MochiDB: A Byzantine Fault Tolerant Datastore

Tigran Tsaturyan
Saravanan Dhakshinamurthy

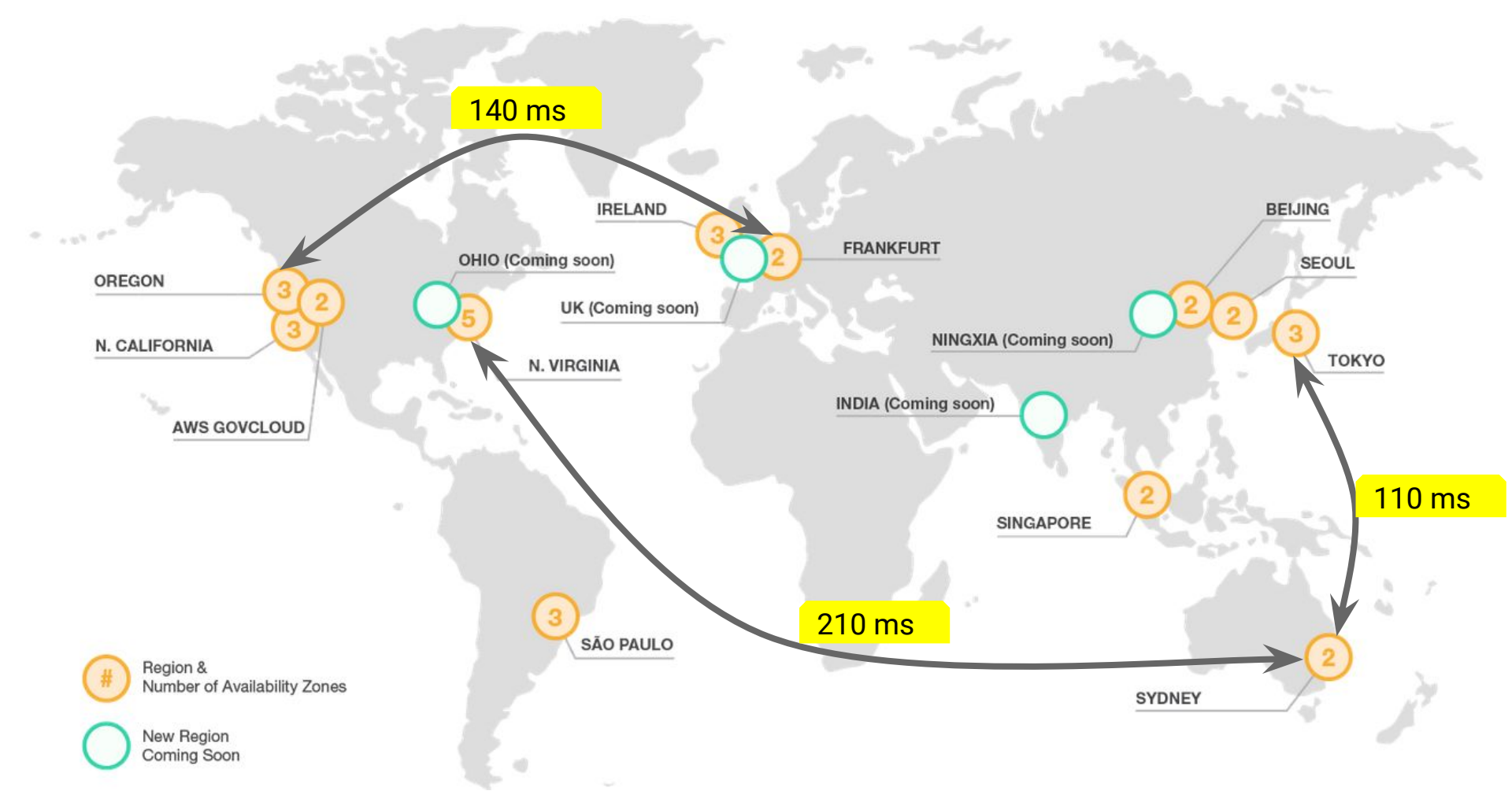
Description

1. BFT KeyValue datastore
   (read(k), write(k,v), delete(k))
2. Consistent
3. Supports transactions
4. In-built sharding
5. Optimized for reads and writes over WAN

Use case

Database to store configurations for infrastructure.

