HA Block Store

Cody Tseng

Ping-Han Chuang

Wuh-Chwen Hwang

Content

- Design
  - Assumptions and Guarantees
  - RPC call
  - System structure
  - Behaviors under failures & recovery
- Experiment Setup
- Performance Measurement
- Test and Demo

Design - Assumptions and Guarantees

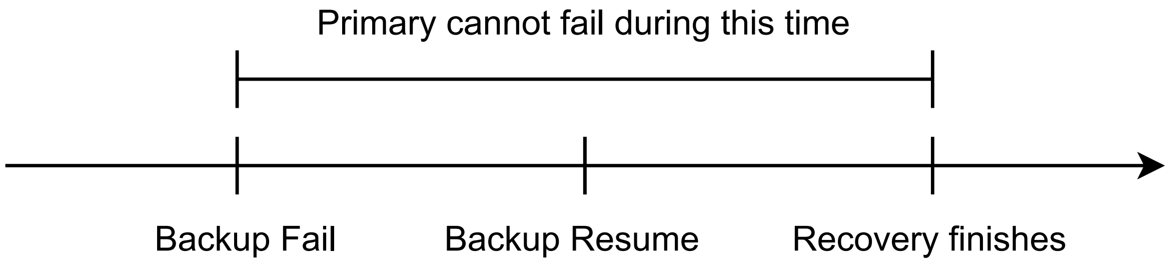
Replication mode: Primary-Backup

Guarantees:

- Availability
- Strongly Consistent

Failure assumptions:

- ≤ 1 server fail-stop
  - Primary cannot fail before recovery finishes
- ≤ 1 network failure (LAN, WAN)



Design - RPC call

Client-Server communication:

```
(status, data) Read  (address)
 status        Write (address, data)
```

Inter-Server communication:

```
 bool          RepliWrite (address, data)
```

Callee: Primary; Caller: Backup

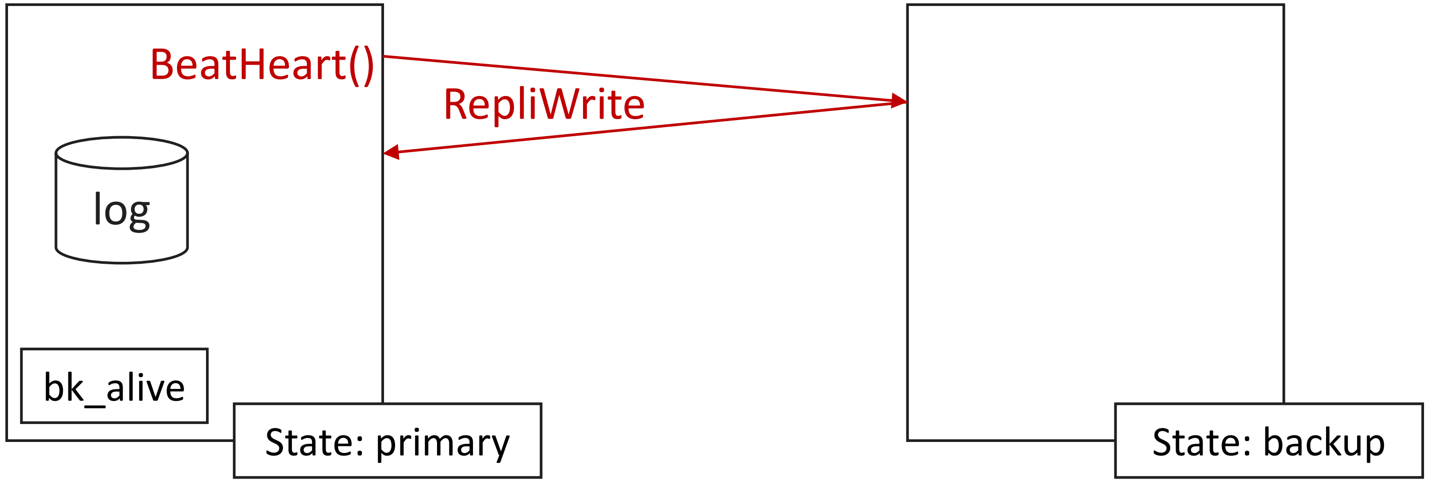
Used for:

