Unicast Multi-Ring Paxos

Implementation & Evaluation

Samuel Benz

USI

June 2013

Table of Contents

Distributed systems

Problem

1. Scalability:
   - Size: Internet scale services
   - Location: Access latency
2. Fault-Tolerance

Solution

1. Distributed Data: Replication
2. Distributed Computing: Coordination

Consensus and Atomic Broadcast

In a crash-stop failure model **consensus** is defined as follows:

1. **Termination:** Every correct process eventually decides.
2. **Agreement:** No two correct processes decide differently.
3. **Uniform integrity:** Every process decides at most once.
4. **Uniform validity:** If a process decides v, then v was proposed by some process.

Additionally **atomic broadcast**:

5. **Total order:** If two correct processes p and q deliver two messages m and m', then p delivers m before m' if and only if q delivers m before m'.

[Chandra *et al.* Unreliable failure detectors for reliable distributed systems. 1996.]

Consensus and Atomic Broadcast

In a crash-stop failure model **consensus** is defined as follows:

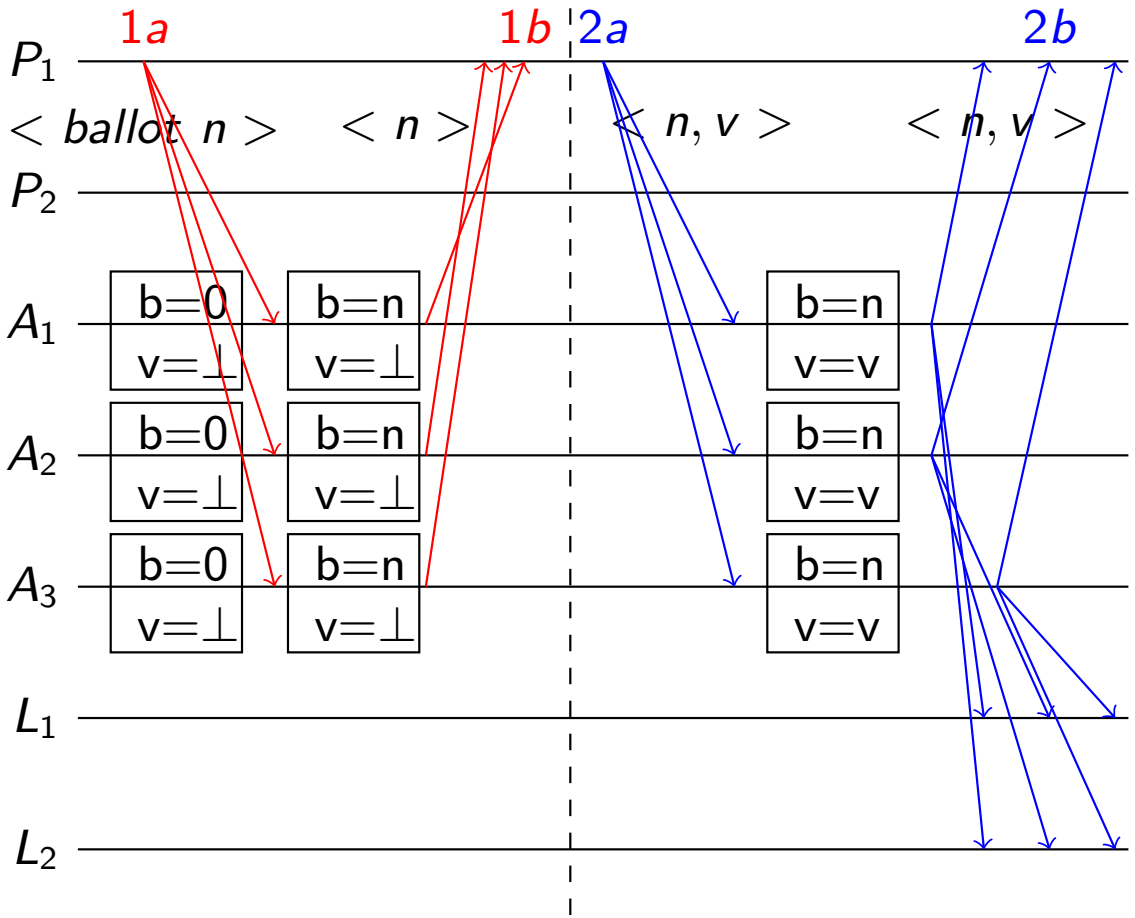
1. **Termination:** Every correct process eventually decides.
2. **Agreement:** No two correct processes decide differently.
3. **Uniform integrity:** Every process decides at most once.
4. **Uniform validity:** If a process decides v, then v was proposed by some process.

Additionally **atomic broadcast**:

5. **Total order:** If two correct processes p and q deliver two messages m and m', then p delivers m before m' if and only if q delivers m before m'.

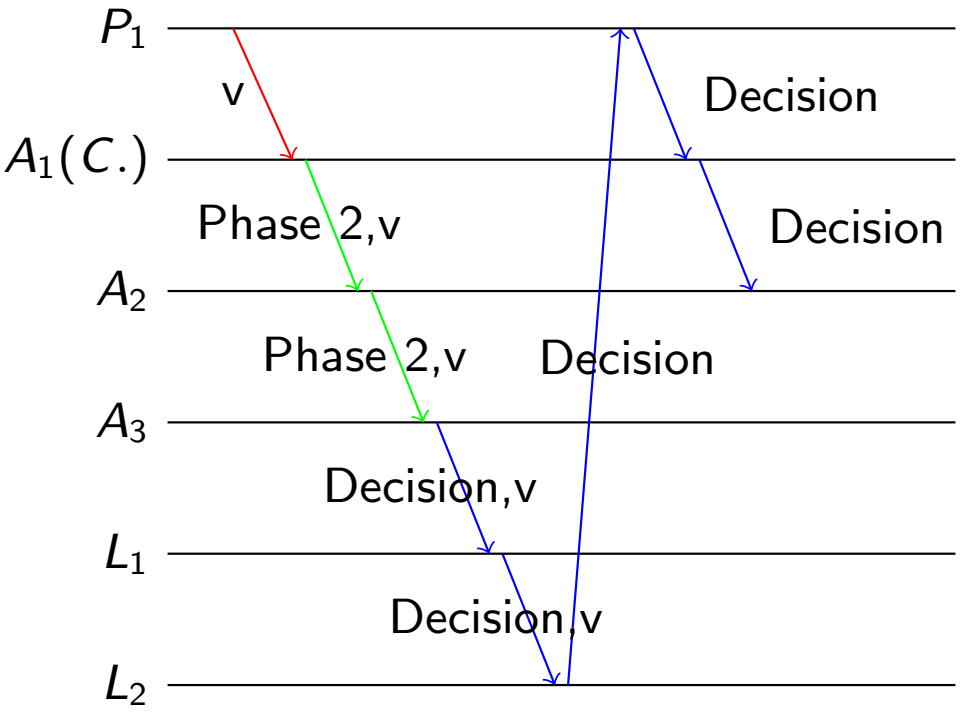
[Chandra *et al.* Unreliable failure detectors for reliable distributed systems. 1996.]

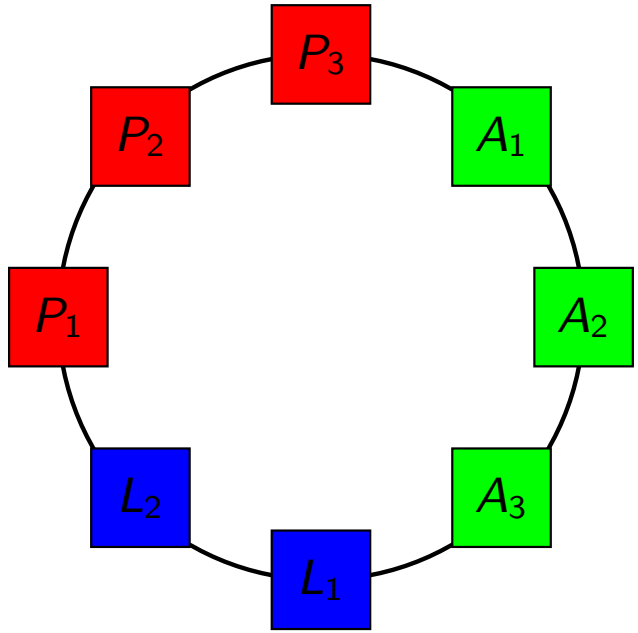
Paxos



[Lamport. The part-time parliament. 1998.]

