Technische
Universität
Braunschweig

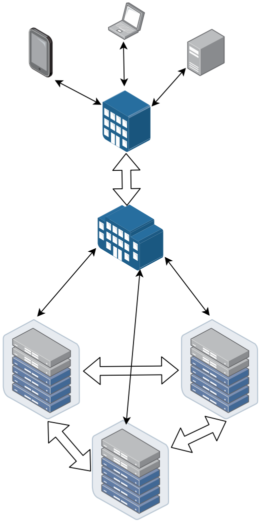
Microsecond Replication for Microsecond Applications

Markus Becker, January 18, 2021

Performance of Distributed Applications

Need for **Microsecond Applications**:

- ultra-low latency
- high throughput
- high availability



Approach

Distribute application with fail-over:

- low-latency
- consistency

Microsecond Applications

Applications

- Distributed Caches
- Key-Value Stores
- Lock-Services
- Financial Trading
- Order Matching

Examples

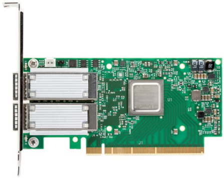
- Redis
- Memcached
- Apache Zookeeper
- Liquibook

Environment

- Datacenters run on commodity hardware
  - Failure prone
- Highly parallel execution
- Large amounts of system memory available
- Specialized high-performance network available
  - Fast switches, links and network cards
  - **Remote Direct Memory Access**

A Promising Networking Technology

Remote Direct Memory Access (RDMA)



Mellanox ConnectX-5
QSFP28x2 NIC
(RDMA capable)

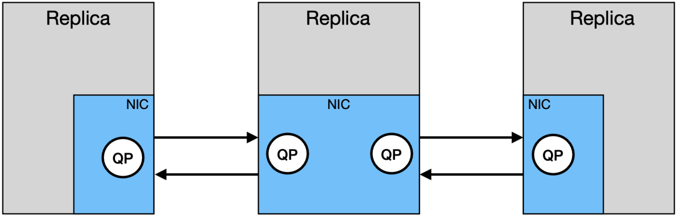
- Read and write directly in remote RAM
- Bypass CPU & Kernel
- Saturate network capabilities
- Access permission system
- Asynchronous
- Latency of 1-3 µs
- Bandwidth of up to 200Gbps

⇒ use RDMA to reduce latency

Contents

- **Motivation**
- **Related Works**
- **Design**
  - Mu
  - Hermes
- **Evaluation**
- **Conclusion**

RDMA Procedural Overview



- Hosts initialize RDMA Connection ⇒ Queue Pair (QP)
  1. Queue: Work Request (WR)
  2. Queue: Work Completion (WC) Events
- App and Network Interface Card (NIC) communicate with QP
  1. Memory areas have to be registered before use
  2. CPU posts WR
  3. NIC performs action and posts WC

Replication Protocols

Consensus Protocols

- Algorithm for sharing state
- Safety & Liveness guarantees

Membership-based Protocol

- Consensus + Replicas + Fail-Over

$\Rightarrow$ Hermes

State Machine Replication (SMR)

- Use consensus protocol to *synchronize* log
  - Safety
  - Liveness
  - Ordering $\Rightarrow$ Strong Consistency
- Apply log as input to deterministic application
- Replicas in identical state

$\Rightarrow$ Mu

Existing RDMA-based Systems

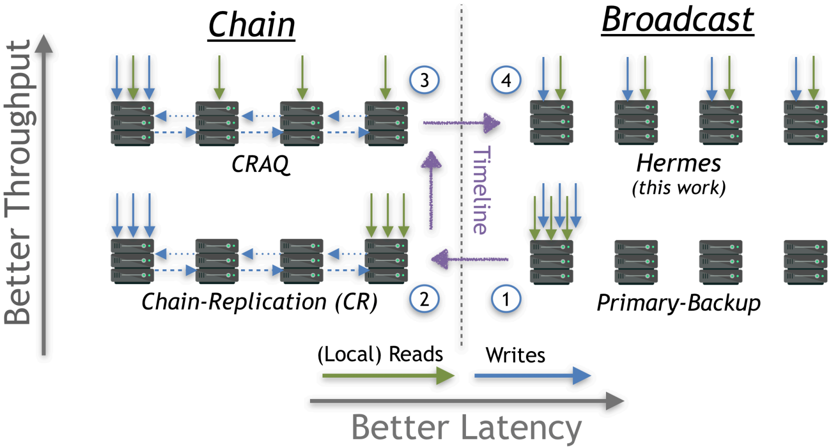
Membership-based Protocol

- ZAB[Junqueira et al., IEEE/IFIP'11]
  - rZAB
- CR / CRAQ[Terrace et al., USENIX'09]
  - rCRAQ