- Sending heartbeat (Primary → Backup)
- Primary writes data to backup

Design - System structure

Primary server

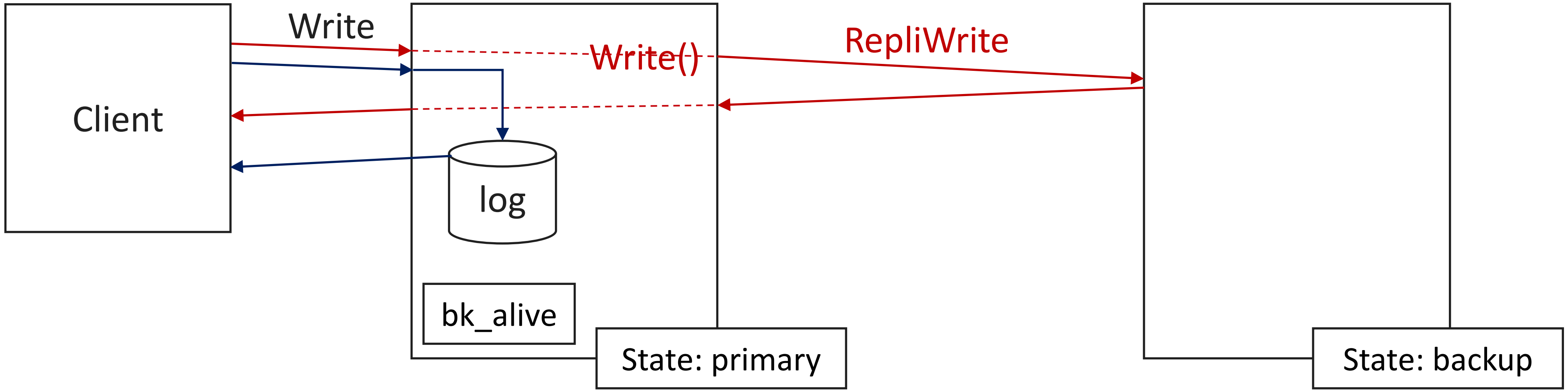
- Heartbeat: Calls `RepliWrite` (empty args) every 200ms
    - If not heard from remote, change `backup_alive` state to `false`



Design - System structure

Primary server

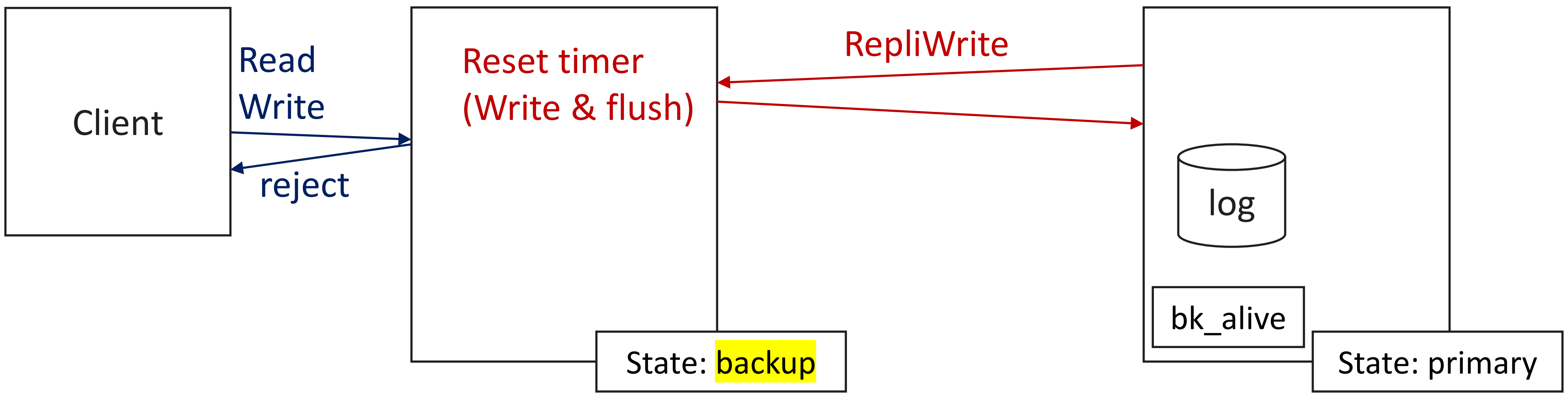
- On receive read request, read and return data.
- On receive write request,
  - If `backup_alive`, call `RepliWrite(addr, data)`
  - If `!backup_alive` or `RepliWrite` failed, write to log
  - Return to client only when data is locally flushed + (written to backup or log)



