Multi-Master Replication

Yossi Gottlieb
Redis Labs

redisday
TLV 2017
Traditional Redis Replication

Been there forever...

Master -> Slave(s) model

Slave may be a master for other slaves

Slave may be read/only or read/write

Basic High Availability building block
Scaling to multi-region

Redis is fast!
  > 1,000,000 ops/sec, < 1 millisecond latency

Light is slower...
  > 20 milliseconds us-east to us-west RTT

And network is even slower...
  > 70 milliseconds RTT

Better keep our data near by!
Read Replicas (Single Master)

- DB1: Read/Write replica
- DB2: Read replica
- DB3: Read replica
- DB4: Read replica
- DB5: Read replica
Multi-Master Replication - Requirements

Support a single global data-set

Maintain dataset consistency

Maintain Redis speed

Maintain Redis commands and data-types
Consistency: Consensus is slow
Eventual Consistency

Reads & Writes are done locally

Writes are converged asynchronously

Reads are not guaranteed to be up to date

Global (partial) ordering - vector clocks
Vector Clocks

A
{A=0}
{A=0, B=1, C=0}
{A=1, B=1, C=0}
{A=1, B=2, C=0}  {A=1, B=3, C=3}

B
{B=0}
{A=0, B=0, C=1}  {A=0, B=0, C=2}  {A=0, B=0, C=3}

C
{C=0}
{A=0, B=0, C=1}  {A=0, B=0, C=2}  {A=0, B=0, C=3}

{A=2, B=3, C=3}
What about conflicts?

Conflict Free Replicated Data Types (CRDTs)

Consensus-free consistency

Long researched in the academy, formally verified

Apply to many data types

Enhanced and adapted to Redis

Mostly transparent to users
Example: Counter

INCRBY my-counter 5
GET my-counter
=> 5

INCRBY my-counter 8
GET my-counter
=> 8

GET my-counter
=> 12

GET my-counter
=> 12

Converge
Example: Set (add-wins/observed remove)

SADD my-set element1
SADD my-set element1

SISMEMBER my-set element1
=> 1
SISMEMBER my-set element1
=> 1

SREM my-set element1
SREM my-set element1

SREM my-set element2
SREM my-set element2

SADD my-set element2
SADD my-set element2

SISMEMBER my-set element2
=> 1
SISMEMBER my-set element2
=> 1
Example: LWW-Register (string)

```
SET my-key value1
GET my-key
=> value1
DEL my-key
GET my-key
=> value3
SET my-key value2
GET my-key
=> value1
SET my-key value3
GET my-key
=> value3
```
Status & Availability

Alpha stage

Will be part of the Redis Enterprise product

CRDT is implemented as a Redis 4.x Data Type module

Dedicated Peer Replication mechanism