High Throughput Replication with Integrated Membership Management

Pedro Fouto, Nuno Preguiça, João Leitão

USENIX ATC 2022
Outline

• Motivation and Related Work

• ChainPaxos
  o Writing
  o Local Linearizable Reads
  o Reconfiguration

• Evaluation
Outline

• Motivation and Related Work

• ChainPaxos
  o Writing
  o Local Linearizable Reads
  o Reconfiguration

• Evaluation
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
Related Work: Multi-Paxos
Related Work: Multi-Paxos

Leader
Related Work: Multi-Paxos

Classic Multi-Paxos
Related Work: Multi-Paxos

Classic Multi-Paxos

Skipping the first phase
Related Work: Multi-Paxos

Classic Multi-Paxos

accepted
Related Work: Multi-Paxos

Classic Multi-Paxos

- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$
Related Work: Multi-Paxos

**Classic Multi-Paxos**

- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$

**Distinguished Learner**
Related Work: Multi-Paxos

Classic Multi-Paxos

- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$

Distinguished Learner

accept

accepted
Related Work: Multi-Paxos

Classic Multi-Paxos

- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$

Distinguished Learner
Related Work: Multi-Paxos

Classic Multi-Paxos

- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$

Distinguished Learner
Related Work: Multi-Paxos

Classic Multi-Paxos
- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$

Distinguished Learner
- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(1)$
Related Work: Multi-Paxos

Classic Multi-Paxos
- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$

Distinguished Learner
- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(1)$

Message complexity proportional to the number of replicas
Related Work: Multi-Paxos

**Classic Multi-Paxos**
- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(n)$

**Distinguished Learner**
- Message complexity:
  - Leader: $O(n)$
  - Followers: $O(1)$

Message complexity proportional to the number of replicas

Guarantees safety in an asynchronous network
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
Related Work: Chain Replication
Related Work: Chain Replication

Image of a diagram showing a chain replication model with labels 'Head' and 'Tail'.
Related Work: Chain Replication
Related Work: Chain Replication

update
Related Work: Chain Replication
Related Work: Chain Replication
Related Work: Chain Replication
Related Work: Chain Replication

- Message complexity:
  - All replicas: $O(1)$
Related Work: Chain Replication

- Message complexity:
  - All replicas: $O(1)$

However, it has its limitations…
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
- Two aspects are often overlooked:
  - Performant linearizable reads
  - Membership management and reconfiguration
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
- Two aspects are often overlooked:
  - Performant linearizable reads
  - Membership management and reconfiguration
Motivation: Linearizable Reads

- Some existing solutions assume a **synchronous** model (e.g. Chain Replication)

- Dealing with asynchrony is complicated. One can:
  - Relax consistency (e.g. ZooKeeper)
  - Add **extra (costly) steps** to write operations
  - Synchronize with other replicas when reading
Motivation: Linearizable Reads
Motivation: Linearizable Reads
Motivation: Linearizable Reads

response
Motivation: Linearizable Reads

Network Partition
Motivation: Linearizable Reads
Motivation: Linearizable Reads
Motivation: Linearizable Reads
Motivation: Linearizable Reads

read
Motivation: Linearizable Reads
Motivation: Linearizable Reads

Breaks linearizability in an asynchronous network model
Motivation: Linearizable Reads

- Some existing solutions assume a *synchronous* model

- Dealing with asynchrony is complicated. One can:
  - Relax consistency (e.g. ZooKeeper)
  - Add extra *(costly)* steps to write operations
  - Synchronize with other replicas when reading
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
- Two aspects are often overlooked:
  - Performant linearizable reads
  - Membership management and reconfiguration
Motivation: Membership Management

- Most consensus solutions overlook membership management.
- Often assume that using an external coordinator service (e.g. ZooKeeper) is trivial and the best solution.
Motivation: Membership Management

- Most consensus solutions overlook membership management
- Often assume that using an external coordinator service (e.g. ZooKeeper) is trivial and the best solution
- This is not the case:
Motivation: Membership Management

- Most consensus solutions overlook membership management
- Often assume that using an external coordinator service (e.g. ZooKeeper) is trivial and the best solution
- This is not the case:
  - Fault-tolerance becomes complex
Motivation: Membership Management

- Your consensus solution:
  - Fault-tolerance: 2
Motivation: Membership Management

- Your consensus solution:
  - Fault-tolerance: 2
- External Coordinator (e.g. ZooKeeper)
  - Fault-tolerance: 1
Motivation: Membership Management