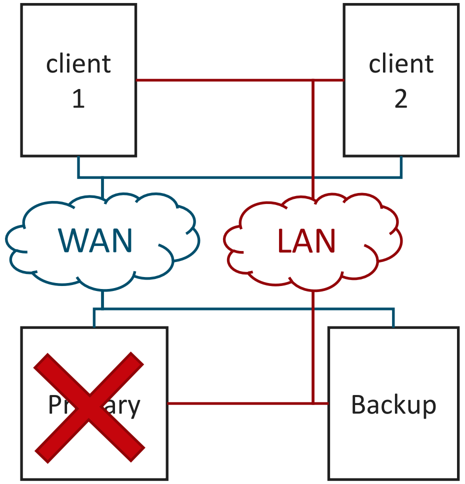
Design - System structure

Backup Server

- On receive `RepliWrite`, reset timer to 1s, (write & flush), return
  - Take over if timer goes off
- On receive client requests, reject requests

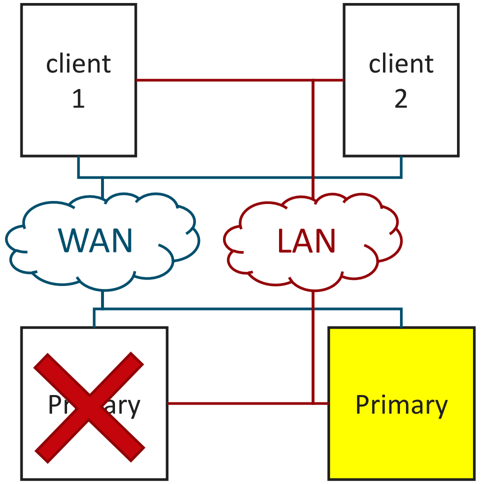


Design - Primary failure



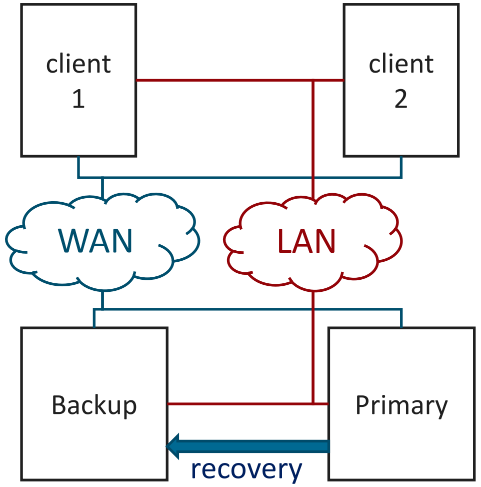
- Right node times out and becomes Primary
- Client write requests stored in Primary’s log
- Left node resume as Backup
  - Primary detects it by heartbeat
- Primary runs recovery

Design - Primary failure



- Right node times out and becomes Primary
- Client write requests stored in Primary’s log
- Left node resume as Backup
  - Primary detects it by heartbeat
- Primary runs recovery