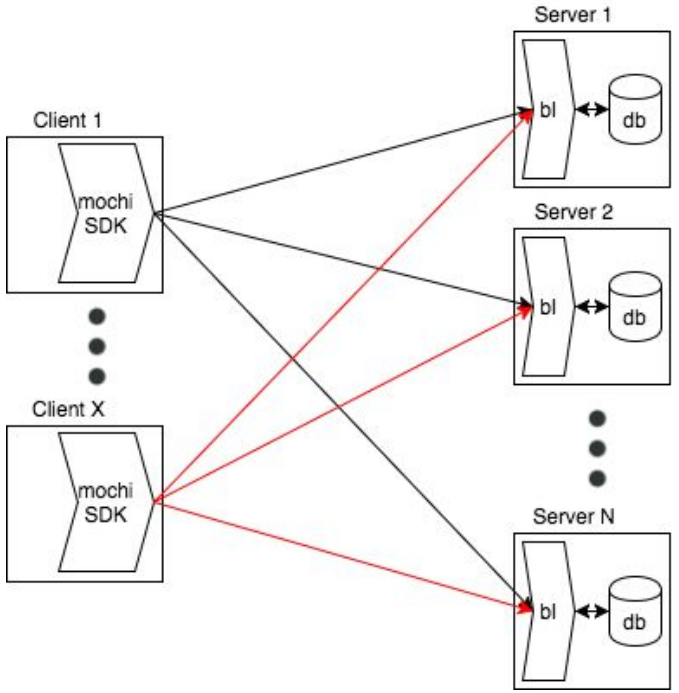
- Most infrastructure as *key -> value*
- Need to update multiple props together
- Infrastructure needs to be consistent
- Located in different part of the world (next slide)



Source: Amazon AWS + https://wondernetwork.com/pings

Architecture

1. Quorum Based BFT Client is a coordinator for transaction
2. Transactions can be two types - READ and WRITE
3. Min server requirement - *3f + 1*



BFT Read

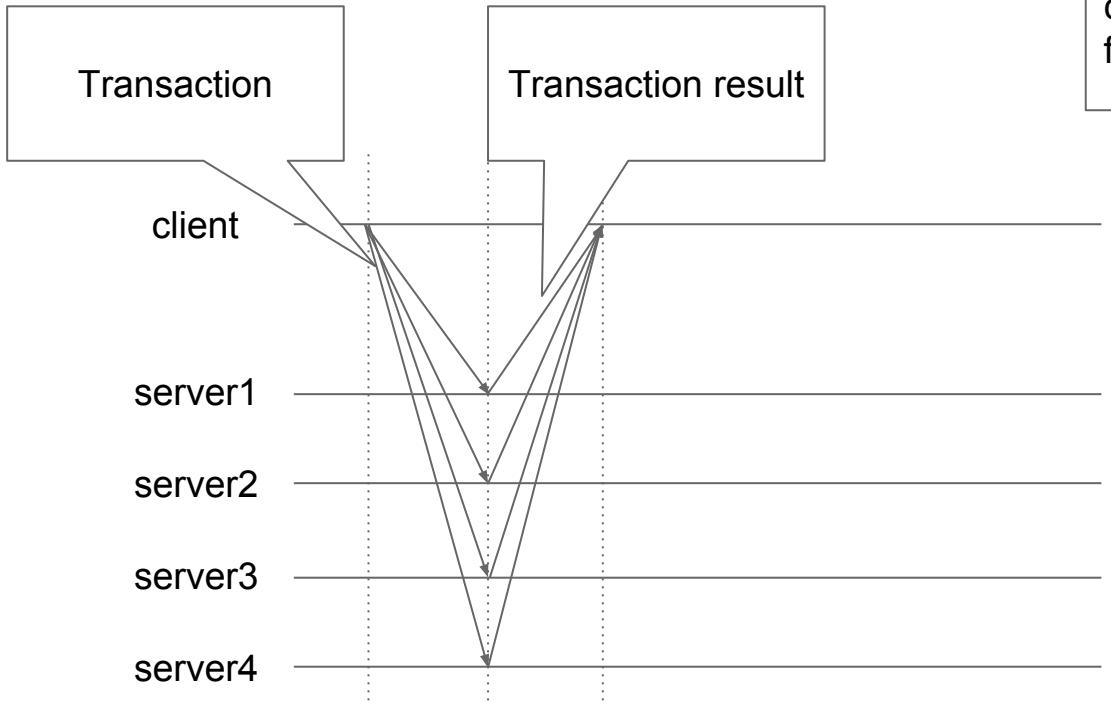
objectX

1. Value
2. WriteCertificate
3. Timestamp (TS)
4. …..

objectY

1. Value
2. WriteCertificate
3. Timestamp (TS)
4. …..

"How that object happens to be that way" (Signed confirmations from the servers)



