Write-Optimized and Consistent RDMA-based Non-Volatile Main Memory Systems

Xinxin Liu, Yu Hua, Xuan Li, Qifan Liu

Huazhong University of Science and Technology

ICCD 2021
Background

- Non-Volatile Main Memory (NVMM)
  - Non-volatility, byte-addressability, high density and DRAM-scale latency.

- Remote Direct Memory Access (RDMA)
  - Allow to directly access remote memory via bypassing kernel and zero memory copy.
    - Two-sided RDMA operations (send and recv):
    - One-sided RDMA operations (read, write and atomic):
      - Provide higher bandwidth/lower latency than two-sided one.
      - Do not involve remote CPU.
Background

- Non-Volatile Main Memory (NVMM)
  - Non-volatility, byte-addressability, high density and DRAM-class latency.

- Remote Direct Memory Access (RDMA)
  - Allow to directly access remote memory via bypassing kernel and zero memory copy.
    - Two-sided RDMA operations (send and recv):
    - One-sided RDMA operations (read, write and atomic):
      - Provide higher bandwidth/lower latency than two-sided one.
      - Do not involve remote CPU.

NVMM can be directly accessed through the RDMA network.

RDMA-based NVMM systems become an important research topic.
Challenges

- RDMA NICs fail to guarantee persistence with NVMM.
- Using one-sided RDMA to access remote NVMM needs to address the challenges of guaranteeing Remote Data Atomicity (RDA):
Challenges

- RDMA NICs fail to guarantee persistence with NVMM.

- Using one-sided RDMA to access remote NVMM needs to address the challenges of guaranteeing **Remote Data Atomicity (RDA):**

![Diagram showing data inconsistency between client, server's NIC, and server's NVMM](image)
Challenges

- RDMA NICs fail to guarantee persistence with NVMM.
- Using one-sided RDMA to access remote NVMM needs to address the challenges of guaranteeing **Remote Data Atomicity (RDA):**

![Diagram showing data transfer issues with RDMA](image)
Challenges

- RDMA NICs fail to guarantee persistence with NVMM.
- Using one-sided RDMA to access remote NVMM needs to address the challenges of guaranteeing **Remote Data Atomicity (RDA):**

![Diagram showing the issues with one-sided RDMA and remote data atomicity](image)

- Client
- Server’s NIC
- Server’s NVMM

The server is unaware of the inconsistency due to no CPU involvement.

Inconsistency!
Existing Solutions

Inefficiency due to:

- **High Network Overheads**
  - ✓ Leverage an extra RDMA read after RDMA write(s)

- **High CPU Consumption**
  - ✓ Logging and COW require the remote CPU to control the sequence among operations.

- **Double NVMM Writes**
  - ✓ Consuming the limited NVMM endurance due to first checking the written data in buffers, and then applying them into the destination addresses.
System Design of Erda

Server

Hash (Object Key)

... Metadata A Metadata X ... Hash Table

Metadata A:
Object Key
Head ID
8-byte Atomic Write Region

8-byte Atomic Write Region:
1-bit Indicator
31-bit (New/Old) Offset
31-bit (Old/New) Offset
1-bit Reserved

Head Array:
Head 1
Head 2
...

Data:
Object M
Object N
...

Log Region

Clients

Object: Checksum K-V Pair

Server

Clients

Object: Checksum K-V Pair
System Design of Erda

1~4 the procedure of reading data

1 Hash (object A) RDMA read

2 RDMA write/read object A

3 Verify checksum over object A

4 If the fetched object is non-atomic, read a previous version via RDMA.

Metadata in server:

... Metadata A ... Hash Table

Log region in server:

Head Node

object A object B object C

... object B object A

1~4 the procedure of writing data

1 Send write requests

2 Update the 8-byte atomic write region in metadata A

3 Return the last written address of log

4 RDMA write_with_imm

Before:

Indicator (New) Offset (Version x+1) Offset (Version x) Offset Reserved

After:

Indicator (Old) Offset (Version x+1) Offset (New) Offset (Version x+2) Offset Reserved

Server CPU

Clients

Object: Checksum K-V Pair

Server

Object: Hash (object A)

Clients

Object: Metadata in server: RDMA write/read object A

Server

Object: RDMA write_with_imm

CPU

Object: Indicator (New) Offset (Version x+1) Offset (Version x) Offset Reserved

Object: Head Array:

Head 1 Head 2 ...

Metadata A

Object Key

Hash (Object Key)

Hash Table

Metadata A

Object Key Head ID 8-byte Atomic Write Region

8-byte Atomic Write Region:

1-bit Indicator 31-bit (New/Old) Offset 31-bit (Old/New) Offset 1-bit Reserved

Object: Log Region

object A object B object C

... object B object A

Object: Data:

Head Array: Server

Clients

Object: object N object M

Object: Checksum K-V Pair
Evaluation

The latency of the update-heavy workload.

The latency of the read-mostly workload.

The throughput of the update-heavy workload.

The throughput of the read-mostly workload.

The number of written bytes. \( N \) is the size of one KV pair. \( \text{Size(key)} \) is the key size.
Conclusion

- **Challenges of guaranteeing Remote Data Atomicity (RDA):**
  - High Network Overheads
  - High CPU Consumption
  - Double NVMM Writes

- **Erda:**
  - A write-optimized log-structured NVMM design for Efficient Remote Data Atomicity.
  - Leverage Out-of-Place Updates & CRC Checksum & 8-Byte Atomic Write.

- Compared with state-of-the-art schemes, **Erda reduces NVMM writes** by 50%, significantly **improves throughput** and **decreases latency**.
Thanks! Q&A