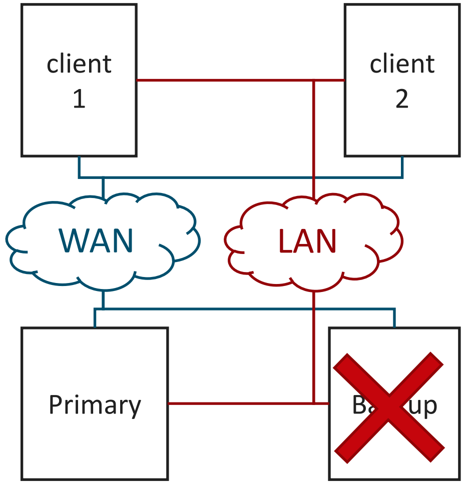
Design - Primary failure



- Right node times out and becomes Primary
- Client write requests stored in Primary’s log
- Left node resume as Backup
  - Primary detects it by heartbeat
- Primary runs recovery

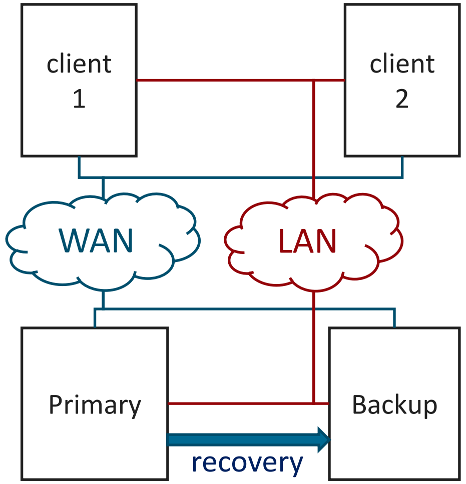
Design – Backup failure

- Client write requests stored in Primary's log
- Right node resume as Backup
- Primary runs recovery



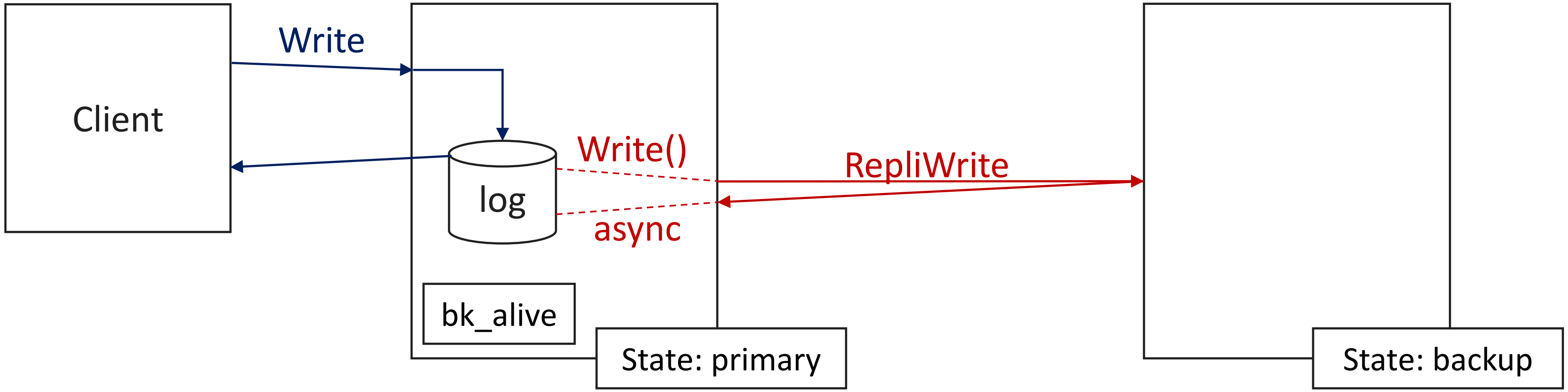
Design – Backup failure

- Client write requests stored in Primary's log
- Right node resume as Backup
- Primary runs recovery



