Silent Data Access Protocol for NVRAM+RDMA Distributed Storage

Qingyue Liu
ql9@rice.edu
ECE Department, Rice University

Peter Varman
pjv@rice.edu
ECE Department, Rice University

May 7, 2020
Background: NVRAM+RDMA Architecture

- Future Distributed Storage Systems: NVRAM + RDMA
  - **NVRAM** is used directly as persistent database or persistent cache
    - Cache-line access
    - Persistent
  - Communication between storage nodes using **RDMA** protocols
    - Bypass TCP/IP stack
    - Microsecond level I/O latency
  - **NVRAM+RDMA**
    - Bypass CPU
Outline

Background

Previous Work

Telepathy
  • RDMA-based Management Structure
  • Telepathy Data Access Protocol

Experiments and Analysis

Conclusion
## Discussion: Data Replication Protocols

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Read Policy</th>
<th>Write Policy</th>
<th>Consistency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous</td>
<td>Read from primary/secondary;</td>
<td>Write initiated at primary</td>
<td>Strong or Eventual</td>
</tr>
<tr>
<td>(e.g. MongoDB [1])</td>
<td></td>
<td></td>
<td>consistency depending on</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>the read protocol</td>
</tr>
<tr>
<td>Two-phase Commit</td>
<td>Read from primary; Write initiated</td>
<td></td>
<td>Strong consistency</td>
</tr>
<tr>
<td>(e.g. Ceph [2])</td>
<td>at primary</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paxos/Raft</td>
<td>Read from primary or contact</td>
<td>Write initiated at primary</td>
<td>External or Snapshot</td>
</tr>
<tr>
<td>(e.g. Cockroach [3],</td>
<td>primary;</td>
<td></td>
<td>consistency</td>
</tr>
<tr>
<td>Spanner [4], Kudu [5])</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quorum</td>
<td>Read from any node; Write initiated</td>
<td></td>
<td>Eventual consistency</td>
</tr>
<tr>
<td>(e.g. Dynamo [6], Cassandra [7])</td>
<td>at any node (quorum rule)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pipeline</td>
<td>Read and write need to contact</td>
<td></td>
<td>Strong consistency</td>
</tr>
<tr>
<td>(e.g. HDFS [8])</td>
<td>name node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Telepathy</td>
<td>Read from any node; Write initiated</td>
<td></td>
<td>Strong Consistency</td>
</tr>
</tbody>
</table>
Previous Work: RDMA in Distributed Storage Systems

• Replace traditional socket-based channel with two-sided RDMA operations
  ➢ Examples: Ceph [2], RDMA-based memcached [9], RDMA-based HDFS [10], FaSST [11] and Hotpot [12]

• Modify the lower-level communication mechanisms and related APIs
  ➢ Examples: FaRM [13], Octopus [14], Derecho [15]
  ➢ Redesign communication channels
    o Use one-sided RDMA pull for reads
    o Use one-sided RDMA push for writes
  ➢ RDMC: An RDMA multicast pattern

• Common issue
  ➢ Data access protocol itself is not changed
  ➢ Benefits only from faster transmission speeds
Overview of Telepathy

• Data access protocol for distributed key-value storage systems in an NVRAM + RDMA cluster

• High-performance read/write protocol
  ➢ Read from any replica
  ➢ Write initiated at any node

• Strong consistency
  ➢ Reads of an object to any replica return the value of the latest write

• Leverage RDMA features for data and control
  ➢ RDMA Atomics for serializing read and write accesses to an object
  ➢ 1-sided silent RDMA Writes and Reads

• Low CPU utilization
Decoupled Communication Channel (DCC)

- DCC is a novel communication channel for use in Telepathy
- NIC card automatically splits different message types at the hardware level
- **Control messages** use RDMA two-sided protocol and are consumed in FCFS order from the receiver’s Control Buffer
- **Data blocks** use RDMA one-sided protocol and are consumed from the receiver’s Data Buffer in an order specified by the sender application
- **Atomic space** is the registered memory region used to arbitrate concurrent updates from remote writers
Remote Bucket Synchronization (RBS) Table

- **Write serialization** and **read consistency** are realized using a Remote Bucket Synchronization Table (RBS Table) in the atomic space region of Telepathy’s registered memory.

- RDMA atomic operation CAS is used to **silently lock** the bucket entry of the inflight update key.
  - The low order bits of each entry hold the coordinator id of the update key.
  - The high-order bits hold some bits of the update key and act as a **Bloom Filter** for detecting conflicting reads.

- Blocked Read Records structure is used when livelock is detected in the default silent-reads fast path *i.e.* if the replica-based read protocol is triggered.
Read Protocol: Replica-based Read

- 3-Step Read Protocol
- Uses RDMA two-sided operations
- Replica nodes wake up for handling the read
- Two situations to use Replica-based Read
  - When the remote address of the data is not cached in the coordinator
  - A fallback path when livestock is detected in the Silent-Read protocol
Read Protocol: Silent Read

- 5-Step Silent Read Protocol
- Only RDMA one-sided semantics are used
- Replica nodes are not interrupted by read
- If strong consistency is not needed, reads can ignore the last version check to get snapshot isolation
Write Protocol: Coordinator Side