SMR System

- Raft
  - *HovercRaft*[Kogias et al., EuroSys'20]
- Paxos
  - DARE[Poke et al., HPDC'15]
  - APUS[Wang et al., SoCC'17]

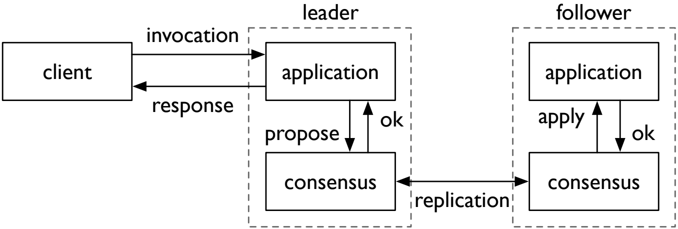
Closer Look at Membership-based Protocols



Design

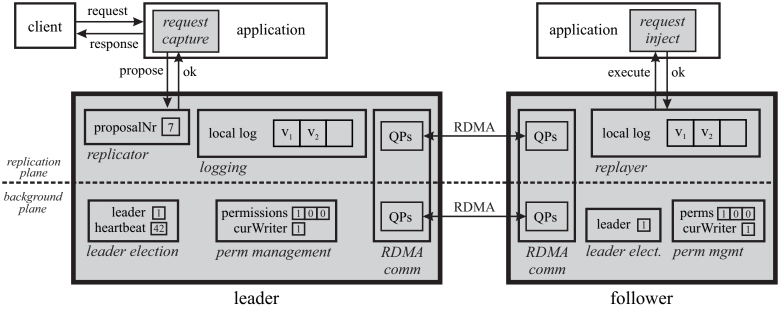
1. Mu: State Machine Replication System
   - Consensus Protocol
   - Strong Consistency
   - Minimizing latency
2. Hermes: Reliable Broadcast Protocol
   - Strong Consistency
   - Cheap to scale

Leader-based Replication



1. Create Replicas (nodes) running State Machines
2. Choose Leader to receive client requests
3. Leader orders and replicates requests to followers
4. Replicas apply requests to own State Machine

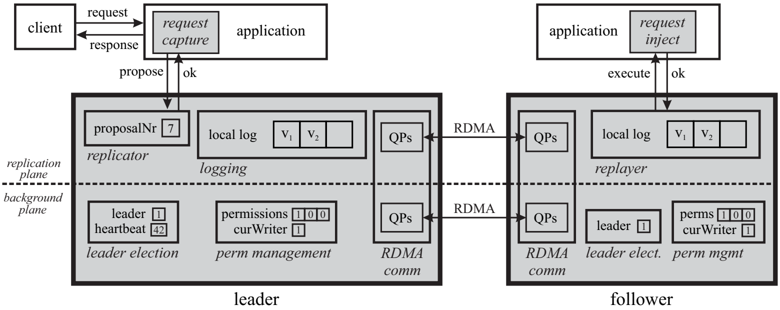
Mu - Application of RDMA



Replicas are connected by two RDMA Connections (Queue Pairs)

1. Leader writes message buffers **directly** into followers logs
2. Followers check heartbeat of leader through **one-sided** reads

Mu - Failure Detection



Largest occurring latency on leader failure ⚡

- Fast permission change critical for low-latency operation!
- If heartbeat stagnates or connection crashes elect new leader
- Followers check heartbeat value of leader **extremely frequently**

New Leader: Follower with lowest Node-ID

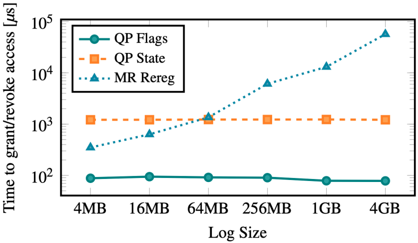
Mu - Permission Switch Survey

On failure

1. Revoke permission from old leader
2. Grant permission to new leader

Permission switch mechanics

- QP Flags:
  Global permission change
- QP State:
  Re-Initialize Queue Pair
- MR Rereg:
  Dereg. and Rereg. buffers



Hermes

Invalidating Replication protocol with datastore application in mind.

- **Membership-based protocol**
  - Acknowledge each write: “read-one/write-all”
- Parallel reads
- **Parallel writes** ↯ bottleneck in ZAB, CRAQ, ...
- **Leaderless** ↯ bottleneck in ZAB, *CRAQ*, ...
- Strong-consistency
  - Safety
  - Liveness
  - Ordering

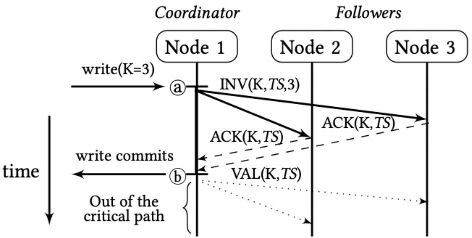
$\Rightarrow$ Completed writes are accessible everywhere

Hermes - Writes

Any replica can initiate write!

