Lecture #24 – Non-Volatile Memory Databases
Final Exam: May 4th @ 12:00pm
→ Multiple choice + short-answer questions.
→ I will provide sample questions this week.

Code Review #2: May 4th @ 11:59pm
→ We will use the same group pairings as before.

Final Presentations: May 9th @ 5:30pm
→ WEH Hall 7500
→ 12 minutes per group
→ Food and prizes for everyone!
TODAY’S AGENDA

Background
Storage & Recovery Methods for NVM
NON-VOLATILE MEMORY

Emerging storage technology that provide low latency read/writes like DRAM, but with persistent writes and large capacities like SSDs. → AKA Storage-class Memory, Persistent Memory

First devices will be block-addressable (NVMe)
Later devices will be byte-addressable.
FUNDAMENTAL ELEMENTS OF CIRCUITS

Capacitor (ca. 1745)

Resistor (ca. 1827)

Inductor (ca. 1831)
In 1971, Leon Chua at Berkeley predicted the existence of a fourth fundamental element.

A two-terminal device whose resistance depends on the voltage applied to it, but when that voltage is turned off it permanently remembers its last resistive state.
FUNDAMENTAL ELEMENTS OF CIRCUITS

Capacitor (ca. 1745)

Resistor (ca. 1827)

Inductor (ca. 1831)

Memristor (ca. 1971)
A team at HP Labs led by Stanley Williams stumbled upon a nano-device that had weird properties that they could not understand.

It wasn’t until they found Chua’s 1971 paper that they realized what they had invented.
MERISTORS

A team at HP Labs led by Stanley Williams stumbled upon a nano-device that had weird properties that they could not understand. It wasn’t until they found Chua’s 1971 paper that they realized what they had invented.
A team at HP Labs led by Stanley Williams stumbled upon a nano-device that had weird properties that they could not understand. It wasn't until they found Chua's 1971 paper that they realized what they had invented.
MEMRISTOR – HYSTERESIS LOOP

Vacuum Circuits (ca. 1948)

TWO CENTURIES OF MEMRISTORS
Nature Materials 2012
TECHNOLOGIES

Phase-Change Memory (PRAM)
Resistive RAM (ReRAM)
Magnetoresistive RAM (MRAM)
PHASE-CHANGE MEMORY

Storage cell is comprised of two metal electrodes separated by a resistive heater and the phase change material (chalcogenide).

The value of the cell is changed based on how the material is heated.
→ A short pulse changes the cell to a ‘0’.
→ A long, gradual pulse changes the cell to a ‘1’.
RESISTIVE RAM

Two metal layers with two TiO\textsubscript{2} layers in between. Running a current one direction moves electrons from the top TiO\textsubscript{2} layer to the bottom, thereby changing the resistance.

May be programmable storage fabric… → Bertrand Russell’s Material Implication Logic
MEMRISTOR
NON-VOLATILE STORAGE

A resistor with memory

› 2006: HP Labs proves fourth fundamental element of electronic circuitry

› 2008: Development ready

FUTURE

RESEARCH CONTRIBUTION

› Replace DRAM and hard drives, transistors

Source: Luke Kilpatrick
MAGNETORESISTIVE RAM

 Stores data using magnetic storage elements instead of electric charge or current flows.

Spin-Transfer Torque (STT-MRAM) is the leading technology for this type of NVM.
→ Supposedly able to scale to very small sizes (10nm) and have SRAM latencies.
WHY THIS IS FOR REAL THIS TIME

Industry has agreed to standard technologies and form factors.

Linux and Microsoft have added support for NVM in their kernels (DAX).

Intel has added new instructions for flushing cache lines to NVM.
NVM DIMM FORM FACTORS

NVDIMM-F (2015)
→ Flash only. Has to be paired with DRAM DIMM.

NVDIMM-N (2015)
→ Flash and DRAM together on the same DIMM.
→ Appears as volatile memory to the OS.

NVDIMM-P (2018)
→ True persistent memory. No DRAM or flash.
NVM FOR DATABASE SYSTEMS

Block-addressable NVM is not that interesting.