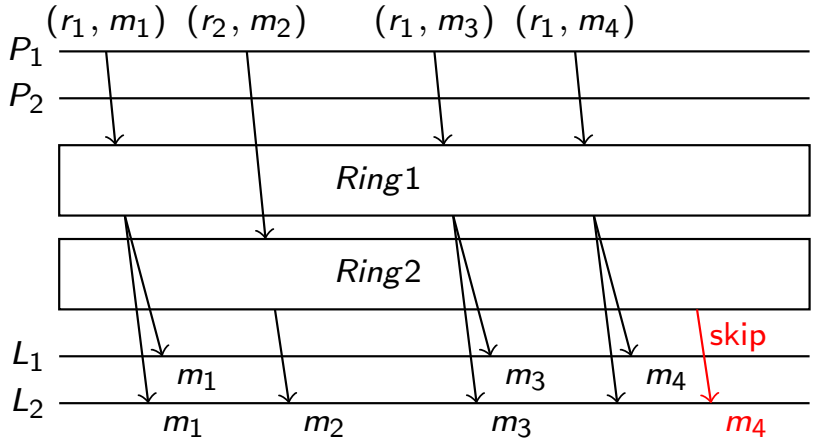
Ring Paxos

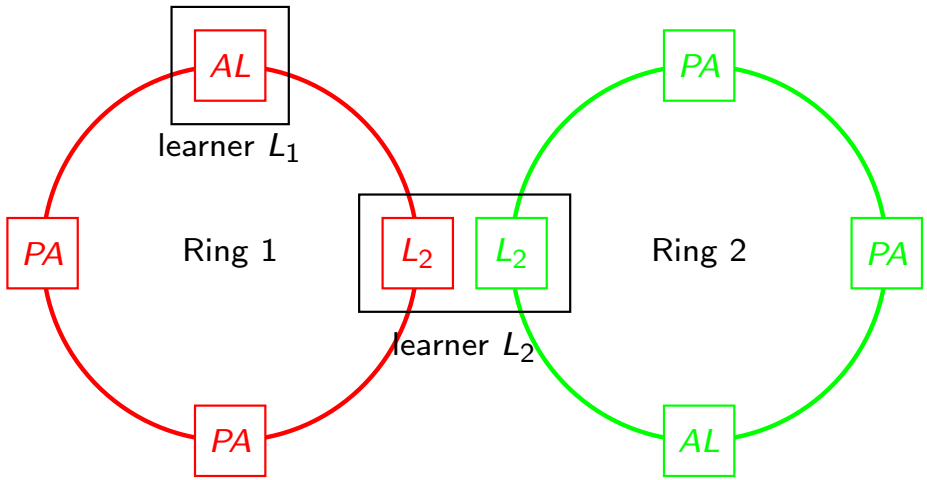




[Marandi *et al.* Ring paxos: A high-throughput atomic broadcast protocol. 2010.]

Multi-Ring Paxos





[Marandi *et al.* Multi-ring paxos. 2012.]

