Consistent and Durable Data Structures for Non-Volatile Byte-Addressable Memory

Shivaram Venkataraman*, Niraj Tolia‡, Parthasarathy Ranganathan* and Roy H. Campbell†

*HP Labs, Palo Alto, ‡Maginatics, and †University of Illinois, Urbana-Champaign
Non-Volatile Byte-Addressable Memory (NVBM)

Phase Change Memory

Memristor

Memristor
Non-Volatile Byte-Addressable Memory (NVBM)

- Non-Volatile
- 50-150 nanoseconds
- Scalable
- Lower energy

Memristor
Access Times

- Hard Disk Writes – 3 ms
- Write to SLC Flash – 200 μs
- Processor clock cycle – 1ns
- Update DRAM – 55ns
- Access L2 cache – 10ns
Access Times

Hard Disk Writes – 3 ms
Write to SLC Flash – 200 μs
Update DRAM – 55ns
Access L2 cache – 10ns
Processor clock cycle – 1ns
Writtes to PCM / Memristor – 100-150 ns
Data Stores - Disk

Traditional DB

File systems
Data Stores - DRAM

RAMCloud
memcached
Memory-based DB
Commit Log - Disk
Data Stores - NVBM

Single-level store
Challenges

Consistency

Durability
Outline

- Motivation
- Consistent durable data structures
  - Consistent durable B-Tree
  - Tembo – Distributed Data Store Implementation
- Evaluation
Consistent Durable Data Structures

- Versioning for consistency across failures
- Restore to last consistent version on recovery
- Atomic change across versions
- No new processor extensions!
Versioning

- Totally ordered – Increasing natural numbers
- Every update creates a new version
- Last consistent version
  - Stored in a well-known location
  - Used by reader threads and for recovery
### Consistent Durable B-Tree

**Key**
- Key: [start, end)

**B** – Size of a B-Tree node

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>[1,-)</td>
<td>[2,-)</td>
<td>[3,-)</td>
<td>[4,6)</td>
<td>[5,7)</td>
<td>[8,-)</td>
<td>[9,-)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[10,-)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Live entry
- Deleted entry
Lookup

Find key 20 at version 5
Insert / Split

5 [2,-)  20 [3,-)  99 [1,-)  40 [4,-)

5 [2,4)  20 [3,4)  99 [1,4)  5 [4,-)  20 [4,-)  40 [4,-)  99 [4,-)
Garbage Collection
Tembo – Distributed Data Store Implementation

Based on open source key-value store

Widely used in production

In-memory dataset
Tembo – Distributed Data Store Implementation

Key Value Server

- Consistent durable B-Tree
- Single writer, shared reader
- Consistent Hashing
Outline

- Motivation
- Consistent durable data structures
  - Consistent durable B-Tree
  - Tembo – Distributed Data Store Implementation
- Evaluation
# Ease of Integration

<table>
<thead>
<tr>
<th>Description</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original STX B-Tree</td>
<td>2110</td>
</tr>
<tr>
<td>CDDS Modifications</td>
<td>1902 (90%)</td>
</tr>
<tr>
<td>Redis (v2.0.0-rc4)</td>
<td>18539</td>
</tr>
<tr>
<td>Tembo Modifications</td>
<td>321 (1.7%)</td>
</tr>
</tbody>
</table>
Evaluation - Setup

- API Microbenchmarks
  - Compare with Berkeley DB
  - Tembo: Versioning vs. write-ahead logging
- End-to-End Comparison
  - NoSQL systems – Cassandra
  - Yahoo Cloud Serving Benchmark
- 15 node test cluster
  - 13 servers, 2 clients
  - 720 GB RAM, 120 cores
Durability - Logging vs. Versioning

**Throughput (Ops/sec)**
- **256** values:
  - Redis - BTree+Logging
  - Redis - Hashtable+Logging
  - Tembo - CDDS BTree
- **1024** values:
- **4096** values:

**Value size (bytes)**
- 2M insert operations, two client threads
Yahoo Cloud Serving Benchmark

Ops/sec vs. Client Threads

- Tembo
- Cassandra-inmemory
- Cassandra-disk

Comparison:
- 286% increase
- 44% increase
Furthermore

- Algorithms for deletion
- Analysis for space usage and height of B-Tree
- Durability techniques for current processors
Related Work

- Multi-version data structures
  - Used in transaction time databases

- NVBM based systems
  - BPFS – File system (SOSP 2009)
  - NV-Heaps – Transaction Interface (ASPLOS 2011)

- In-memory data stores
  - H-Store – MIT, Brown University, Yale University
  - RAMCloud – Stanford University
Work-in-progress

- Robust reliability testing
- Support for transaction-like operations
- Integration of versioning and wear-leveling
Conclusion

- Changes in storage media
  - Rethink software stack

- Consistent Durable Data Structures
  - Single-level store
  - Durability through versioning
  - Up to 286% faster than memory-backed systems