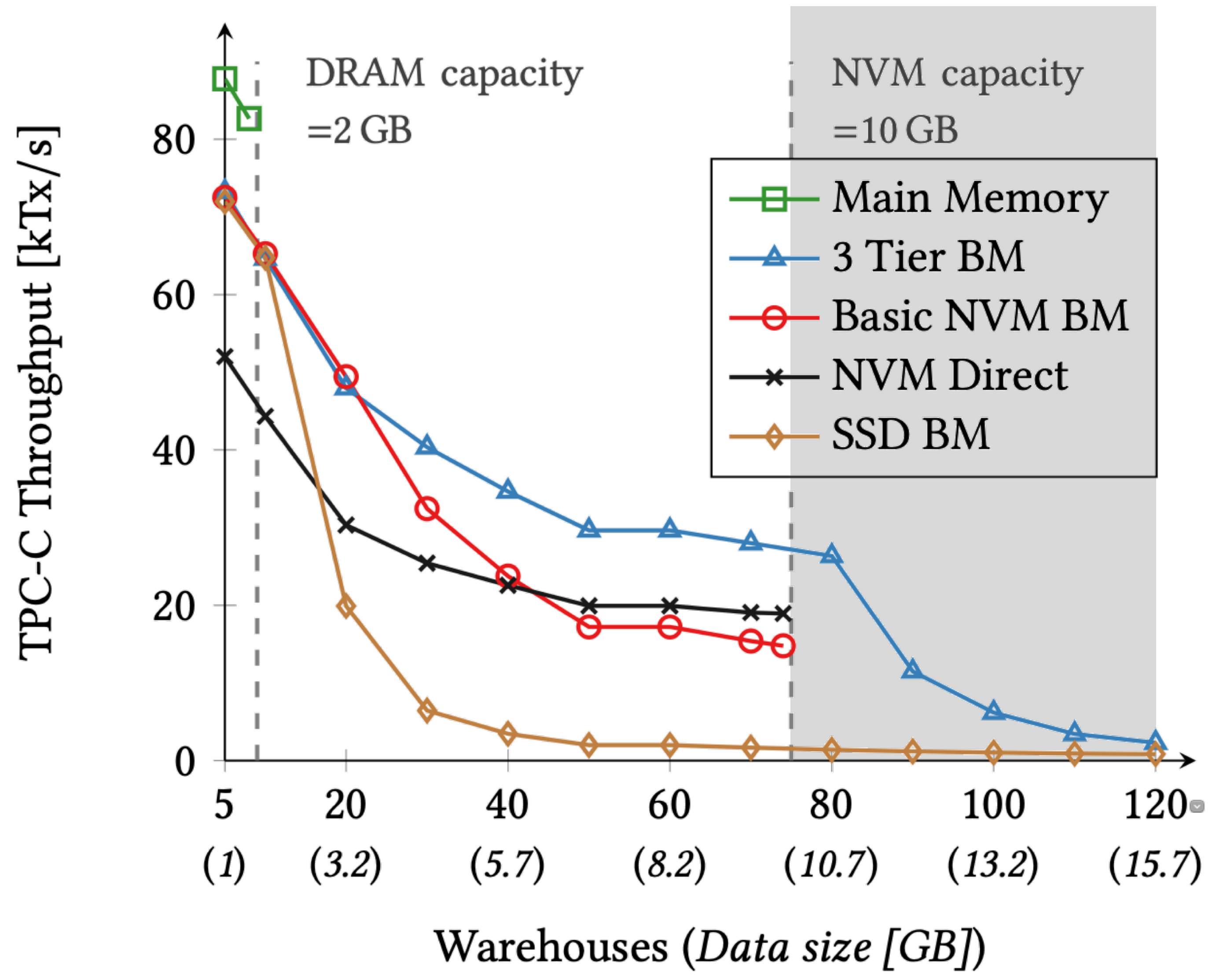
Data fit in NVM

Main memory has insufficient space

3 Tier BM outperforms Basic NVM BM and NVM Direct

SSD BM has the worse performance

Evaluation – TPC-C

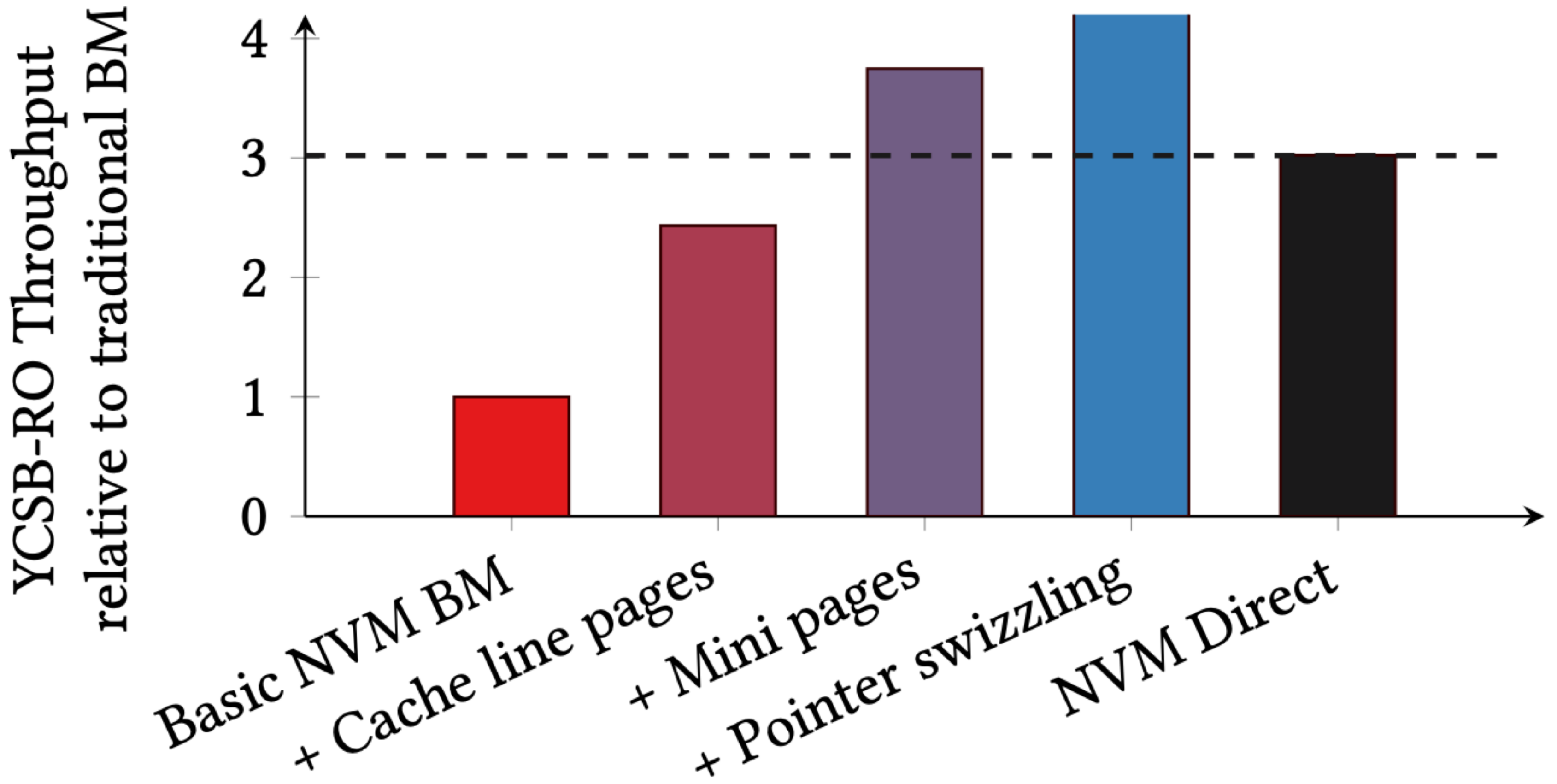


Data fit in SSD

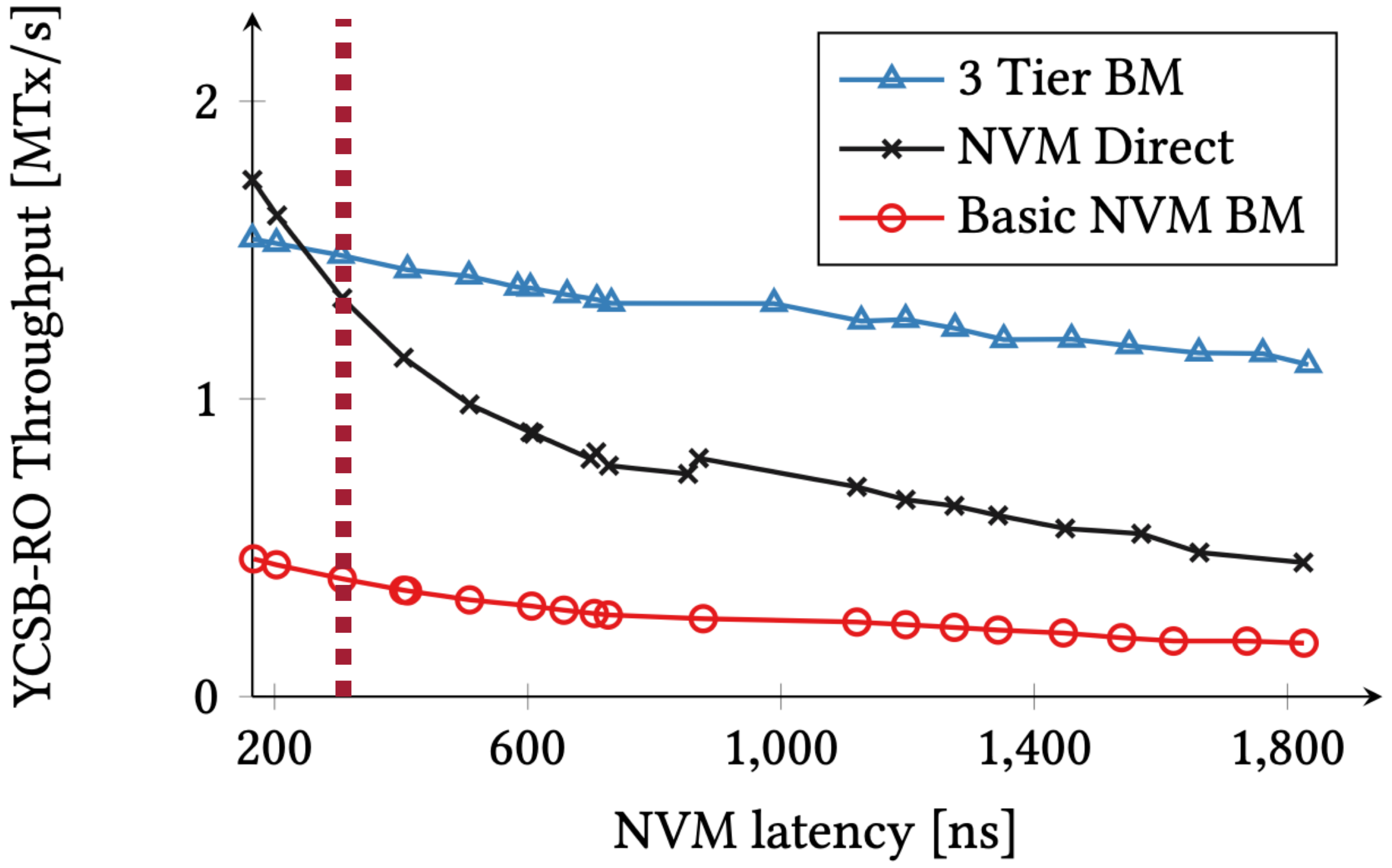
Only 3-Tier BM and SSD BM work

3-Tier BM has better performance

Evaluation – Performance Drill Down



Evaluation – NVM Latency



Summary

NVM: new device in the storage hierarchy

- Byte-addressable
- Non-volatile

Taking advantage of byte-addressability to improve performance

- Cache-Line-Grained pages
- Mini pages
- Pointer swizzling
- Three-tier buffer management

NVM – Q/A

Fault tolerance

- REDO/UNDO write-ahead logging (Why both REDO and UNDO?)

Writing to DRAM increases NVM endurance

What about the non-volatile property?

How does the system compare to using NVM as a cache of disks?

Is NVM practical? Obstacles of wider adoption?

Worry of systems being too complex in the future

Group Discussion

The paper was written before memory-mode existed. How would memory-mode affect the design proposed in the paper?

What are the advantages of implementing an NVM DB over the app-direct mode vs. the memory mode?

How would you design an NVM DB differently?

Before Next Lecture

Submit discussion summary to https://wisc-cs839-ngdb20.hotcrp.com

- **Deadline: Friday 11:59pm**

Submit review for

- Write-behind logging
- [optional] Let's talk about storage & recovery methods for non-volatile memory database systems