Table of Contents

Implementation

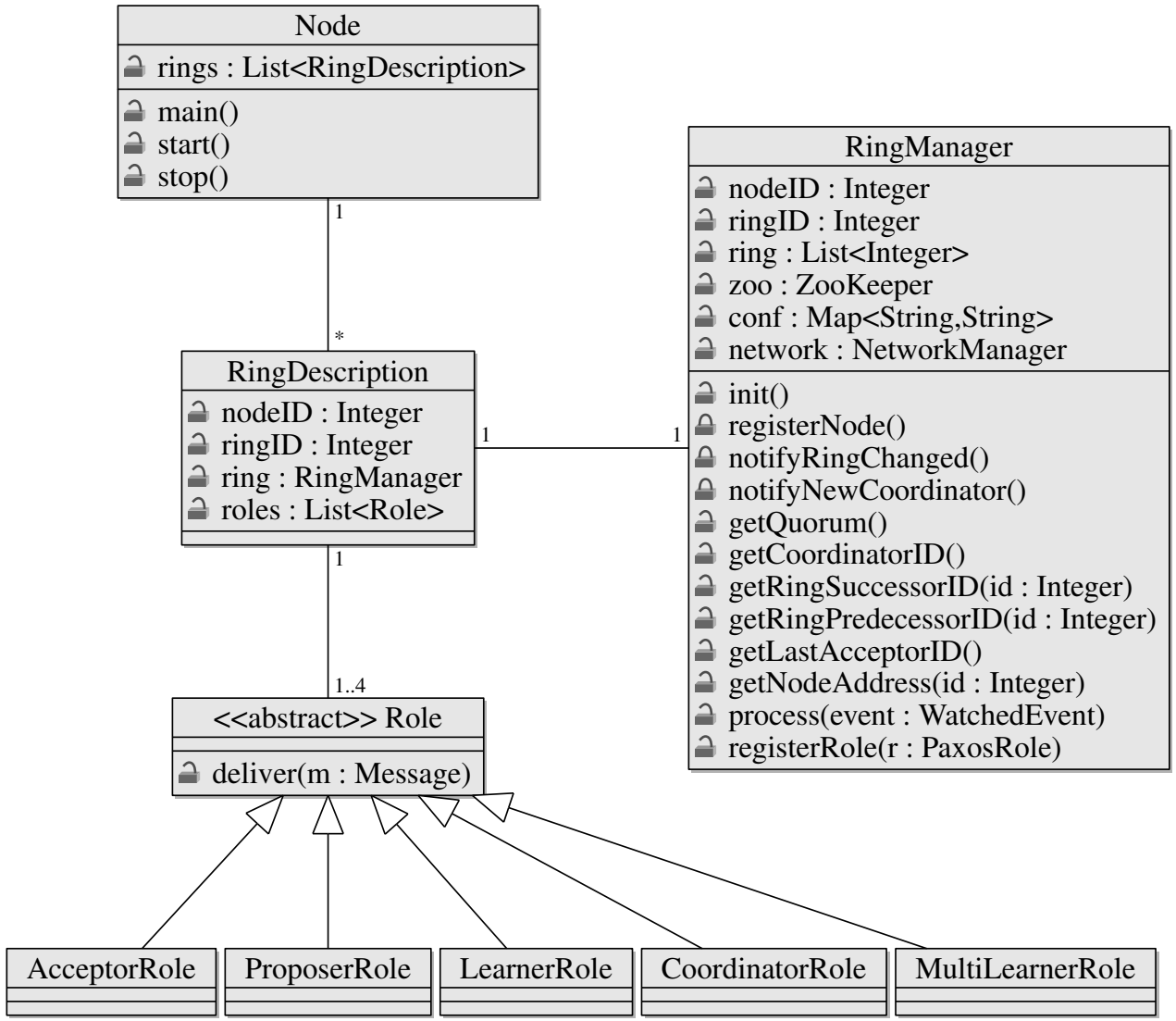
Java: 5030 lines

1. Good code readability, maintainability
2. Comprehensive collection and concurrency APIs
3. Portable
4. Fast

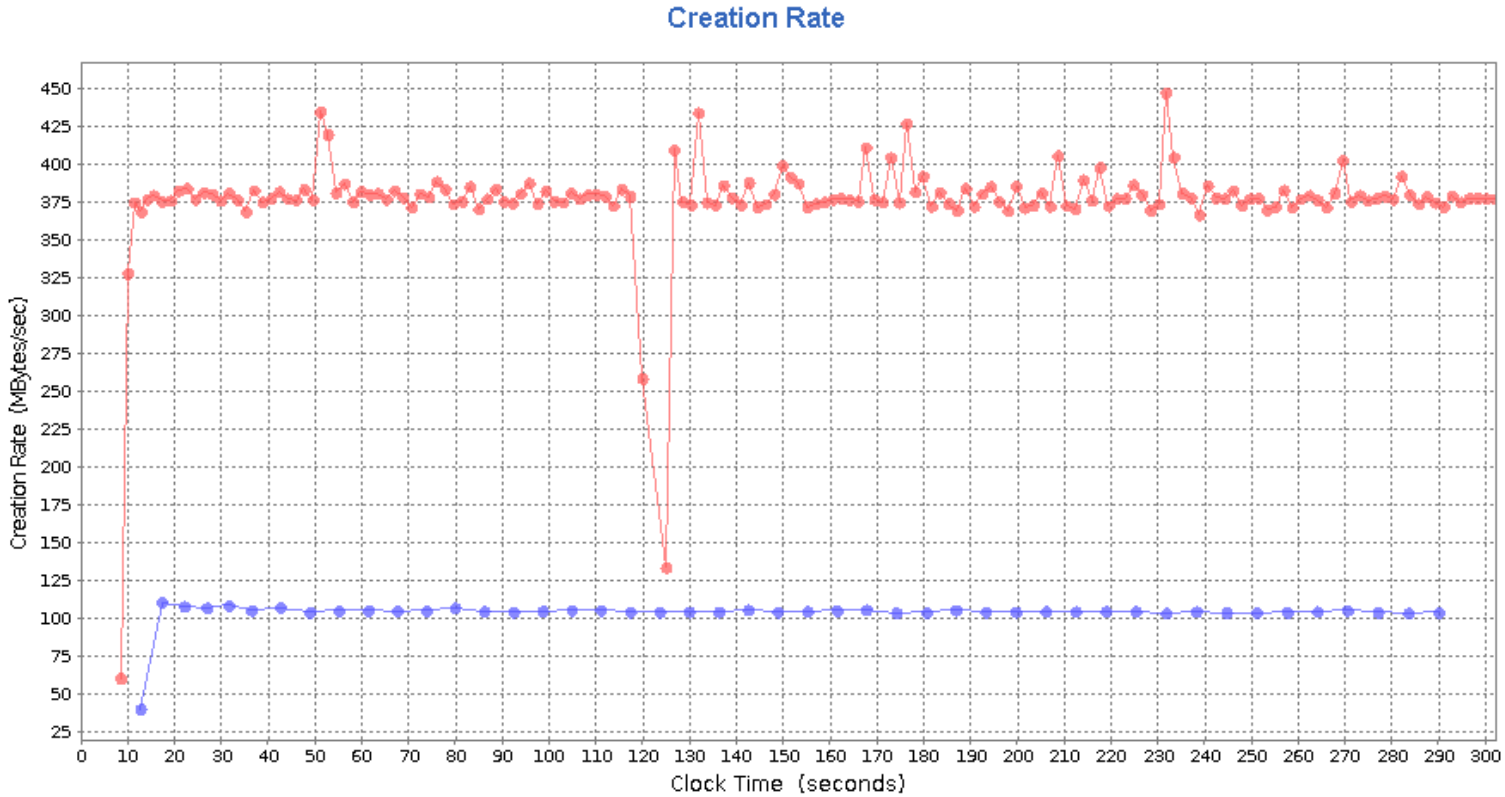
C: 53 lines

1. Manual memory management
2. Even faster

Overview


Node
rings : List<RingDescription>
main()
start()
stop()
1
*
RingDescription
nodeID : Integer
ringID : Integer
ring : RingManager
roles : List<Role>
1
1
RingManager
nodeID : Integer
ringID : Integer
ring : List<Integer>
zoo : ZooKeeper
conf : Map<String,String>
network : NetworkManager
init()
registerNode()
notifyRingChanged()
notifyNewCoordinator()
getQuorum()
getCoordinatorID()
getRingSuccessorID(id : Integer)
getRingPredecessorID(id : Integer)
getLastAcceptorID()
getNodeAddress(id : Integer)
process(event : WatchedEvent)
registerRole(r : PaxosRole)
1
1..4
<<abstract>> Role
deliver(m : Message)
AcceptorRole
ProposerRole
LearnerRole
CoordinatorRole
MultiLearnerRole

Serialization



Java object creation rate in MByte/s. (protobuf/direct serialization)

Stable Storage

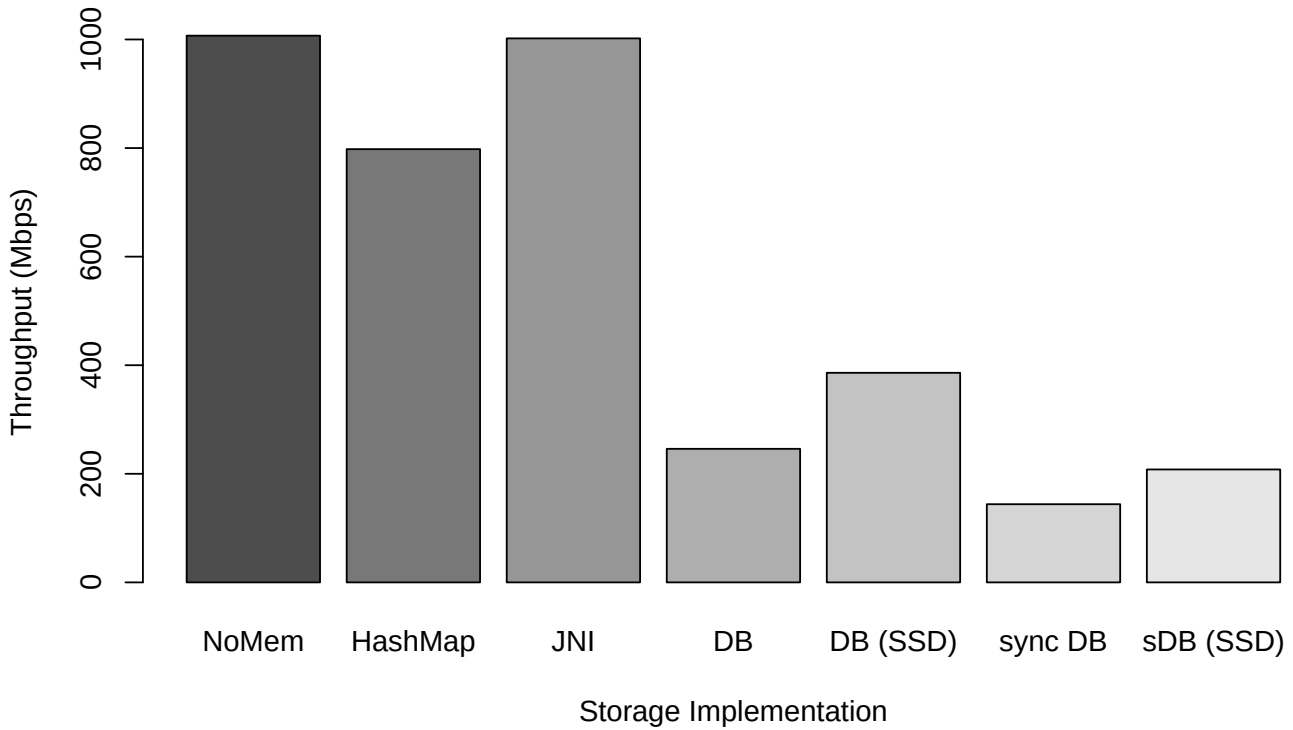


Table of Contents

Environment

USI (20 nodes)

1. 8 cores, 8 GB RAM, local disk: 7.2k RPM and SSD
2. 1 Gbit/s network connections

Switch (6 nodes)

1. 4 cores, 16 GB RAM, kvm virtualization
2. 10 Gbit/s virtualized network adapters

Amazon EC2 (10 nodes)

