WHAT NON-VOLATILE MEMORY MEANS FOR THE FUTURE OF DATABASE SYSTEMS
2016
The Future
Non-Volatile Memory

- Persistent storage with byte-addressable operations.
- Fast read/write latencies.
- No difference between random vs. sequential access.
What does NVM mean for DBMSs?

- Thinking of NVM as just a faster SSD is not interesting.
- We want to use NVM as permanent storage for the database, but this has major implications.
  - Operating System Support
  - Cloud Provider Provisioning
  - Database Management System Architectures
Existing Systems

NVM-Only Storage

Hybrid DBMS
Chapter I – Existing Systems

- Investigate how existing systems perform with NVM for write-heavy transaction processing (OLTP) workloads.
- Evaluate two types of DBMS architectures.
  - Disk-oriented (MySQL)
  - In-Memory (H-Store)
DISK-ORIENTED

IN-MEMORY

Buffer Pool

Table Heap

Log

Snapshots

Table Heap

Log

Snapshots
Intel Labs NVM Emulator

- Instrumented motherboard that slows down access to the memory controller with tunable latencies.
- Special assembly to emulate upcoming Xeon instructions for flushing cache lines.
Experimental Evaluation

- Compare architectures on Intel Labs NVM emulator.
- Yahoo! Cloud Serving Benchmark:
  - 10 million records (~10GB)
  - 8x database / memory
  - Variable skew
YCSB // Read-Only Workload
2x Latency Relative to DRAM

TXN/SEC vs. SKEW AMOUNT (HIGH → LOW)

- H-Store
- MySQL
YCSB // 50% Reads / 50% Writes Workload
2x Latency Relative to DRAM

TXN/SEC  SKEW AMOUNT (HIGH → LOW)

0  10,000  20,000  30,000  40,000

MySQL

H-Store

8x Latency
LESSONS

1. NVM Latency does not have a large impact.

2. Logging is a major performance bottleneck.

3. Legacy DBMSs are not prepared to run on NVM.
What would Larry Ellison do?
Chapter II – NVM-only Storage

• Evaluate storage and recovery methods for a system that only has NVM.
• Testbed DBMS with a pluggable storage engines.
• We had to build our own NVM-aware memory allocator.
DBMS Architectures

In-Place
- Table Heap
- Log + Snapshots

Copy-on-Write
- Table Heap
- No Logging

Log-Structured
- No Table Heap
- Log-only Storage

Examples:
- In-Place: VoltDB
- Copy-on-Write: LMDB
- Log-Structured: RocksDB
In-Place Engine

UPDATE table SET val=ABC
WHERE id=123

1. Log
2. Table Heap
3. Snapshots

New Tuple
NVM
Delta Record
New Tuple
NVM-Optimized Architectures

- Use non-volatile pointers to only record what changed rather than how it changed.

- Be careful about how & when things get flushed from CPU caches to NVM.
NVM-Aware In-Place Engine

UPDATE table SET val=ABC
WHERE id=123

New Tuple

Tuple Pointers

Table Heap

Log

Log Record

TxnId

Pointer
Evaluation

• Testbed system using the Intel NVM hardware emulator.
• Yahoo! Cloud Serving Benchmark
  — 2 million records + 1 million transactions
  — High-skew setting
YCSB // 10% Reads / 90% Writes Workload
2x Latency Relative to DRAM

In-Place Copy-on-Write Log-Structured

- Traditional
- NVM-Optimized

<table>
<thead>
<tr>
<th></th>
<th>TXN/SEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-Place</td>
<td></td>
</tr>
<tr>
<td>Copy-on-Write</td>
<td></td>
</tr>
<tr>
<td>Log-Structured</td>
<td></td>
</tr>
</tbody>
</table>

↑ 63%
↑ 122%
↑ 50%
YCSB // 10% Reads / 90% Writes Workload
2x Latency Relative to DRAM

NVM STORES (M)

- In-Place: Traditional 40% NVM-Optimized
- Copy-on-Write: Traditional 25% NVM-Optimized
- Log-Structured: Traditional 20% NVM-Optimized
<table>
<thead>
<tr>
<th>Recovery Time (ms)</th>
<th>In-Place</th>
<th>Copy-on-Write</th>
<th>Log-Structured</th>
</tr>
</thead>
<tbody>
<tr>
<td>$10^3$</td>
<td>0.01</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>$10^4$</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$10^5$</td>
<td>1</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

**No Recovery Needed**
LESSONS

1. Using NVM correctly improves throughput & reduces weadown.
2. Avoid block-oriented components.
3. NVM-only systems are 15-20 years away.
What would Nikita Kahn do?
Chapter III – Hybrid DBMS

• Design and build a new in-memory DBMS that will be ready for NVM when it becomes available.

• Hybrid Storage + Hybrid Workloads
  — DRAM + NVM oriented architecture
  — Fast Transactions + Real-time Analytics
Adaptive Storage

**UPDATE**
```
UPDATE myTable
SET A = 123,
B = 456,
C = 789
WHERE D = “xxx”
```

**SELECT**
```
SELECT AVG(B)
FROM myTable
WHERE C < “yyy”
```

**Original Data**

**Adapted Data**

BRIDGING THE ARCHIPELAGO BETWEEN ROW-STORES AND COLUMN-STORES FOR HYBRID WORKLOADS

*SIGMOD 2016*
LESSONS
Peloton

The Self-Driving Database Management System
NVM Ready
Query Compilation
Vectorized Execution
Autonomous
Apache Licensed