- Your consensus solution:
  - Fault-tolerance: 2

- External Coordinator (e.g. ZooKeeper)
  - Fault-tolerance: 2
Motivation: Membership Management

- Your consensus solution:
  - Fault-tolerance: 2

- External Coordinator (e.g. ZooKeeper)
  - Fault-tolerance: 2

Replicas: 10
Fault-tolerance: 2
Motivation: Membership Management

- Most consensus solutions overlook membership management

- Often assume that using an external coordinator service (e.g. ZooKeeper) is trivial and the best solution

- This is not the case:
  - **Fault-tolerance** becomes complex
Motivation: Membership Management

- Most consensus solutions overlook membership management
- Often assume that using an external coordinator service (e.g. ZooKeeper) is trivial and the best solution
- This is not the case:
  - Fault-tolerance becomes complex
  - Complex (and redundant) integration with consensus
Motivation: Membership Management
Motivation: Membership Management
Motivation: Membership Management

Consensus round to decide removal
Motivation: Membership Management

Asynchronous decision propagation to consensus replicas
Motivation: Membership Management

Replicas cannot change view individually!
Motivation: Membership Management

Another (redundant) consensus round is required
Motivation: Membership Management

- Most consensus solutions overlook membership management
- Often assume that using an external coordinator service (e.g. ZooKeeper) is trivial and the best solution
- This is not the case:
  - **Fault-tolerance** becomes complex
  - **Complex (and redundant) integration** with consensus
Motivation: Membership Management

- Most consensus solutions overlook membership management

- Often assume that using an external coordinator service (e.g. ZooKeeper) is trivial and the best solution

- This is not the case:
  - Fault-tolerance becomes complex
  - Complex (and redundant) integration with consensus
  - More vulnerable to partial network partitions¹

Motivation: Membership Management

Partition between coordinator and consensus replicas
Motivation: Membership Management

Correct replicas will be removed
Motivation: Membership Management

Partition between consensus replicas
Motivation: Membership Management

System will halt until partition is healed
Motivation: Consensus and SMR

- Building blocks of numerous practical replication systems
- Their performance is critical
- Many alternatives have been designed
- Two very relevant ones:
  - (Multi-)Paxos
  - Chain Replication
- Two aspects are often overlooked:
  - Performant linearizable reads
  - Membership management and reconfiguration
Proposal: ChainPaxos

Novel consensus algorithm:

- **Combining** the best properties of Multi-Paxos and Chain Replication
  - Correction in an *asynchronous network*
  - *Constant* message *complexity*
Proposal: ChainPaxos

Novel consensus algorithm:

• **Combining** the best properties of Multi-Paxos and Chain Replication
  o Correction in an **asynchronous network**
  o **Constant** message **complexity**

• Going beyond existing solutions:
  o **Maximizing throughput** of both read and write operations
  o Providing **local linearizable reads** in any replica
  o **Integrated reconfiguration** and fault-tolerance
Outline

• Motivation and Related Work

• ChainPaxos
  o Writing
  o Local Linearizable Reads
  o Reconfiguration

• Evaluation
Outline

• Motivation and Related Work

• ChainPaxos
  o Writing (commits + garbage collection)
  o Local Linearizable Reads
  o Reconfiguration

• Evaluation
ChainPaxos: Write Path
ChainPaxos: Write Path

Leader (regular Multi-Paxos election)
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path

Encapsulate multiple Multi-Paxos messages

accept
+ accept ok
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path

Operation is “locked-in”
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path

Need to garbage collect + execute on the first replicas
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path
ChainPaxos: Write Path
Outline

- Motivation and Related Work
- ChainPaxos
  - Writing
  - Local Linearizable Reads
  - Reconfiguration
- Evaluation
Local Linearizable Reads

Requirements for linearizability:

- The result of a read must contain **all writes that completed** before it started.
- The result of a read must contain the result of **all reads that completed** before it started.
Local Linearizable Reads

Requirements for linearizability:

• The result of a read must contain all writes that completed before it started
• The result of a read must contain the result of all reads that completed before it started

Challenge:

• Read from any replica
• No extra communication steps
Local Linearizable Reads

Requirements for linearizability:

- The result of a read must contain all writes that completed before it started
- The result of a read must contain the result of all reads that completed before it started

Challenge:

- Read from any replica
- No extra communication steps

The chain topology is our friend!
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads

Response must:
- Contain all completed writes
- Contain all completed reads
Local Linearizable Reads