Byte-addressable NVM will be a game changer but will require some work to use correctly.
→ In-memory DBMSs will be better positioned to use byte-addressable NVM.
→ Disk-oriented DBMSs will initially treat NVM as just a faster SSD.
STORAGE & RECOVERY METHODS

Understand how a DBMS will behave on a system that only has byte-addressable NVM.

Develop NVM-optimized implementations of standard DBMS architectures.

Based on the N-Store prototype DBMS.
Existing programming models assume that any write to memory is non-volatile. → CPU decides when to move data from caches to DRAM.

The DBMS needs a way to ensure that data is flushed from caches to NVM.
If the DBMS process restarts, we need to make sure that all of the pointers for in-memory data point to the same data.
NAMING

If the DBMS process restarts, we need to make sure that all of the pointers for in-memory data point to the same data.
If the DBMS process restarts, we need to make sure that all of the pointers for in-memory data point to the same data.
If the DBMS process restarts, we need to make sure that all of the pointers for in-memory data point to the same data.
Feature #1: Synchronization
→ The allocator writes back CPU cache lines to NVM using the \texttt{CLFLUSH} instruction.
→ It then issues a \texttt{SFENCE} instruction to wait for the data to become durable on NVM.

Feature #2: Naming
→ The allocator ensures that virtual memory addresses assigned to a memory-mapped region never change even after the OS or DBMS restarts.
DBMS ENGINE ARCHITECTURES

Choice #1: In-place Updates
→ Table heap with a write-ahead log + snapshots.
→ Example: VoltDB

Choice #2: Copy-on-Write
→ Create a shadow copy of the table when updated.
→ No write-ahead log.
→ Example: LMDB

Choice #3: Log-structured
→ All writes are appended to log. No table heap.
→ Example: RocksDB
IN-PLACE UPDATES ENGINE

In-Memory Index

In-Memory Table Heap

Durable Storage

Tuple #00

Tuple #01

Tuple #02

Write-Ahead Log

Snapshots
IN-PLACE UPDATES ENGINE

In-Memory Index

In-Memory Table Heap
- Tuple #00
- Tuple #01
- Tuple #02

Durable Storage
- Write-Ahead Log
  - Tuple Delta
- Snapshots
IN-PLACE UPDATES ENGINE

In-Memory Index

In-Memory Table Heap

Tuple #00

Tuple #01 (!)

Tuple #02

Durable Storage

Write-Ahead Log

Tuple Delta

Snapshots
IN-PLACE UPDATES ENGINE

In-Memory Index

In-Memory Table Heap

Tuple #00

Tuple #01 (!)

Tuple #02

Durable Storage

Write-Ahead Log

Tuple Delta

Snapshots

Tuple #01 (!)
IN-PLACE UPDATES ENGINE

- In-Memory Table Heap
- Tuple #00
- Tuple #02
- Durable Storage
- Write-Ahead Log
- Tuple Delta
- In-Memory Index
- Tuple #01
- Snapshots
- Tuple #01 (!)

⚠️ Duplicate Data
⚠️ Recovery Latency
Leverage the allocator’s non-volatile pointers to only record what changed rather than how it changed.

The DBMS only has to maintain a transient UNDO log for a txn until it commits.
→ Dirty cache lines from an uncommitted txn can be flushed by hardware to the memory controller.
→ No REDO log because we flush all the changes to NVM at the time of commit.
NVM IN-PLACE UPDATES ENGINE

NVM Index

NVM Table Heap
- Tuple #00
- Tuple #01
- Tuple #02

NVM Storage
- Write-Ahead Log
NVM IN-PLACE UPDATES ENGINE

NVM Index

NVM Table Heap
- Tuple #00
- Tuple #01
- Tuple #02

NVM Storage
- Write-Ahead Log
  - Tuple Pointers
NVM IN-PLACE UPDATES ENGINE

NVM Index

NVM Table Heap

Tuple #00

Tuple #01 (!)