1. 2 cores, 3.7 GB RAM, m1.medium instances
2. N. Virginia, Ireland, Oregon

Experiments

Ring Paxos performance

1. TCP buffer size
2. Value size
3. Ring performance
4. Ring size

Multi-Ring Paxos performance

1. Efficiency of the skip messages
2. Scale disk writes
3. Scale network usage
4. Globally deployed rings

Experiments

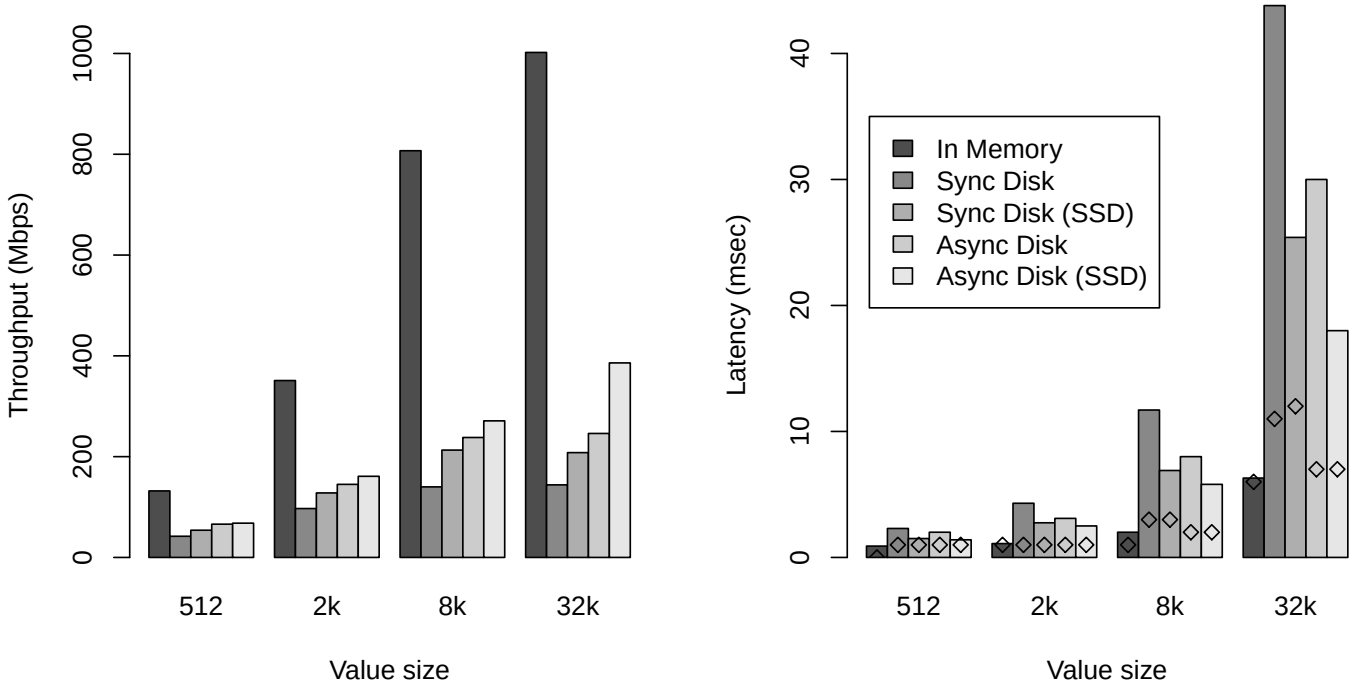
Ring Paxos performance

1. TCP buffer size
2. Value size
3. Ring performance
4. Ring size

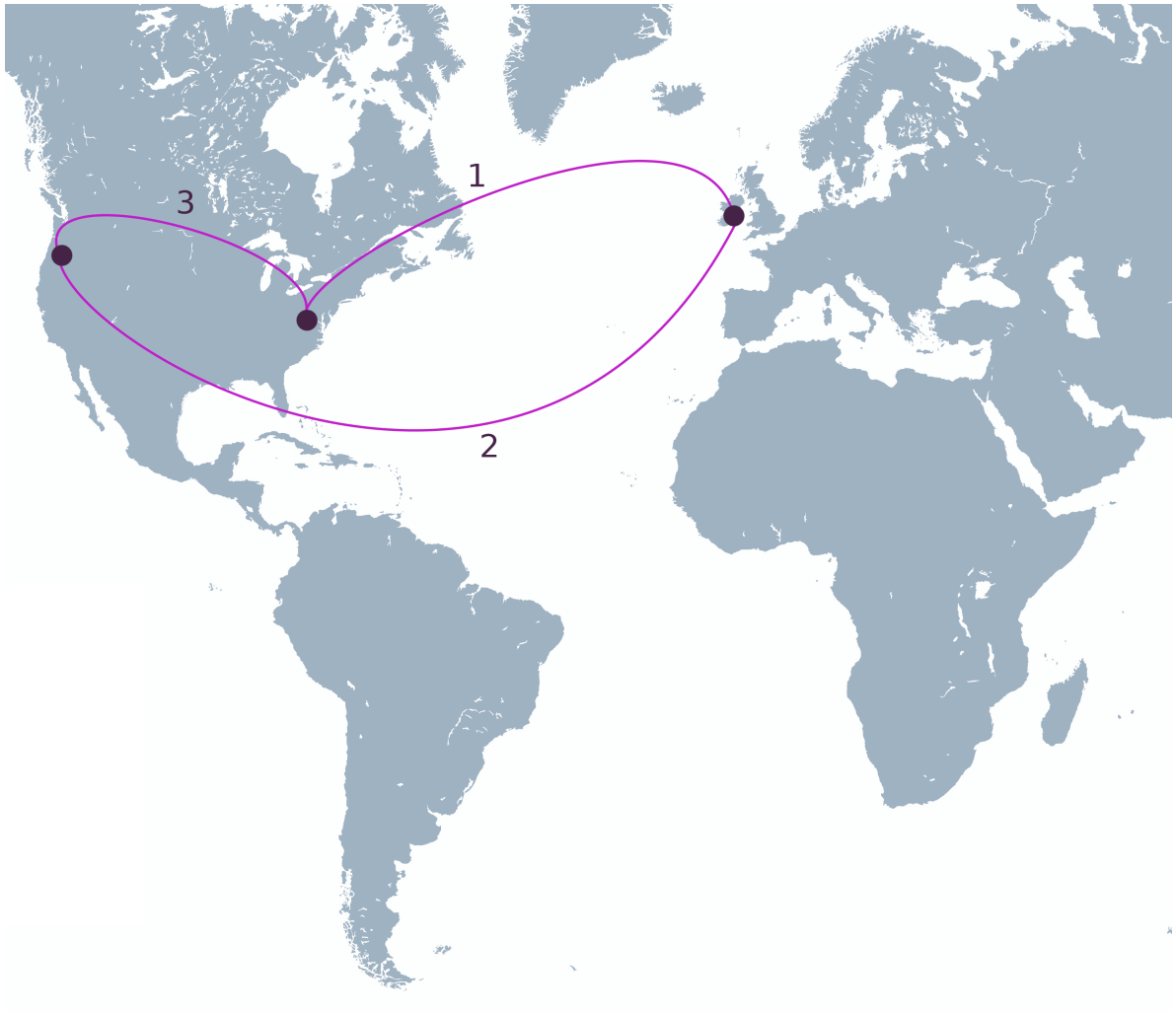
Multi-Ring Paxos performance

1. Efficiency of the skip messages
2. Scale disk writes
3. Scale network usage
4. Globally deployed rings

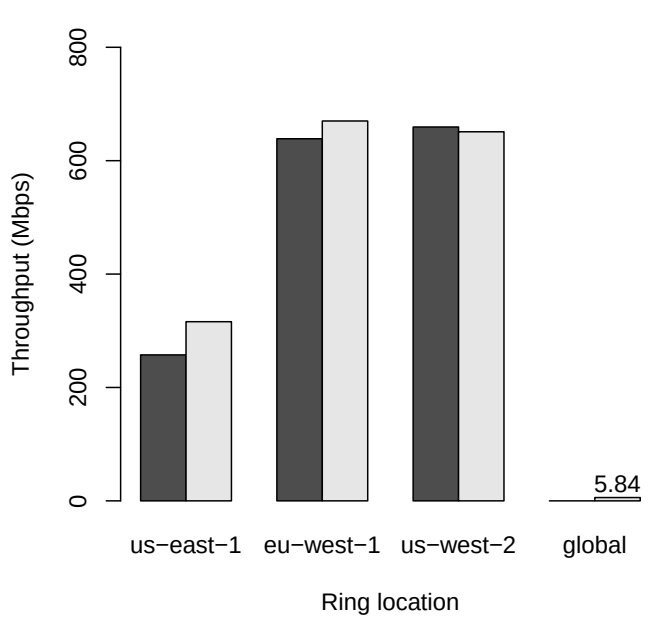
USI


Throughput (Mbps)
0
200
400
600
800
1000
512
2k
8k
32k
Value size
Latency (msec)
0
10
20
30
40
In Memory
Sync Disk
Sync Disk (SSD)
Async Disk
Async Disk (SSD)
512
2k
8k
32k
Value size

Amazon



Amazon


Throughput (Mbps)
0
200
400
600
800
5.84
us−east−1
eu−west−1
us−west−2
global
Ring location

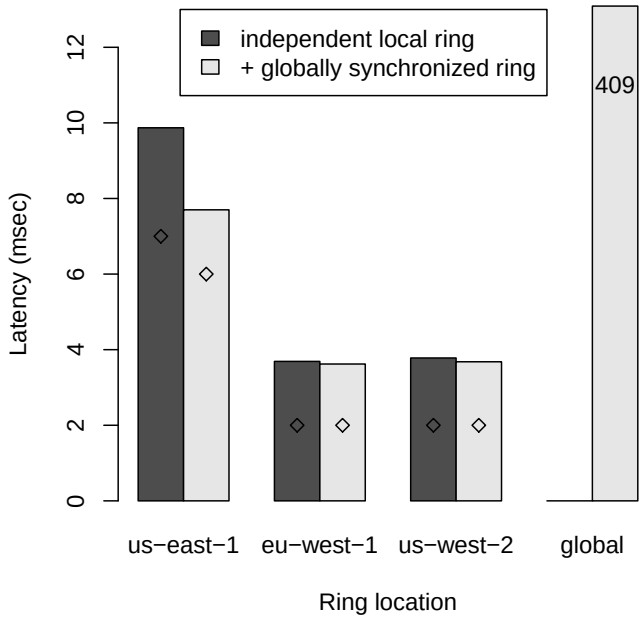

independent local ring
+ globally synchronized ring
Latency (msec)
0
2
4
6
8
10
12
409
us−east−1
eu−west−1
us−west−2
global
Ring location

Table of Contents

Discussion

Contributions

1. Ring Paxos performance with different storage evaluated
2. Multi-Ring Paxos scalability evaluated
   - Disk writes
   - Network usage
3. Previous results validated with new implementation
4. New scenarios could be tested with unicast connections
   - Globally deployed rings