Response must:
- Contain all completed writes
- Contain all completed reads
Local Linearizable Reads

Response must:
- Contain all completed writes (■)
- Contain all completed reads

Pedro Fouto
USENIX ATC - July 2022
Local Linearizable Reads

Response must:
- Contain all completed writes (■)
- Contain all completed reads
Local Linearizable Reads

Attach read to next unseen operation (green)
Local Linearizable Reads
Local Linearizable Reads

Read is now attached to green
Local Linearizable Reads

We now wait until the green ack goes around the chain

Read is now attached to green
Local Linearizable Reads

We now wait until the green ack goes around the chain

Read is now attached to green
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads

Response must:
- Contain all completed writes (☐)
- Contain all completed reads

Acknowledge of green will unlock read
Local Linearizable Reads

Response must:
- Contain all completed writes (■)
- Contain all completed reads

Respond with current state
Local Linearizable Reads

Stronger than strictly required (✓)
Best we can do without extra communication

Response must:
• Contain all completed writes (✓)
• Contain all completed reads

Respond with current state
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads

Response must:
- Contain all completed writes
- Contain all completed reads (✓)
Local Linearizable Reads

Response must:
- Contain all completed writes
- Contain all completed reads (✓)
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads

Red operation will “push” every pending operation and ack
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads
Local Linearizable Reads

Response must:
- Contain all completed writes
- Contain all completed reads (☐)
Local Linearizable Reads

Response must:
- Contain all completed writes
- Contain all completed reads (■)
Local Linearizable Reads: Summary

- Read is dilated to guarantee linearizability:
  - Ensures all previously **completed reads and writes** are visible

- **No additional communication** steps are required
  - More **conservative** than required, but **unavoidable without coordination**

- Only possible due to **chain topology**
Outline

• Motivation and Related Work

• ChainPaxos
  o Writing
  o Local Linearizable Reads
  o Reconfiguration

• Evaluation
Membership Management: Removal
Membership Management: Removal

Replicas monitor the next replica in the chain
Membership Management: Removal

Pedro Fouto
Membership Management: Removal
Membership Management: Removal

Removal requests are handled like regular operations
Membership Management: Removal
Membership Management: Removal

Skip suspected replica
Membership Management: Removal

Make sure operations are still ordered
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
Membership Management: Removal
ChainPaxos: Summary

- Aggregates **Multi-Paxos messages for correction**
- **Minimizes communication cost** for write operations
- Provides **local linearizable reads** in any replica
  - With no additional communication
- **Integrated reconfiguration** and fault-tolerance
  - Avoiding external coordination services
Outline

- Motivation and Related Work
- ChainPaxos
  - Writing
  - Local Linearizable Reads
  - Reconfiguration
- Evaluation
Evaluation: Goals

- How does ChainPaxos' **performance** compare against the state-of-the-art?
- What is the **latency overhead** of the chain?
- How much do **local reads** improve on the performance?
- Is ChainPaxos adequate to be used in a **practical setting**?
Evaluation: Setup

Compare with state-of-the-art
Evaluation: Setup

Compare with state-of-the-art

- Implemented a replicated key-value store

- Compared ChainPaxos against:
  - MultiPaxos (multiple variants)
  - Chain Replication
  - EPaxos (with and without conflicts)
  - (U-)RingPaxos
Evaluation: Setup

Compare with state-of-the-art

- Implemented a replicated key-value store

- Compared ChainPaxos against:
  - MultiPaxos (multiple variants)
  - Chain Replication
  - EPaxos (with and without conflicts)
  - (U-)RingPaxos

Evaluate a more realistic scenario
Evaluation: Setup

Compare with state-of-the-art

- Implemented a replicated key-value store
- Compared ChainPaxos against:
  - MultiPaxos (multiple variants)
  - Chain Replication
  - EPaxos (with and without conflicts)
  - (U-)RingPaxos

Evaluate a more realistic scenario

- Integrated ChainPaxos in Zookeeper
- Replaced ZAB (ZooKeeper’s atomic broadcast) with ChainPaxos
Evaluation: Methodology

Using Grid5000 testbed

Emulating clients with YCSB
Evaluation: Methodology

Using Grid5000 testbed

Emulating clients with YCSB

Measured:
- Throughput (operations per second)
- Latency (as perceived by clients)
Evaluation: Methodology

Using Grid5000 testbed

Emulating clients with YCSB

Measured:
- Throughput (operations per second)
- Latency (as perceived by clients)