Tuple #02

NVM Storage

Write-Ahead Log

Tuple Pointers

1

2
COPY-ON-WRITE ENGINE

Master Record

Current Directory

Leaf 1

Leaf 2

Slotted Page #00

Slotted Page #01
COPY-ON-WRITE ENGINE

Master Record

Current Directory

Leaf 1

Leaf 2

Updated Leaf 1

Slotted Page #00

Slotted Page #01

Slotted Page #00
COPY-ON-WRITE ENGINE

Master Record

Current Directory

Dirty Directory

Leaf 1

Leaf 2

Updated Leaf 1

Slotted Page #00

Slotted Page #01

Slotted Page #00
COPY-ON-WRITE ENGINE

Master Record

Current Directory

Dirty Directory

Leaf 1

Leaf 2

Updated Leaf 1

Slotted Page #00

Slotted Page #01

Slotted Page #00
COPY-ON-WRITE ENGINE

Expensive Copies

Carried Directory

Dirty Directory

Leaf 1

Leaf 2

Updated Leaf 1

Slotted Page #00

Slotted Page #01

Slotted Page #00
NVM COPY-ON-WRITE ENGINE

Current Directory

Master Record

Leaf 1
- Tuple #00

Leaf 2
- Tuple #01
NVM COPY-ON-WRITE ENGINE

Master Record

Current Directory

Leaf 1
- Tuple #00

Leaf 2
- Tuple #01

Updated Leaf 1
- Tuple #00 (!)

Only Copy Pointers
NVM COPY-ON-WRITE ENGINE

Master Record

Current Directory

Leaf 1
TUPLE #00

Dirty Directory

Leaf 2
TUPLE #01

Updated Leaf 1
TUPLE #00 (!)

Only Copy Pointers
LOG-STRUCTURED ENGINE

**MemTable**

**SSTable**

*Write-Ahead Log*

*Bloom Filter*
LOG-STRUCTURED ENGINE

MemTable

SSTable

Write-Ahead Log

Tuple Delta

Bloom Filter
LOG-STRUCTURED ENGINE

MemTable

1 Write-Ahead Log
2 Tuple Delta
3 Tuple Data

SSTable

1 Bloom Filter
2 Tuple Delta
3 Tuple Data
LOG-STRUCTURED ENGINE

⚠️ Duplicate Data

⚠️ Compactions

Tuple Data

1 2 3
NVM LOG-STRUCTURED ENGINE

MemTable

SSTable

Write-Ahead Log

Tuple Delta

Bloom Filter

Tuple Delta

Tuple Data
NVM LOG-STRUCTURED ENGINE

MemTable

Write-Ahead Log

Tuple Delta

SSTable

Bloom Filter

Tuple Data

Tuple Delta

1

2

3
NVM LOG-STRUCTURED ENGINE

MemTable

Write-Ahead Log

Tuple Delta 1
SUMMARY

Storage Optimizations
→ Leverage byte-addressability to avoid unnecessary data duplication.

Recovery Optimizations
→ NVM-optimized recovery protocols avoid the overhead of processing a log.
→ Non-volatile data structures ensure consistency.
EVALUATION

N-Store DBMS testbed with pluggable storage manager architecture.
→ H-Store-style concurrency control

Intel Labs NVM Hardware Emulator
→ NVM latency = 2x DRAM latency

Yahoo! Cloud Serving Benchmark
→ 2 million records + 1 million transactions
→ 10% Reads / 90% Writes
→ High-skew setting
RUNTIME PERFORMANCE

YCSB Workload – 10% Reads / 90% Writes
NVRAM – 2x DRAM Latency

Throughput (txn/sec)

- Traditional
- NVM-Optimized

- In-Place
- Copy-on-Write
- Log-Structured
WRITE ENDURANCE

YCSB Workload – 10% Reads / 90% Writes
NVRAM – 2x DRAM Latency

- Traditional
- NVM-Optimized

NVM Stores (M)

- In-Place ↓40%
- Copy-on-Write ↓25%
- Log-Structured ↓20%
RECOVERY LATENCY

Elapsed time to replay log on recovery
NVRAM – 2x DRAM Latency

- Traditional
- NVM-Optimized

No Recovery Needed
PARTING THOUGHTS

Designing for NVM is important
→ Non-volatile data structures provide higher throughput and faster recovery

Byte-addressable NVM is going to be a game changer when it comes out.
Final Exam Review
Marcel Kornacker (Cloudera Impala)