Future work

Implementation

1. Automatic latency optimal ring sorting
2. Improved TCP framing

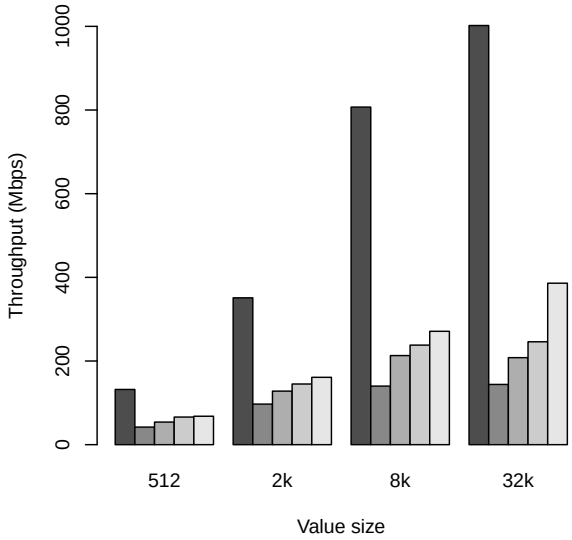
Research

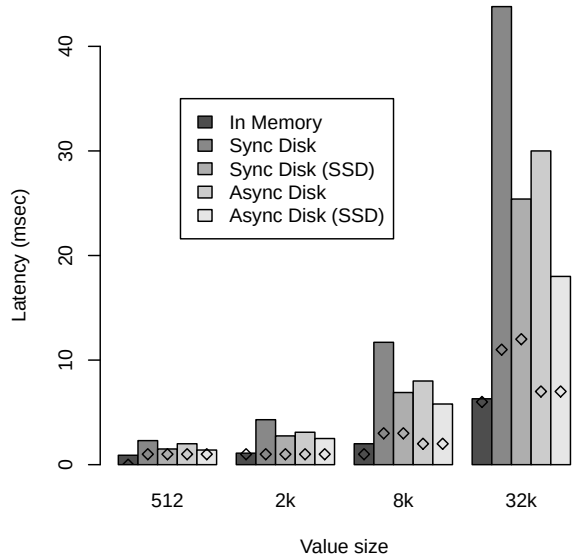
3. Application aware acceptor storage

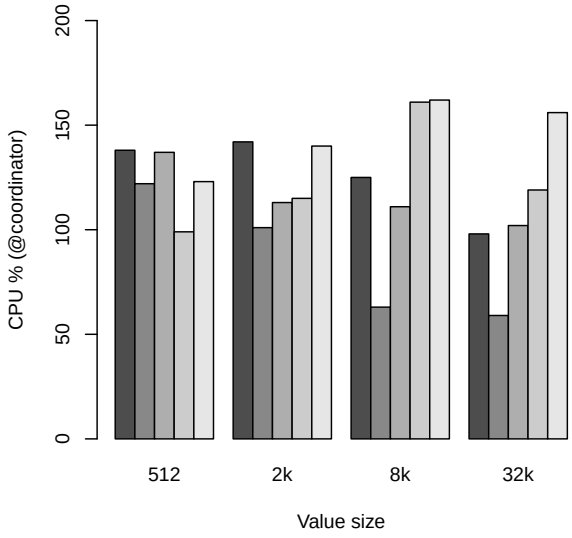
https://github.com/sambenz/URingPaxos

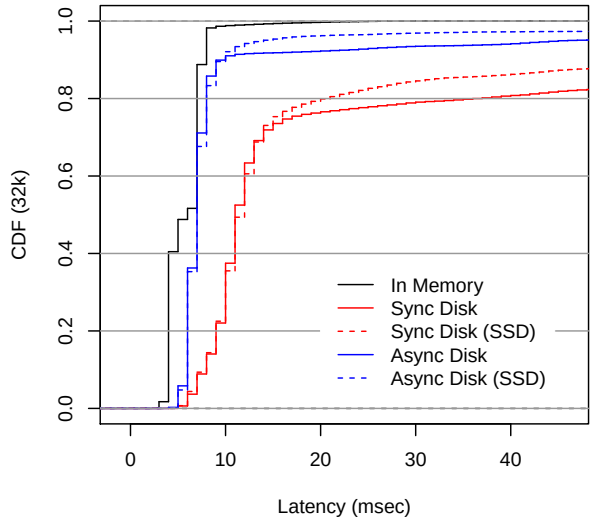
Table of Contents

USI

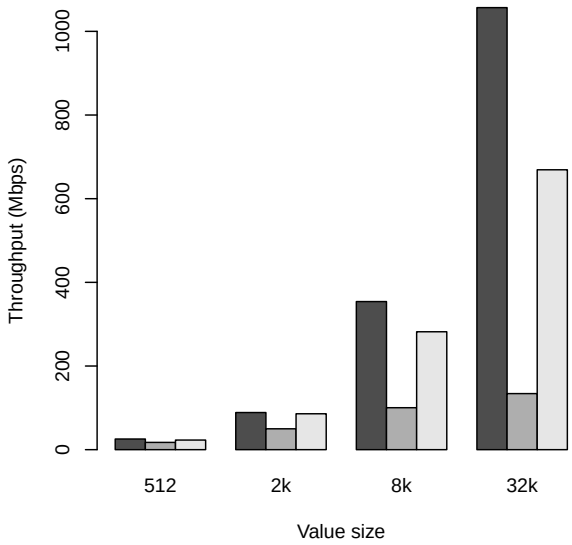

Throughput (Mbps)
0
200
400
600
800
1000
512
2k
8k
32k
Value size

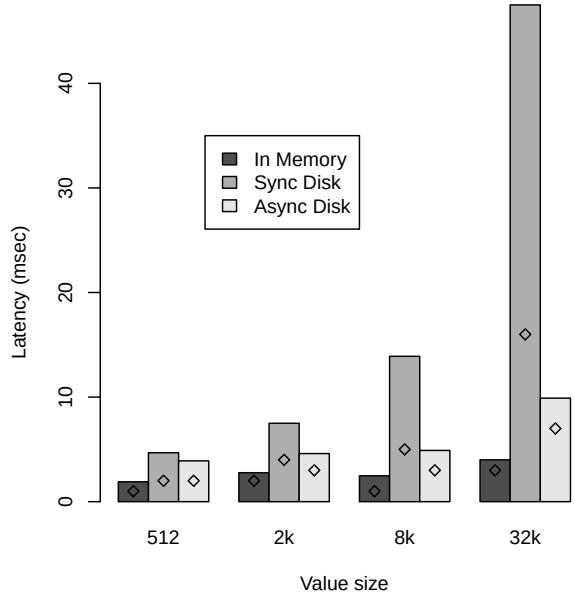

Latency (msec)
0
10
20
30
40
In Memory
Sync Disk
Sync Disk (SSD)
Async Disk
Async Disk (SSD)
512
2k
8k
32k
Value size

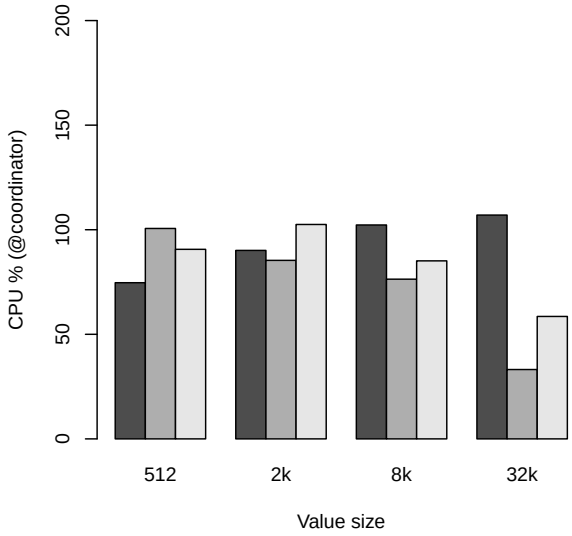

CPU % (@coordinator)
0
50
100
150
200
512
2k
8k
32k
Value size

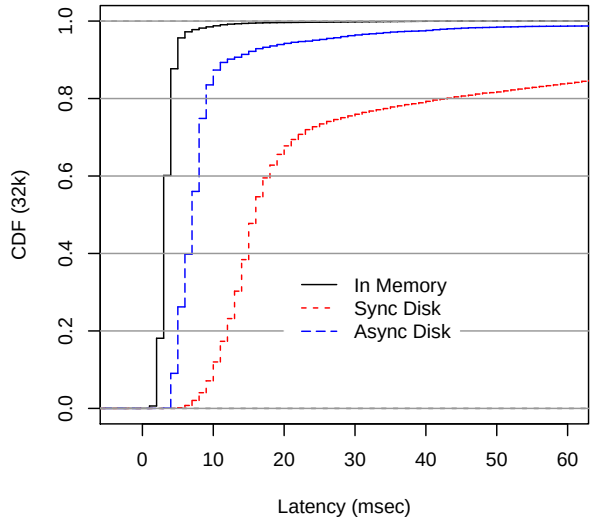

CDF (32k)
0.0
0.2
0.4
0.6
0.8
1.0
0
10
20
30
40
In Memory
Sync Disk
Sync Disk (SSD)
Async Disk
Async Disk (SSD)
Latency (msec)