Varying:
- Number of consensus replicas (3, 5, 7)
- Load on the system (YCSB clients)
- Workload (read/write ratio)
Evaluation: Setup

Compare with state-of-the-art

- Implemented a replicated key-value store

- Compared ChainPaxos against:
  - MultiPaxos (multiple variants)
  - Chain Replication
  - EPaxos (with and without conflicts)
  - (U-)RingPaxos

Evaluate a more realistic scenario

- Integrated ChainPaxos in Zookeeper
- Replaced ZAB (ZooKeeper’s atomic broadcast) with ChainPaxos
Evaluation: Results

How does ChainPaxos’ performance compare against the state-of-the-art?
Evaluation: Results

*How does ChainPaxos’ performance compare against the state-of-the-art?*
Evaluation: Results

How does ChainPaxos’ performance compare against the state-of-the-art?

Minimizing the number of messages maximizes throughput.
Evaluation: Results

What is the *latency overhead* of the chain?
Evaluation: Results

What is the latency overhead of the chain?
Evaluation: Results

*What is the latency overhead of the chain?*

Latency is lower with a small number of replicas.
What is the **latency overhead** of the chain?

Latency is lower with a small number of replicas.

Only starts being a problem with >7 replicas.
Evaluation: Results

*How much do local reads improve on the performance?*
Evaluation: Results

*How much do local reads improve on the performance?*

![Graph showing throughput vs average latency for different read strategies.](image)
Evaluation: Results

How much do **local reads** improve on the performance?

Reading through the chain is slow
Evaluation: Results

How much do local reads improve on the performance?

Reading through the chain is slow

Performance increases drastically with % of reads
Evaluation: Results

How much do local reads improve on the performance?

Reading through the chain is slow

Performance increases drastically with % of reads
Evaluation: Results

How much do local reads improve on the performance?

- Reading through the chain is slow
- Performance increases drastically with % of reads
- Latency overhead is minimal

Chart showing average latency (ms) vs throughput (1000 ops/s) for different read percentages and EPaxos-NoDeps 100%.
Evaluation: Results

How much do local reads improve on the performance?

Performance increases with number of replicas

~1300 with 3 replicas
Evaluation: Results

How much do local reads improve on the performance?

Performance increases with number of replicas
~1300 with 3 replicas
~1600 with 7 replicas
Evaluation: Setup

Compare with state-of-the-art

- Implemented a replicated key-value store
- Compared ChainPaxos against:
  - MultiPaxos (multiple variants)
  - Chain Replication
  - EPaxos (with and without conflicts)
  - (U-)RingPaxos

Evaluate a more realistic scenario

- Integrated ChainPaxos in Zookeeper
- Replaced ZAB (ZooKeeper’s atomic broadcast) with ChainPaxos
Evaluation: Results

Is ChainPaxos adequate to be used in a practical setting?
Evaluation: Results

Is ChainPaxos adequate to be used in a practical setting?

![Graph showing evaluation results for different throughput rates and latency times for Zk-Zab and Zk-Chain configurations.](image-url)
Evaluation: Results

Is ChainPaxos adequate to be \textit{used in a practical setting}? 

ChainPaxos shows higher throughput
Evaluation: Results

Is ChainPaxos adequate to be used in a practical setting?

ChainPaxos shows higher throughput

Zab’s throughput decreases with number of replicas
Evaluation: Results

Is ChainPaxos adequate to be used in a practical setting?
Evaluation: Results

Is ChainPaxos adequate to be used in a practical setting?

ChainPaxos’ linearizable reads show better performance
Evaluation: Results

Is ChainPaxos adequate to be used in a practical setting?

- ChainPaxos’ linearizable reads show better performance
- Nearly match performance of weakly-consistent Zab reads
Is ChainPaxos adequate to be used in a practical setting?

ChainPaxos’ linearizable reads show better performance
Nearly match performance of weakly-consistent Zab reads
With minimal latency overhead

Evaluation: Results
Recap: ChainPaxos

Novel consensus algorithm:

- **Combining** the best properties of Multi-Paxos and Chain Replication
  - Correction in an **asynchronous network**
  - **Constant** message complexity

- Going beyond existing solutions:
  - **Maximizing throughput** of both read and write operations
  - Providing **local linearizable reads** in any replica
  - **Integrated reconfiguration** and fault-tolerance
High Throughput Replication with Integrated Membership Management

Pedro Fouto, Nuno Preguiça, João Leitão

USENIX ATC 2022