BFT Write: Protocol view

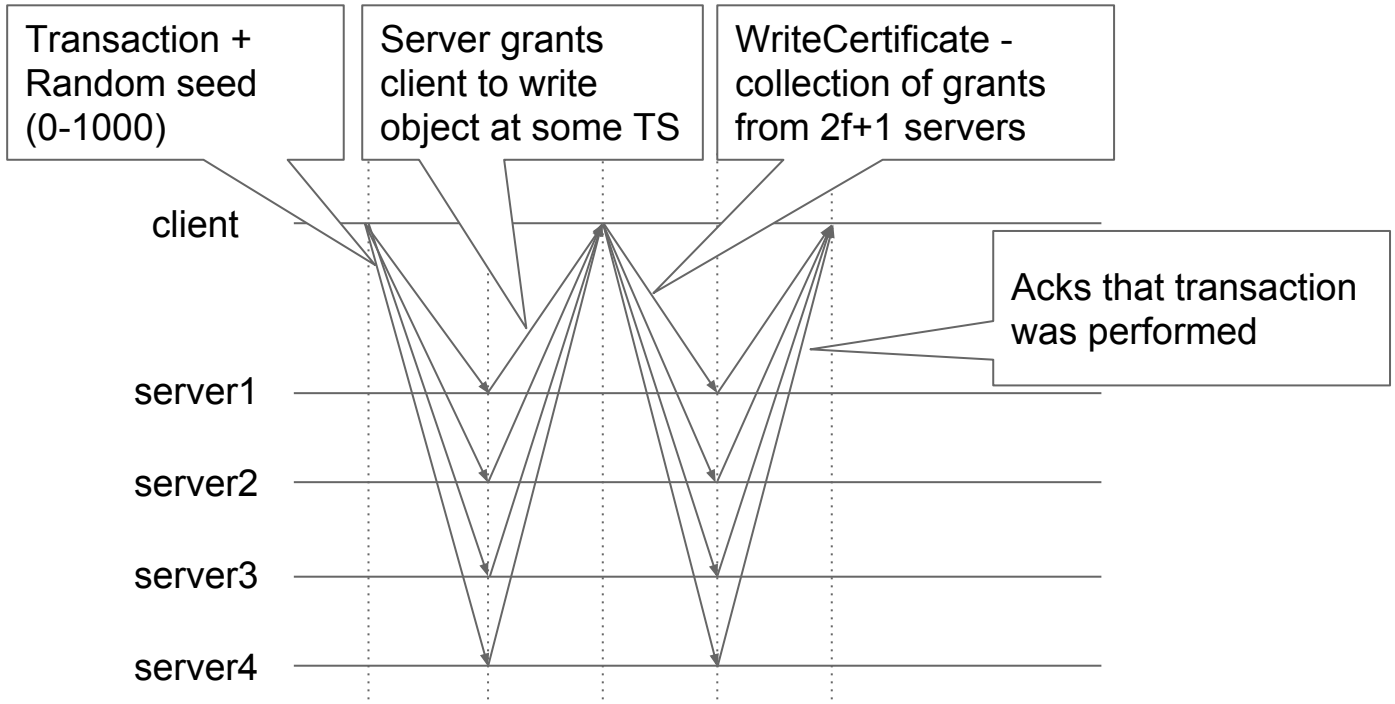
objectX

1. Value
2. WriteCertificate
3. Timestamp (TS)
4. …..

objectY

1. Value
2. WriteCertificate
3. Timestamp (TS)
4. …..

Collection of grants (object, timestamp, trHash)