Switch

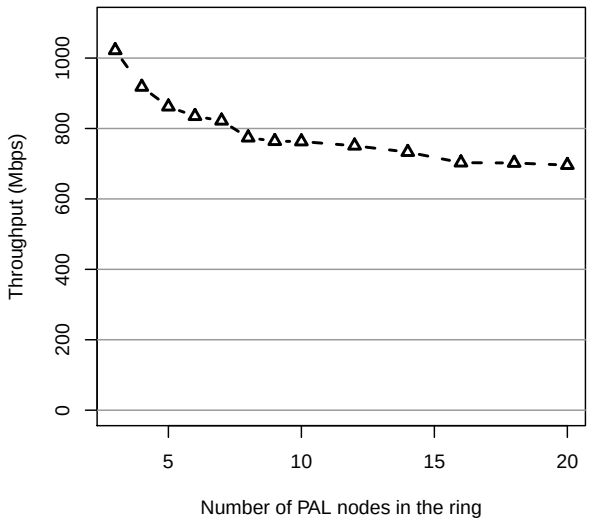

Throughput (Mbps)
0
200
400
600
800
1000
512
2k
8k
32k
Value size

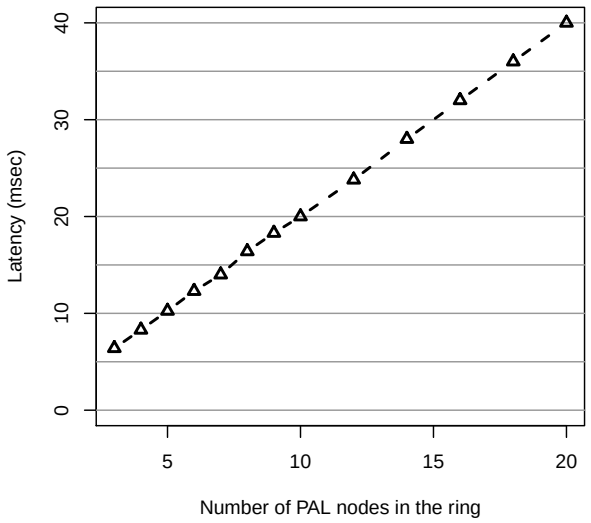

Latency (msec)
0
10
20
30
40
In Memory
Sync Disk
Async Disk
512
2k
8k
32k
Value size

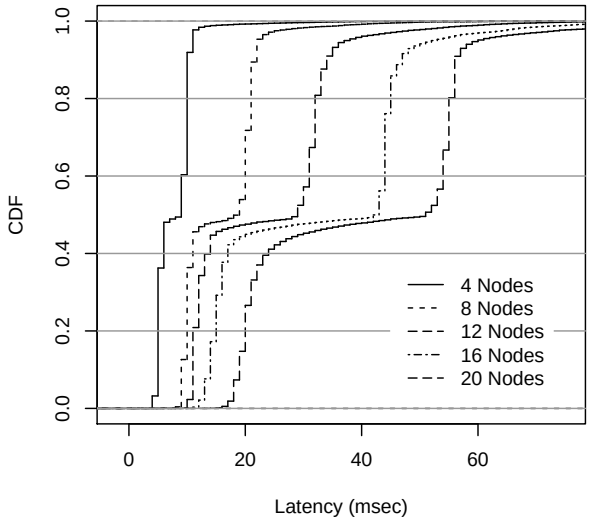

CPU % (@coordinator)
0
50
100
150
200
512
2k
8k
32k
Value size


CDF (32k)
0.0
0.2
0.4
0.6
0.8
1.0
In Memory
Sync Disk
Async Disk
0
10
20
30
40
50
60
Latency (msec)

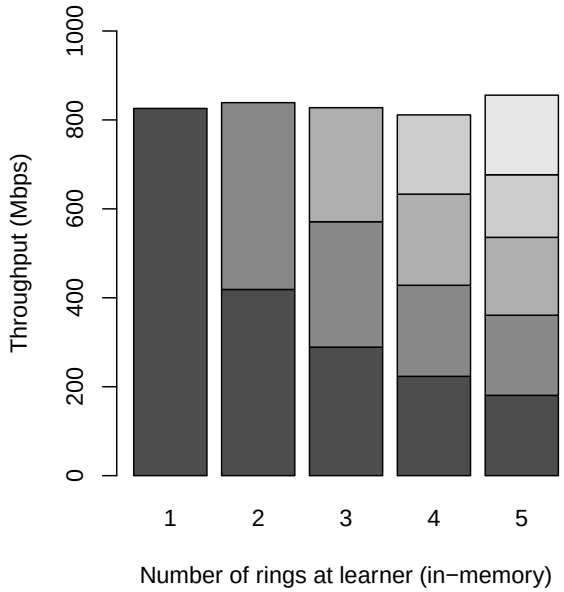
USI

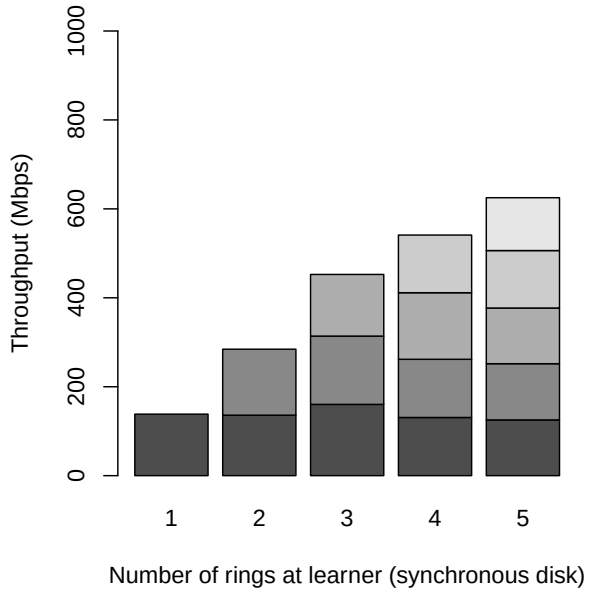

Throughput (Mbps)
0
200
400
600
800
1000
5
10
15
20
Number of PAL nodes in the ring


Latency (msec)
0
10
20
30
40
5
10
15
20
Number of PAL nodes in the ring

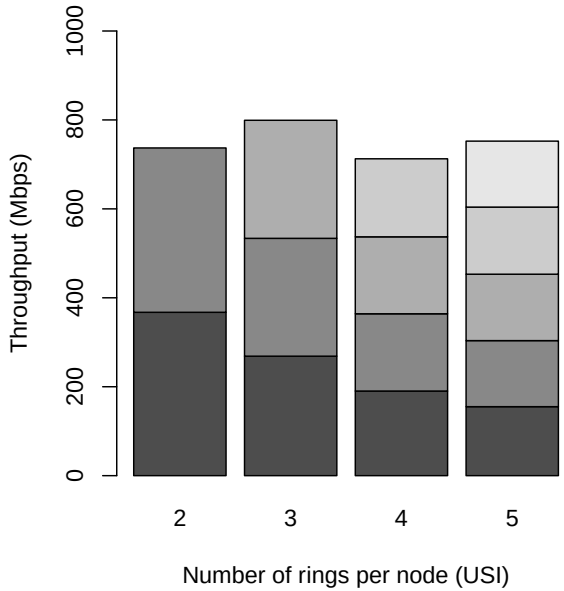

CDF
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
Latency (msec)
4 Nodes
8 Nodes
12 Nodes
16 Nodes
20 Nodes

