# NVM-aware DBMS: A Survey

Ethan Zhu

Department of Computer Science and Technology
Nanjing University, Nanjing, P.R.China

Dec, 22, 2020

## Table of Contents

## Table of Contents

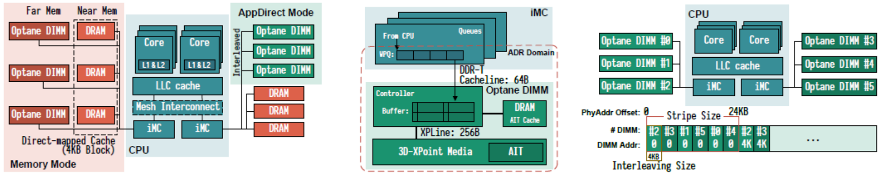
## Table of Contents

## Properties of NVMM

1. An empirical guide to the behavior and use of scalable persistent memory
2. Basic Performance Measurements of the Intel Optane DC Persistent Memory Module
3. A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems
4. A Survey of Non-Volatile Main Memory Technologies: State-of-the-Arts, Practices, and Future Directions

## Properties of Optane DIMM [1] [2]

- The behavior of NVDIMM of past and nowadays
  - ✓ It was broadly similar to DRAM-based DIMM.
  - ✓ But, it has high latency and low bandwidth.
  - ✓ So, "slower, persistent DRAM" is not correct.
  - ✓ Optane DIMM performance is dependent on the access size, access type, pattern, and degree of concurrency than DRAM performance.

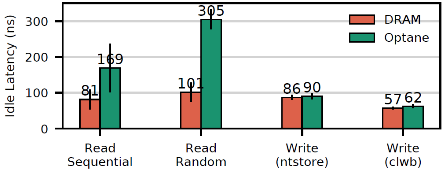


(a) Optane Platform Modes (Memory and AppDirect) (b) Optane DIMM Overview (c) Interleaved Optane DIMMs

Figure 1: **Overview of (a) Optane platform, (b) Optane DIMM and (c) how Optane memories interleave across channels** Optane DIMM can work as either volatile far memory with DRAM as cache (memory mode), or persistent memory with DRAM as main memory (AppDirect mode).

[1] An Empirical Guide to the Behavior and Use of Scalable Persistent Memory
[2] https://zhuanlan.zhihu.com/p/229211653

## Read and Write Latency I

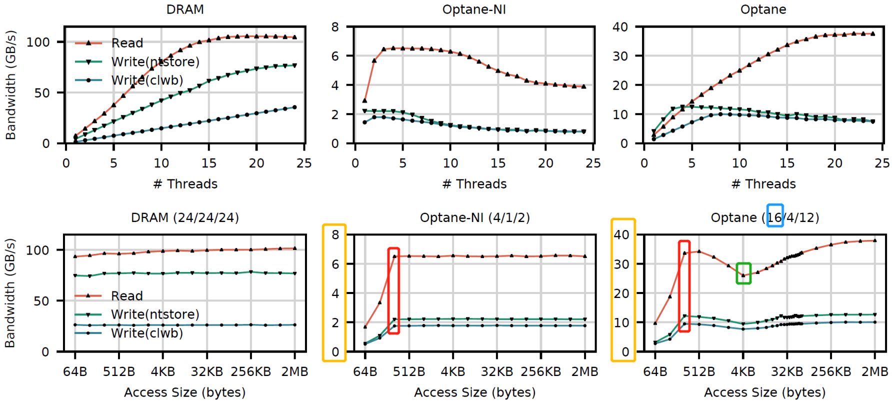


- The read latency for optane is 2×-3× higher than DRAM.
  - ✓ This difference is due to Optane's longer media latency. And Optane memory is also pattern-dependent than DRAM.
- The random-vs-sequential gap is 20% for DRAM but 80% for Optane memory.
- Optane memory is similar to DRAM at store latency.
  - The memory store and fence instructions commit once the data reachs the ADR at the iMC.

## Read and Write Latency II

- Non-temporal stores (ntstore) are more expensive than writes with cache flushes (clwb).
- In general, the latency variance for Optane is extremely small, save for an extremely small number of ”outliers”.
- The sequential read latencies for Optane DIMMs have higher variances, as the first cache line access loads the entire XPLine into XPBuffer, and the following three accesses read data in the buffer.

## Bandwidth I



- Optane bandwidth for random accesses under 256B (read/write units' size) is poor.

## Bandwidth II

- However, DRAM bandwidth does not exhibit a similar "knee" at 8kB.
  - ✓ Because the cost of opening a page of DRAM is much lower than accessing a new page of Optane.
- Interleaving improves peak read and write bandwidth by 5.8× and 5.6×.
  - The speedups match the number of DIMMs in the system and highlight the per-DIMM bandwidth limitations of Optane.
- The most striking feature of the graph is a dip in performance at 4 kB —this dip is an emergent effect caused by contention at the iMC, and it is maximized when threads perform random accesses close to the interleaving size.