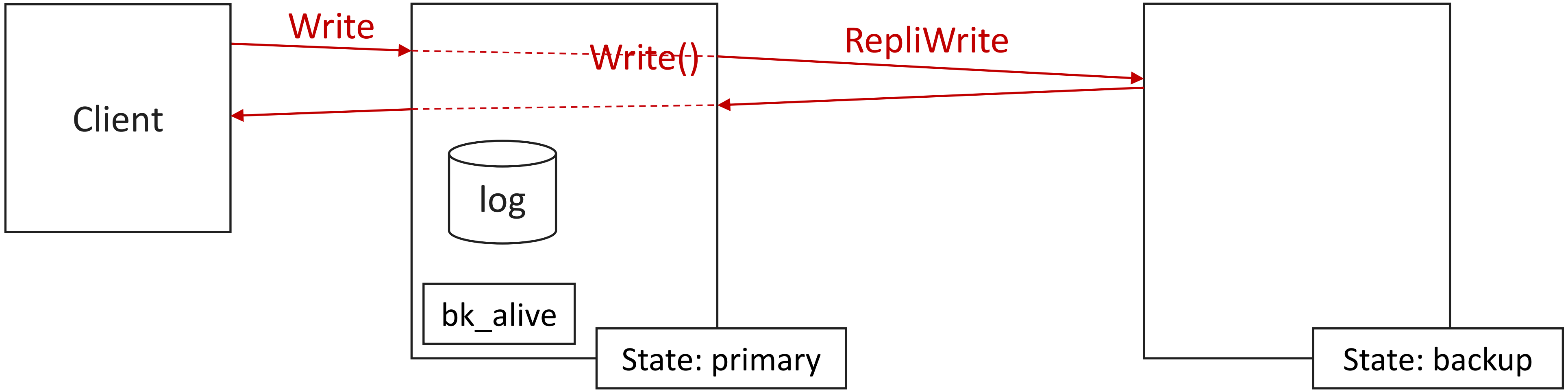
Design – Recovery

- Primary sends log entries to Backup in order
- Primary continues to accept client requests.
- Write requests are saved in log.

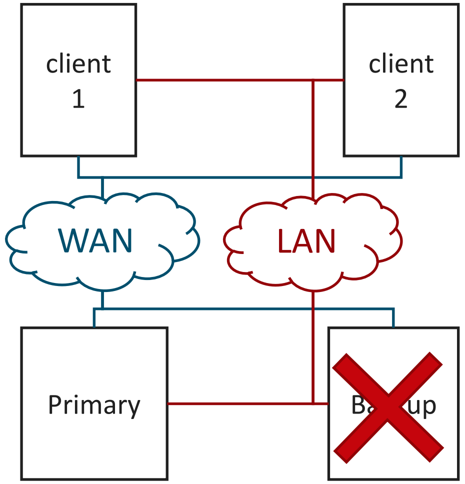


Design – Recovery

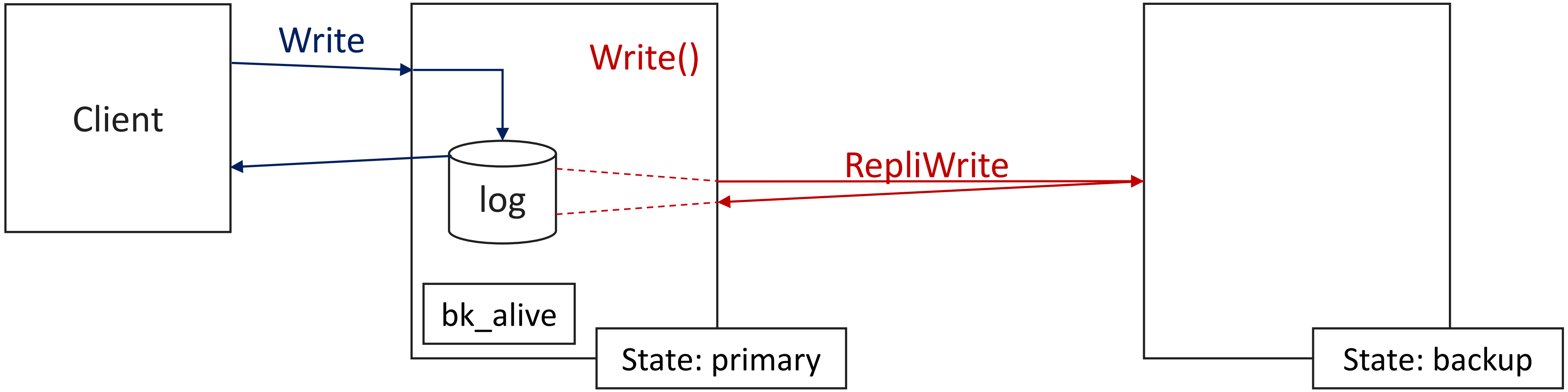
- When log is emptied, change `backup_alive` to `true`
- From then on, client write requests goes to Backup