- At Coordinator side:
  - RDMA Atomics are used to **silently resolve write conflicts** among multiple coordinators.
  - **Silent data transmission** is separated from control flow.

```
[Diagram showing the Write Protocol:
  1. Initial Phase: CAS
  2. Lock Phase: RC WRITE
  3. Data Transmission Phase: RC SEND
  4. Commit Phase: RC SEND
  5. Release Phase: CAS
]
```

```
Key: K
Data: D_K

1. R = Consistent_Hash(K)
2. B = Hash(K) % Num_Buckets
3. Reserve slot I_r in data buffer from node r \( r \in R \)

Write \( D_K \) to all \( I_r \)

Data Transmission Phase:

\( L_{r,B} = RDMA\_CAS(\text{LSB}_{32}(K) \ll 32 | r, 0) \)

- \( L_{r,B} = 0 \) && Write Persistent
  - Release all \( I_r \) and \( L_{r,B} \)
  - Sleep random time \( t \)

Commit Phase:
Send \{K, I_r\} to all \( r \)

Release Phase:
Wait for \( r \) and release \( I_r \)
```
Write Protocol: Replica Side

At Replica side:

- CPU side will not be interrupted until the commit phase

![Diagram showing the Write Protocol: Replica Side]

- Received Key: $K$
  - Data buffer id: $I_r$

- Lock Phase
  - $B = \text{Hash}(K)\%\text{Num\_Buckets}$

- Data Transmission Phase
  - Store $K$ with $D_K = \text{Data\_Buffer}(I_r)$

- Release Phase
  - Release $L_{r,B}$ through $\text{RDMA\_CAS}$
  - Release linked block read with $L_{r,B}$
  - Trigger release $I_r$ in coordinator

- Send to Coordinator
- Send ERROR to Coordinator when $L_{r,B} = 0$
Experimental Setup

- Telepathy is implemented on a cluster of servers connected through an Infiniband (IB) network.
- The system is deployed on 12 servers in the Chameleon cluster infrastructure [16].
- The configuration of each server is shown as follows:

<table>
<thead>
<tr>
<th>CPU information</th>
<th>Intel(R) Xeon(R) CPU E5-2670 v3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor speed</td>
<td>2.30 GHz</td>
</tr>
<tr>
<td>Number of CPUs</td>
<td>48</td>
</tr>
<tr>
<td>Threads/Core</td>
<td>2</td>
</tr>
<tr>
<td>Memory size</td>
<td>128 GB</td>
</tr>
<tr>
<td>NIC Card</td>
<td>MT27500 Family [ConnectX-3]</td>
</tr>
<tr>
<td>Network</td>
<td>Infiniband</td>
</tr>
</tbody>
</table>

- DRAM is used as our storage backend, due to limitations of our testbed.
- YCSB benchmark is used to evaluate our designs.
Comparison Protocol: 2 Phase Commit (2PC)

- RDMA two-sided operations are used to optimize the conventional 2PC protocol
- **2PC Read:**
  - The coordinator directly sends the key to the primary to obtain the data
  - Primary send back data
- **2PC Write:**
  - The coordinator sends the key-data pair together with the write command to the primary
  - Phase 1: Primary forwards the key-data pair to all replicas
  - Phase 2: After the primary receive replies from all replicas, it sends them commit messages
Bandwidth: Read Protocol

- Replay trace of 1 million pure reads from different coordinators
- Bandwidths of three different read protocols (Silent Read, Replica-based Read, 2PC) are compared

Experiment 1:
- Data nodes: 1
- Replicas: 1
- Coordinators: 1~5

Experiment 2:
- Data nodes: 3
- Replicas: 3
- Coordinators: 9
Bandwidth: Write Protocol

- Replay trace of 1 million pure writes from different coordinators
- Bandwidths of two different write protocols (Telepathy Write, 2PC Write) are compared

- **Experiment 1:**
  - Data nodes: 3
  - Replicas: 3
  - Coordinators: 1~5

- **Experiment 2:**
  - Data nodes: 6
  - Replicas: 3~6
  - Coordinators: 6

- **Experiment 3:**
  - Data nodes: 3~6
  - Replicas: 3
  - Coordinators: 9
Bandwidth: Uniform vs. Skewed Node Access

- **Uniform Distribution**
  - Data nodes have equal probability of being a primary node

- **Skewed Distribution**
  - Primary node are Zipf distributed with exponent set to 4
  - For three data nodes the probabilities for being a primary are 93%, 5.8% and 1.2%

- **Experiments**
  - Data nodes: 3
  - Replicas: 3
  - Coordinators: 9
  - Percentage of reads: 0%, 25%, 50%, 75%, 100%
Latency: Read & Write

- Experiments
  - Data nodes: 3
  - Replicas: 3
  - Coordinators: 9
  - Percentage of reads: 0%, 25%, 50%, 75%, 100%
CPU Efficiency Improved by Telepathy

- Experiments
  - Data nodes: 3
  - Replicas: 3
  - Coordinators: 9
  - Run a CPU-intensive background task on each core of all servers
  - Number of IOs completed with and without the background task for 100% reads and 100% writes
Conclusion

• Telepathy is a novel data replication and access mechanism for RDMA-based distributed KV stores

• Telepathy is a fully distributed mechanism
  - IO writes are handled by any server
  - IO reads are served by any of the replicas

• Strong consistency is guaranteed while providing high IO concurrency

• Hybrid RDMA semantics are used to directly and efficiently transmit data to target servers

• Telepathy can achieve low IO latency and high throughput, with extremely low CPU utilization
Reference