## Best Practise for Optane DIMMs

- Avoid random accesses smaller than 256 B.
  - ✓ Avoid small stores, but if that is not possible, limit the working set to 16 kB per Optane DIMM.
- Use non-temporal stores when possible for large transfers, and control cache evictions.
- Limit the number of concurrent threads accessing an Optane DIMM.
- Avoid NUMA accesses (especially read-modify-write sequences).

## Table of Contents

## The Challenges to manage PM Efficiently

1. Persistent memory is widely managed in the form of file systems.
   - As the byte-addressable NVMMs offer much better random access performance than traditional blocks devices.
   - The performace bottleneck of PM-based file systems have shifted from the hardware to the systems software stack.
   - It is essential to shorten the data path in the software stack.
2. Many CPUs use write-back cache to achieve high performance for write operations.
   - The last level cache (LLC) may change the order of data written back to PM.
   - In case of power failure or system crash, it may cause a data inconsistency problem.
   - So, to guarantee data consistency in PM, the order of write operations and a write atomicity model are required.
3. Persistent objects and data structures are more promising to PM programming compared to PM-based file systems。
   - However, theose persistent objects and data structures still face the challenges of guaranteeing data consistency.

## Write Order Guarantee

1. Hardware Primitives
   - NV-tree: A consistent and workload-adaptive tree structure for non-volatile memory
   - Bztree: A high-performance latch-free range index for non-volatile memory
   - Intel architecture instruction set extensions programing reference [3] [4]
2. Write-through Cache
   - Mnemosyne: Lightweight persistent memory
   - Architecture exploration for ambient energy harvesting nonvolatile processors
3. Persistent Cache
   - Kiln: Closing the performance gap between systems with and without persistence support

[3]https://software.intel.com/sites/landingpage/IntrinsicsGuide/

[4]https://software.intel.com/sites/default/files/managed/c5/15/architecture-instruction-set-extensions-programming-reference.pdf

## Atomic Updating

1. System software for persistent memory
2. Better I/O through byte-addressable, persistent memory
3. NOVA: A log-structured file system for hybrid volatile/non-volatile main memories
4. Fault-tolerant precise data access on distributed log-structured merge-tree
5. Atomic persistence for SCM with a non-intrusive backend controller

## Table of Contents

## Table of Contents

## Write Reduction I

1. Data migration
   - RAMZzz: Rank-aware DRAM power management with dynamic migrations and demotions
   - PDRAM: A hybrid PRAM and DRAM main memory system
   - Exploring phase change memory and 3D die-stacking for power/thermal friendly, fast and durable memory architectures
   - CLOCK-DWF: A write-history-aware page replacement algorithm for hybrid PCM and DRAM memory architectures
   - Effcient page caching algorithm with prediction and migration for a hybrid main memory
   - RAMinate: Hypervisor-based virtualization for hybrid main memory systems
2. Caching or Buffering
   - Scalable high performance main memory system using phase-change memory technology

## Write Reduction II

3. Inner-NVM write reduction
   - Flip-N-Write: A simple deterministic technique to improve PRAM write performance, energy and endurance
   - Preventing PCM banks from seizing too much power
   - Kiln: Closing the performance gap between systems with and without persistence support
4. A lazy write mechanism
   - Scalable high performance main memory system using phase-change memory technology
5. Memory compression mechanisms
   - CompEx: Compression-Expansion coding for energy, latency, and lifetime improvements in MLC/TLC NVM
   - Write-aware management of NVM-based memory extensions

## Table of Contents

## Wear Leveling I

1. PDRAM: A hybrid PRAM and DRAM main memory system
2. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling
3. Zombie memory: Extending memory lifetime by reviving dead blocks
4. Dynamically Replicated Memory: Building Reliable Systems from Nanoscale Resistive Memories
5. Optimizing Systems for Byte-Addressable NVM by Reducing Bit Flipping
6. Kevlar-Software Wear Management for Persistent Memories

## Table of Contents

## Table of Contents

## Architecture of NVM-aware's DBMS I

1. Scalable high performance main memory system using phase-change memory technology → DRAM+PCM
2. A Hybrid solid-state storage architecture for the performance, energy consumption, and lifetime improvement → Flash SSD + PCM
3. Evaluating Phase Change Memory for Enterprise Storage Systems → Flash SSD + PCM + DRAM
4. A Prolegomenon on OLTP Database Systems for Non-Volatile Memory
5. Let's Talk About Storage & Recovery Methods for Non-Volatile Memory Database Systems → pure NVM + CPU cache and No DRAM
6. Storage Management in the NVRAM Era → NVM + DRAM