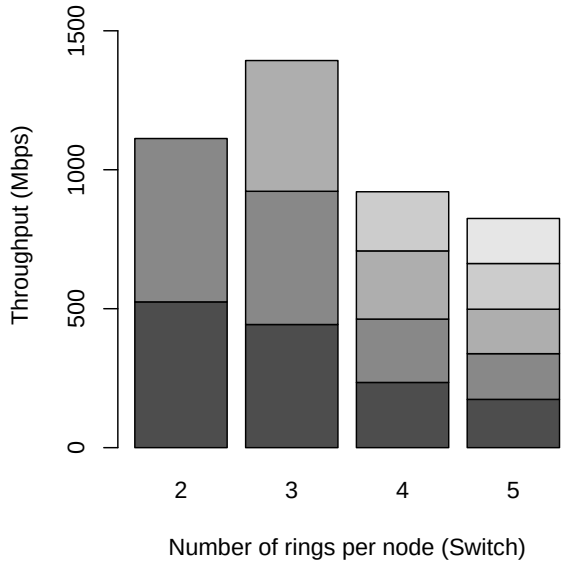
USI: Scaling disk writes



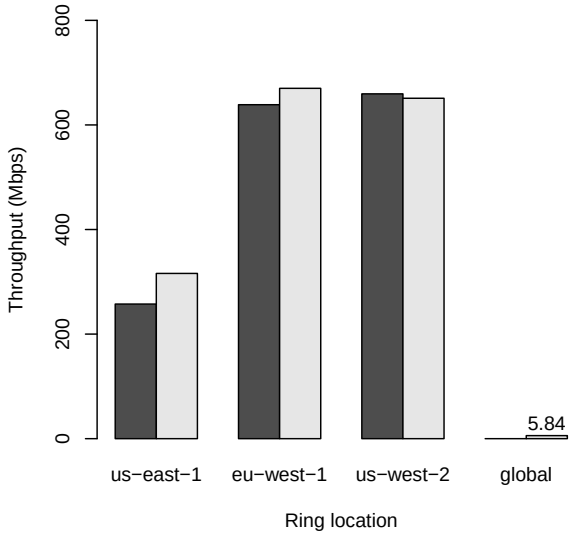


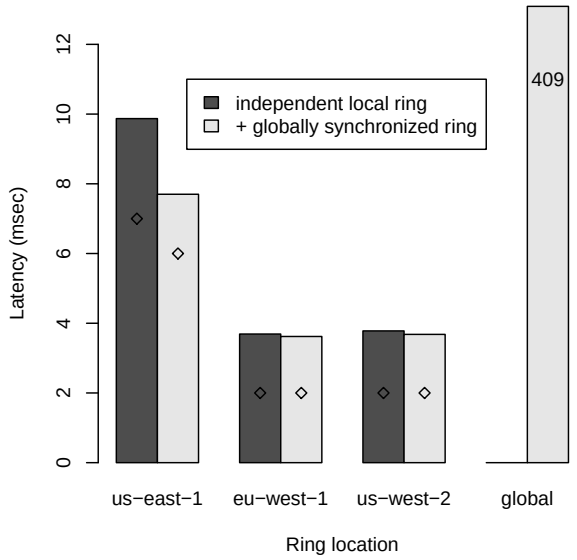
Switch: Scaling network usage

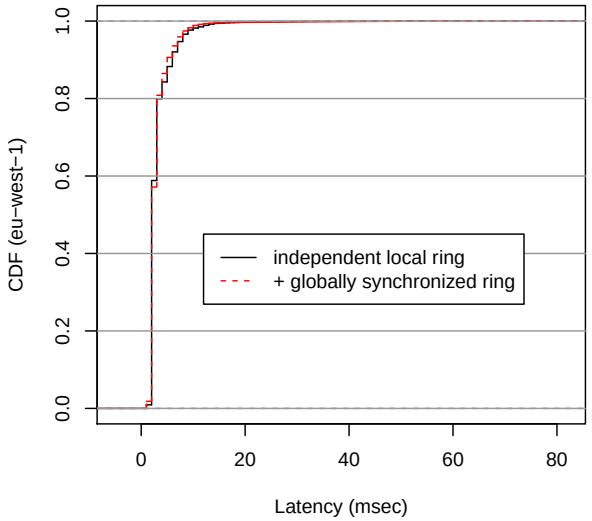


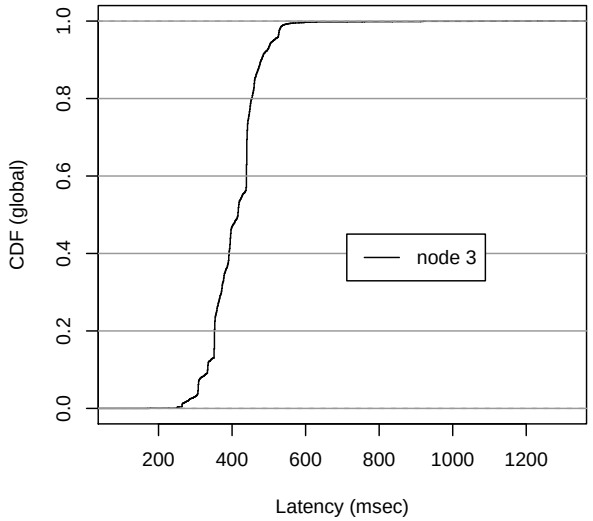


Amazon

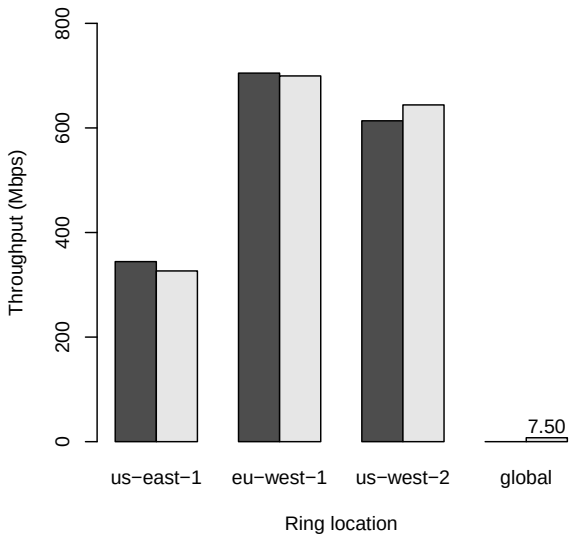

Throughput (Mbps)
0
200
400
600
800
5.84
us−east−1
eu−west−1
us−west−2
global
Ring location

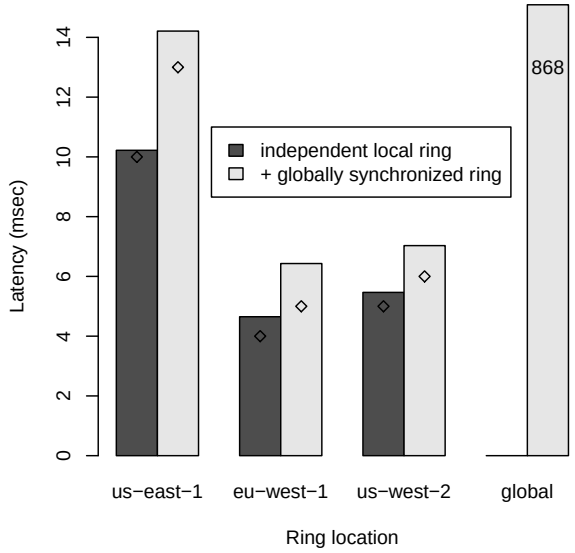

Latency (msec)
0
2
4
6
8
10
12
409
independent local ring
+ globally synchronized ring
us−east−1
eu−west−1
us−west−2
global
Ring location

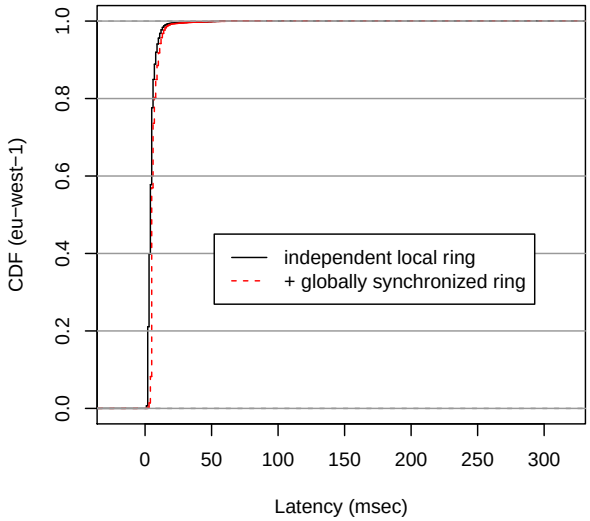

CDF (eu−west−1)
0.0
0.2
0.4
0.6
0.8
1.0
independent local ring
+ globally synchronized ring
0
20
40
60
80
Latency (msec)

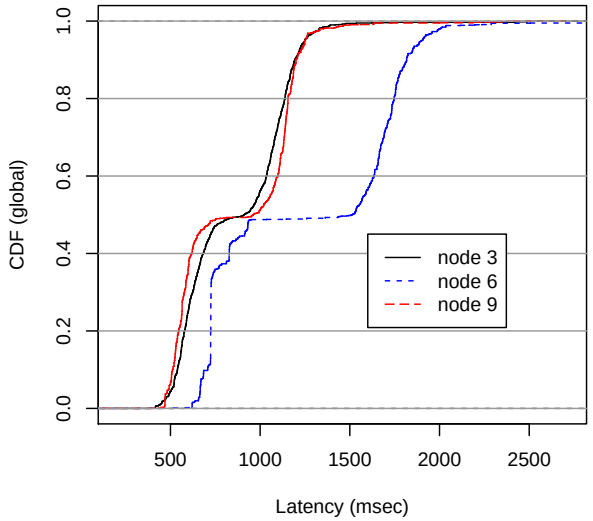

CDF (global)
0.0
0.2
0.4
0.6
0.8
1.0
node 3
200
400
600
800
1000
1200
Latency (msec)