1. Become Coordinator for write
2. Send Invalidation containing:
   - Key
   - Value
   - Logical timestamp
3. Collect Acknowledgements
4. Validate after receiving Acknowledgements
5. Swap object value

⇒ On collision: lower node-id wins



Hermes successful write

Hermes - Reads

Invalidated:

Write is currently in progress or has failed.
Attempt to initiate a **write-replay**:

1. Send Invalidation with:
   - Requested key
   - Local object value
   - Timestamp of *failed* write
2. Proceed like a write
3. If **write-replay** succeeds resolve read with local value, else return error

Validated:

Any replica can return the contents without further communication.

$\Downarrow$

For common operation this is default and very fast!

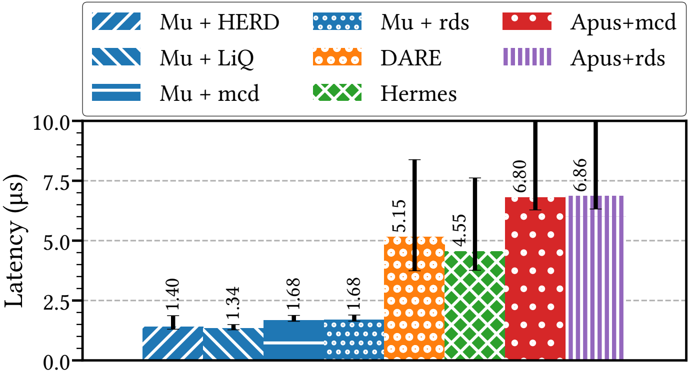
Evaluation

- Mu replicating State Machines on 4-node cluster

CPU	2x Intel Xeon E5-2640 v4 @ 2.40GHz
Memory	2x 128GiB
NIC	Mellanox Connect-X 4
Links	100 Gbps Infiniband
Switch	Mellanox MSB7700 EDR 100 Gbps
Kernel	Ubuntu 18.04.4 LTS 4.15.0-72-generic
RDMA Driver	Mellanox OFED 4.7-3.2.9.0

- Hermes also run on weaker 7-node cluster

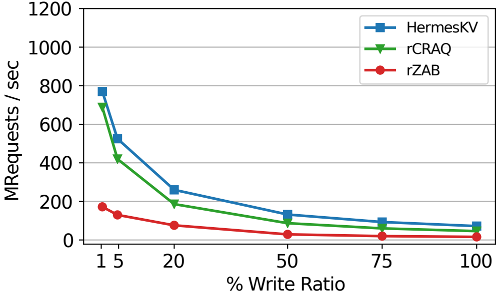
Evaluation: Mu



- Benchmarked replicating:
  - In-memory data-stores: HERD, memcached, redis
  - Order matching service: Liquibook
- Mu achieves best-in-class latency

Evaluation: Hermes

Benchmarks based on specially developed HermesKV (RDMA in-memory distributed key-value store)



- Reads: 2µs – 15µs
- Writes: 29µs – 42µs (3.9× faster than rCRAQ)

Conclusion

- Hermes and Mu provide state-of-the-art replication
- Both require currently datacenter-only hardware
- Slightly different use-case

Mu SMR System

\+ Ultra low-latency

\+ Extremely simple leader-change

\+ Easy to use (plug-and-play)

\- Possibly bottlenecked by leader

Hermes Replication Protocol

\+ Easily load-balanced

\+ Profits from higher parallelism

\- Requires application to be build with protocol in mind

Sources

[1] M. K. Aguilera, N. Ben-David, R. Guerraoui, V. J. Marathe, A. Xygkis, and I. Zablotchi, “Microsecond consensus for microsecond applications,” 2020.

[2] A. Katsarakis, V. Gavrielatos, M. S. Katebzadeh, A. Joshi, A. Dragojevic, B. Grot, and V. Nagarajan, “Hermes: A fast, fault-tolerant and linearizable replication protocol,” *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar 2020. [Online]. Available: http://dx.doi.org/10.1145/3373376.3378496

[3] Mellanox, “Benefits of Remote Direct Memory Access Over Routed Fabrics,” Tech. Rep., 2018. [Online]. Available: https://www.mellanox.com/related-docs/solutions/benefits-of-RDMA-over-routed-fabrics.pdf

[4] C. Wang, J. Jiang, X. Chen, N. Yi, and H. Cui, “Apus: fast and scalable paxos on rdma,” 09 2017, pp. 94–107.

[5] M. Poke and T. Hoefler, “Dare,” *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing - HPDC ’15*, 2015. [Online]. Available: http://dx.doi.org/10.1145/2749246.2749267

[6] L. Lamport, “Paxos made simple,” *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pp. 51–58, December 2001. [Online]. Available: https://www.microsoft.com/en-us/research/publication/paxos-made-simple/