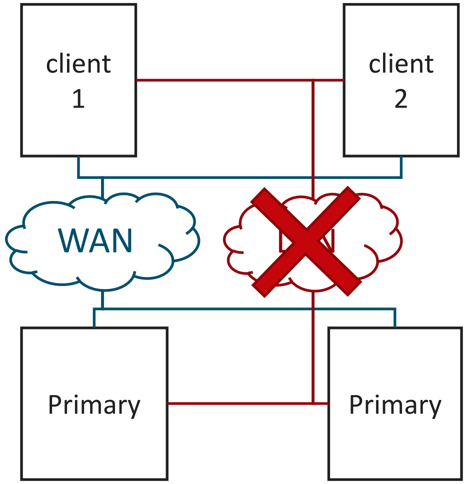
Design – Failure during recovery



- Simply continue the recovery once backup recovers
- Log entries are not removed until `RepliWrite` call successfully returns
- `backup_alive` stays false until "log" is empty



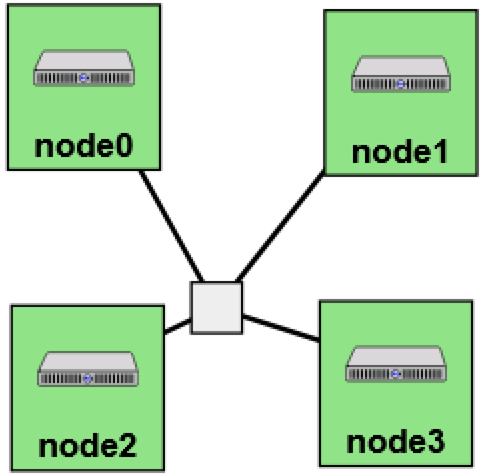
Design - Behaviors under network failure



- 2 networks (LAN, WAN)
- Act normally if either network is working

Experiment Setup

- CloudLab c220g1
  - CPU: 32 logical cores @ 2.40 GHz
  - Memory: 128 GB DDR4 1866 MHz
  - SSD: Intel DC S3500 480 GB 6G SATA SSDs
- Persistence
  - Servers read from/write to raw SSD partition (`/dev/sdc1`). No filesystem.
- Network
  - LAN: 10 Gbps
  - WAN (backup network): 1 Gbps