Amazon

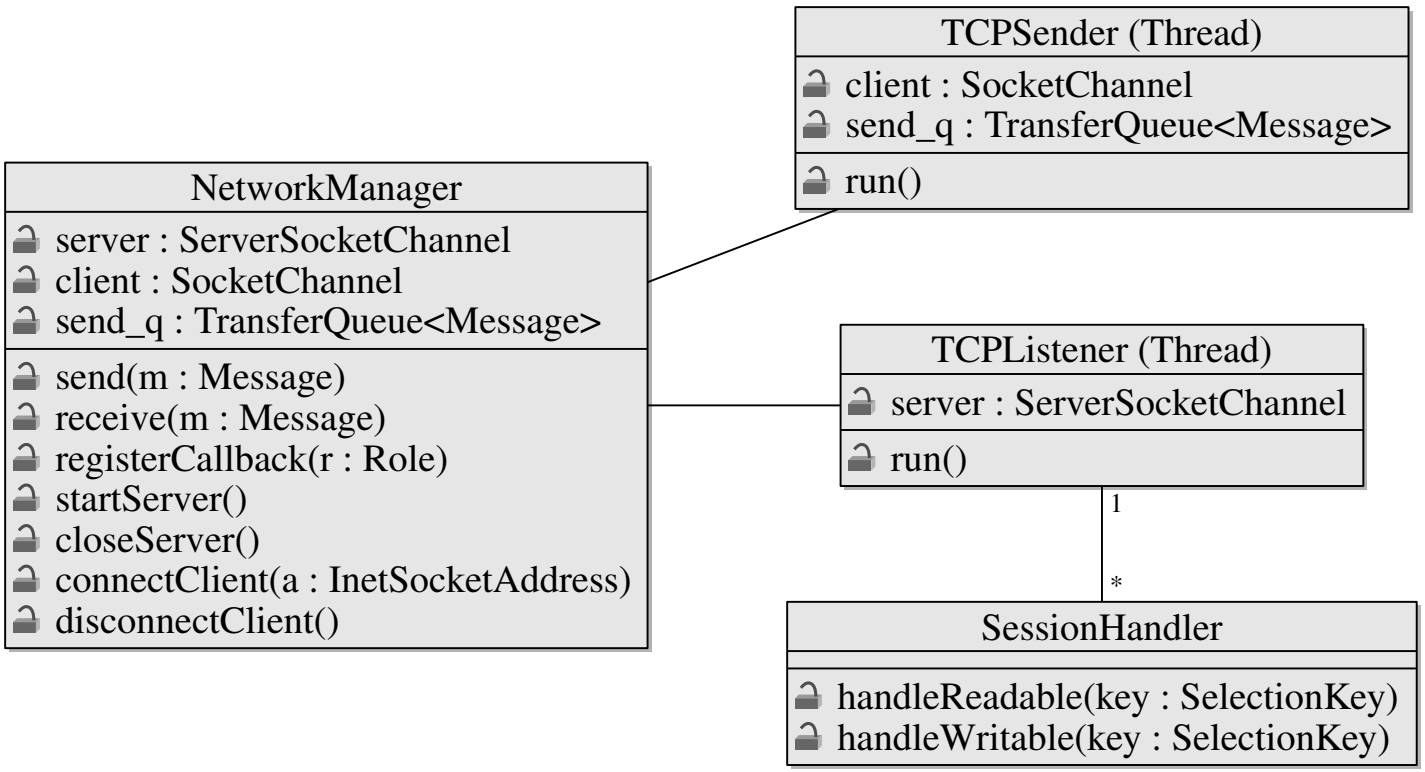

Throughput (Mbps)
0
200
400
600
800
7.50
us−east−1
eu−west−1
us−west−2
global
Ring location


Latency (msec)
0
2
4
6
8
10
12
14
868
independent local ring
+ globally synchronized ring
us−east−1
eu−west−1
us−west−2
global
Ring location

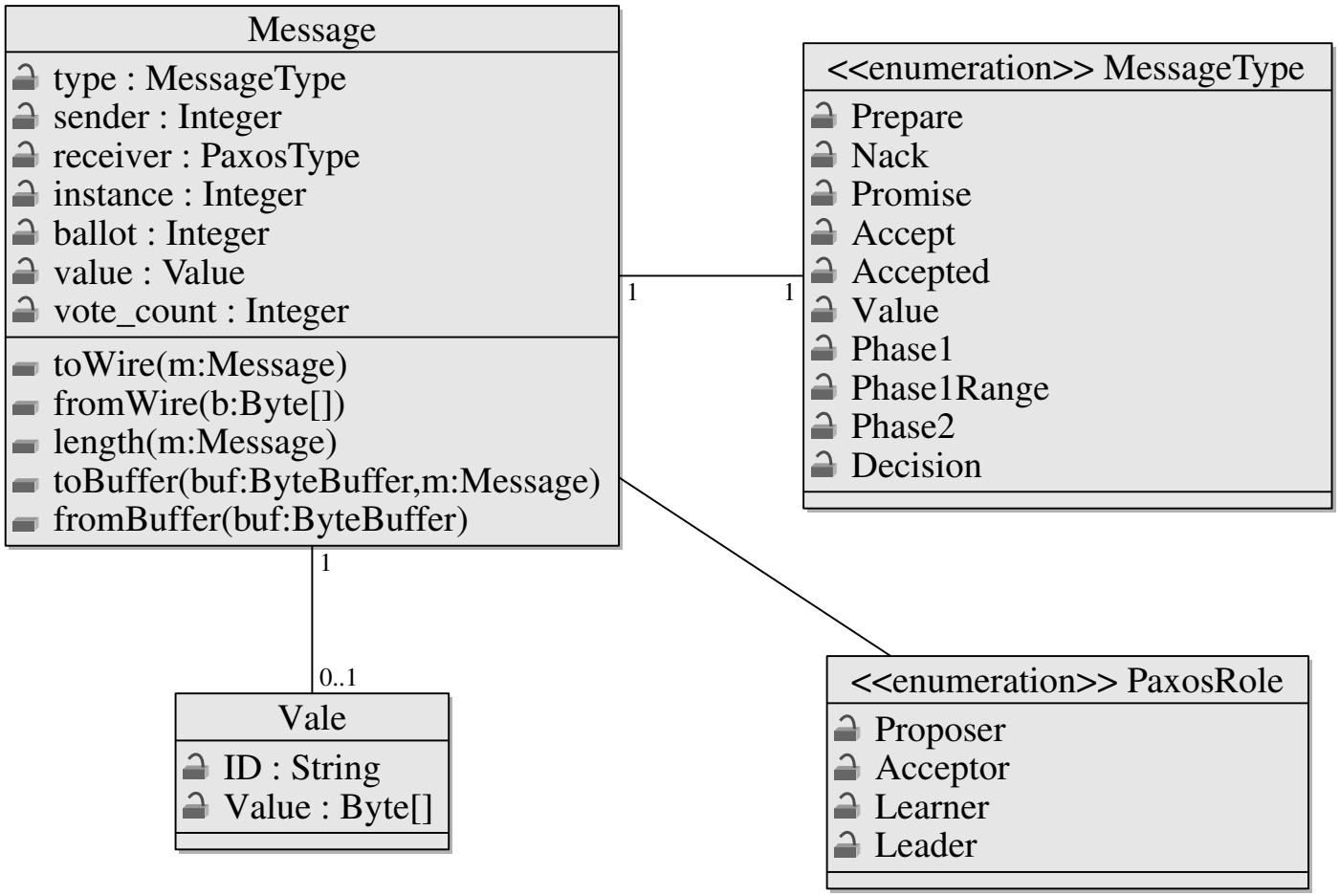

CDF (eu−west−1)
0.0
0.2
0.4
0.6
0.8
1.0
independent local ring
+ globally synchronized ring
0
50
100
150
200
250
300
Latency (msec)


CDF (global)
0.0
0.2
0.4
0.6
0.8
1.0
node 3
node 6
node 9
500
1000
1500
2000
2500
Latency (msec)

Network


TCPSender (Thread)
client : SocketChannel
send_q : TransferQueue<Message>
run()
NetworkManager
server : ServerSocketChannel
client : SocketChannel
send_q : TransferQueue<Message>
send(m : Message)
receive(m : Message)
registerCallback(r : Role)
startServer()
closeServer()
connectClient(a : InetSocketAddress)
disconnectClient()
TCPListener (Thread)
server : ServerSocketChannel
run()
1
*
SessionHandler
handleReadable(key : SelectionKey)
handleWritable(key : SelectionKey)

Message


Message
type : MessageType
sender : Integer
receiver : PaxosType
instance : Integer
ballot : Integer
value : Value
vote_count : Integer
toWire(m:Message)
fromWire(b:Byte[])
length(m:Message)
toBuffer(buf:ByteBuffer,m:Message)
fromBuffer(buf:ByteBuffer)
1
1
<<enumeration>> MessageType
Prepare
Nack
Promise
Accept
Accepted
Value
Phase1
Phase1Range
Phase2
Decision
1
0..1
Vale
ID : String
Value : Byte[]
<<enumeration>> PaxosRole
Proposer
Acceptor
Learner
Leader