## Architecture of NVM-aware's DBMS II

7. NVRAM-aware Logging in Transaction Systems → logging in NVMM and others in HDD/SSD
8. Scalable Logging through Emerging Non-Volatile Memory → logging in NVMM and others in HDD/SSD
9. Managing Non-Volatile Memory in Database Systems
10. Better IO Through Byte-addressable Persistent Memory → BPFS
11. System Software for Persistent Memory → PMFS
12. NOVA: A Log-structured File System for Hybrid Volatile Non-volatile Main Memory → NOVA
13. The Design and Implementation of a Non-Volatile Memory Database Management System
14. High Performance Multi-core Transaction Processing via Deterministic Execution → phd thesis

## Table of Contents

## Index I

1. A Survey of B-Tree Locking Techniques
2. NV-tree: A Consistent and Workload-adaptive Tree Structure for Non-volatile Memory → optimization of hardware primitives for write order guarantee
3. The Bw-Tree: A B-tree for New Hardware
4. Building A Bw-Tree Takes More Than Just Buzz Words → Bw-Tree
5. BzTree: A High-Performance Latch-free Range Index for Non-Volatile Memory → Bz-tree → optimization of hardware primitives for write order guarantee [5]
6. Evaluating Persistent Memory Range Indexes
7. Dash: Scalable Hashing on Persistent Memory
8. Easy Lock-Free Indexing in Non-Volatile Memory → PMWCAS

## Index II

9. PiBench Online: Interactive Benchmarking of Persistent Memory Indexes
10. Bloom Filter
    - SuRF: Practical Range Query Filtering with Fast Succinct Tries
11. LSM-tree
    - LSM-based storage techniques: a survey $\rightarrow$ VLDBJ
12. Learning Index
    - The Case for Learned Index Structures
    - ALEX: An Updatable Adaptive Learned Index
    - Learning Multi-dimensional Indexes
    - XIndex: A Scalable Learned Index for Multicore Data Storage

[5] https://github.com/EthanZhu-DB/bztree

## Table of Contents

## Persistent Data Structure

1. Don't persist all: Efficient persistent data structures
2. Closing the performance gap between DRAM and PM for in-memory index structures
3. Flatstore: An ecient log-structured key-value storage engine for persistent memory

## Table of Contents

## Crash Recovery for Hybrid Storage DBMS: A Survey I

1. The Recovery Manager of the System R Database Manager
2. ARIES a transaction recovery method supporting fine-granularity locking and partial rollbacks using write-ahead logging
3. Segment-Based Recovery Write-ahead logging revisited
4. From ARIES to MARS Transaction Support for Next-Generation Solid State Drives
5. Fast Databases with Fast Durability and Recovery Through Multicore Parallelism
6. Rethinking Main Memory OLTP Recovery
7. Scalable Logging through Emerging Non-Volatile Memory
8. Low-Overhead Asynchronous Checkpointing in Main-Memory Database Systems
9. Write-Behind Logging

## Crash Recovery for Hybrid Storage DBMS: A Survey II

10. Constant Time Recovery in Azure SQL Database
11. An Empirical Evaluation of In-Memory Multi-Version Concurrency Control
12. Fast Serializable Multi-Version Concurrency Control for Main-Memory Database Systems
13. Scalable Garbage Collection for In-Memory MVCC Systems [6]

[6] https://github.com/EthanZhu-DB/Survey

## Table of Contents

## Cold and Hot data Sepration [7]

1. Reducing the Storage Overhead of Main-Memory OLTP Databases with Hybrid Indexes
2. LeanStore: In-Memory Data Management Beyond Main Memory
3. Larger-than-Memory Data Management on Modern Storage Hardware for In-Memory OLTP Database Systems

[7] https://zhuanlan.zhihu.com/p/88618415

## Table of Contents

## Tuning Parameters

1. Automatic Database Management System Tuning Through Large-scale Machine Learning → OtterTune
2. An End-to-End Automatic Cloud Database Tuning System Using Deep Reinforcement Learning → CBDTune
3. QTune: A QueryAware Database Tuning System with Deep Reinforcement Learning → QTune

## Table of Contents

## Instantaneous Recovery and Intermittent Computation

1. SOFORT: A Hybrid SCM-DRAM Storage Engine for Fast Data Recovery
2. Constant Time Recovery in Azure SQL Database
3. Intermittent Computation without Hardware Support or Programmer Intervention
4. Instant Recovery with Write-Ahead Logging: Page Repair, System Restart, Media Restore, and System Failover

## Table of Contents

## Technology Selection

- Enviroments Setting
- DBMSs testbed and Programming
  - ∗ File System. It goes through the kernel virtual filesystem (VFS) layer.
  - ✓ libpm Library.
- Microbenchmark and Macrobenchmark
- Evaluation Methods
  - ✓ Latency
  - ✓ Bandwidth
  - ✓ Recovery Time
  - ✓ Execution Time Breakdown using Perf [8]

[8]https://github.com/brendangregg/perf-tools

## Table of Contents

Thank you!
Welcome for any questions!

Ethan Zhu
Department of Computer Science and Technology
Nanjing University, Nanjing, P.R.China