BFT Write: Server processing

Transaction 1

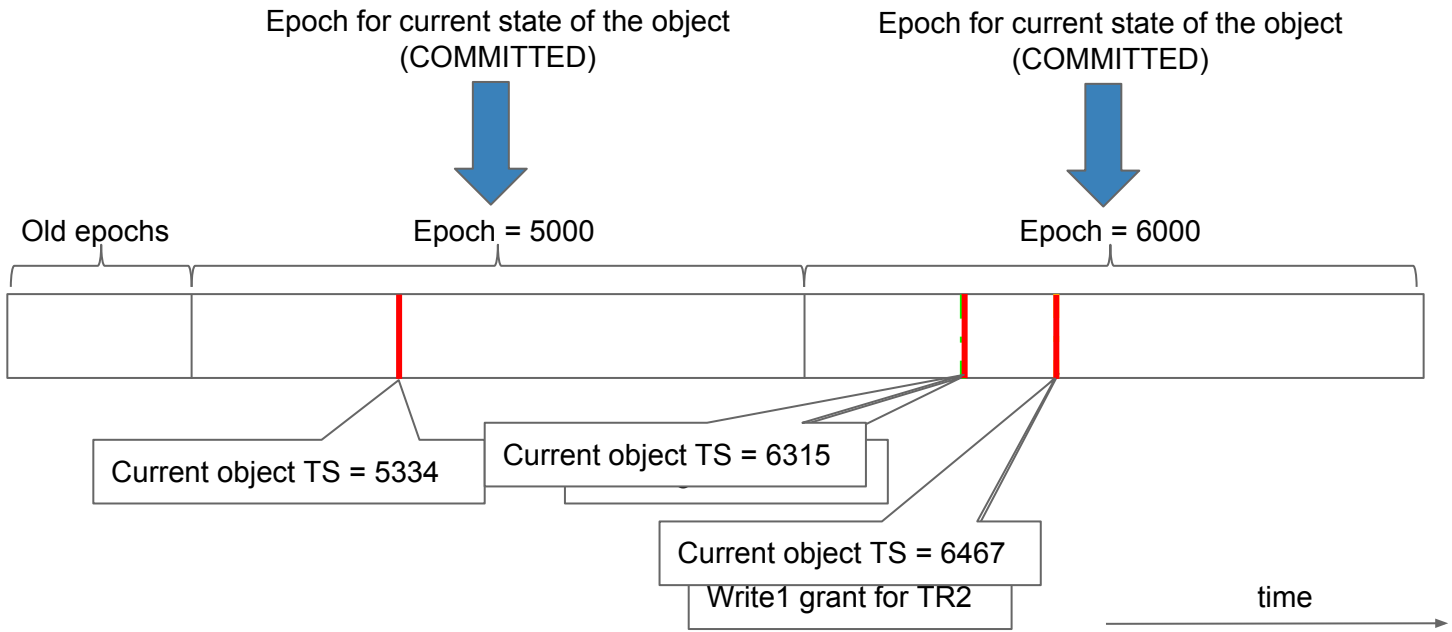
WRITE(“ObjectX”, “12”)
RAND_seed = 315

Transaction 2

WRITE(“ObjectX”, “48”)
RAND_seed = 467

Order

TR1	TR2
Write1	Write1
Write2	
	Write2



Features

- Sharding:
  1024 tokens equally spread across the ring and assign to servers. Data is replicated (*replicationFactor*) on the Nth subsequent servers
- GC:
  Need to cleanup old write grants that are never fulfilled. Server initiates GC, get agreement on object TS, prune non needed data
- Permissions:
  Client have READ, WRITE, ADMIN permissions embedded into its certificate
- Configuration changes:
  Similar to 2PC
- more….

Engineering

Implementation

- Java/Netty/ProtoBufs/Spring
- In-memory object store (for now)

Lessons learned

- Async IO, AWS fees
- Full cluster within JVM and testing framework
- Releasing resources
- Concurrent operations
- Do not make presentation in google docs :)

Testing

- See paper
- Local: 6ms -50%, 20 ms - 99% - READS; 16 ms - 50%, 60 ms - 99% WRITES

Conclusion

THANK YOU!

Ready to run images
https://hub.docker.com/r/mochidb/mochi-db/
Source code (48,310 lines of code):
https://github.com/saravan2/mochi-db

CONTRIBUTIONS APPRECIATED!

Mochi