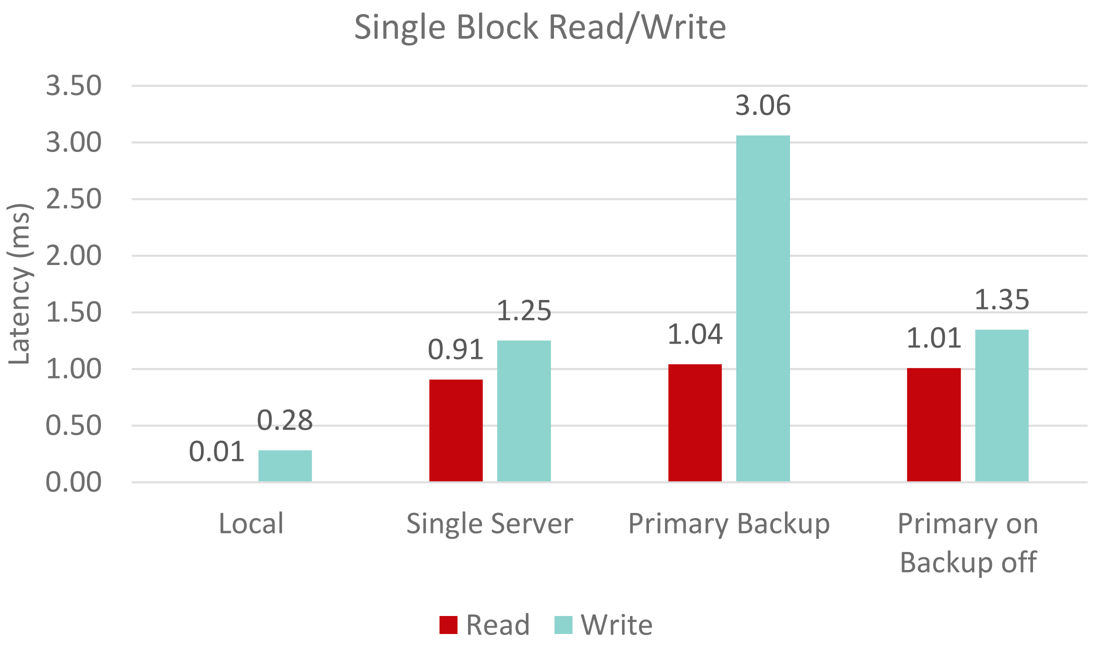


Performance Measurement

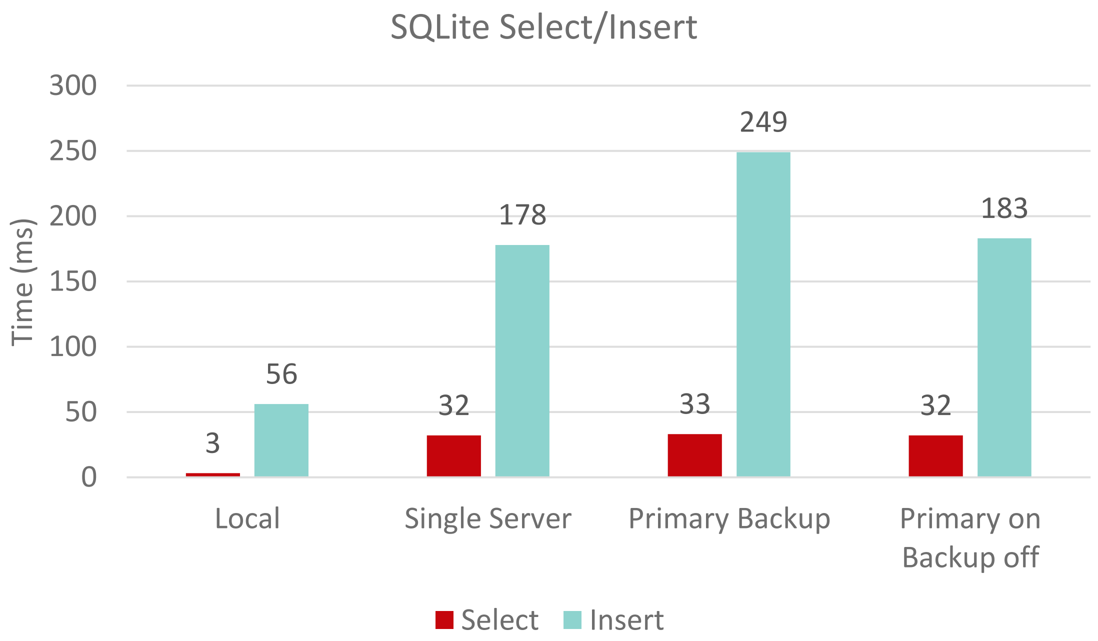
- Latency
  - Workload
    - Single block read/write
    - SQLite select/insert on a simple FUSE filesystem based on HA block store
  - Machine setting
    - Local (single machine)
    - Single server
    - Primary and Backup
    - Primary only (Backup crash)
- 4K-aligned-address request vs. unaligned-address request
- Recovery time

Latency – Single Block / SQLite

- In case of client and server
  - Lowest latency in single server case
  - Highest latency in PB write – extra write to backup



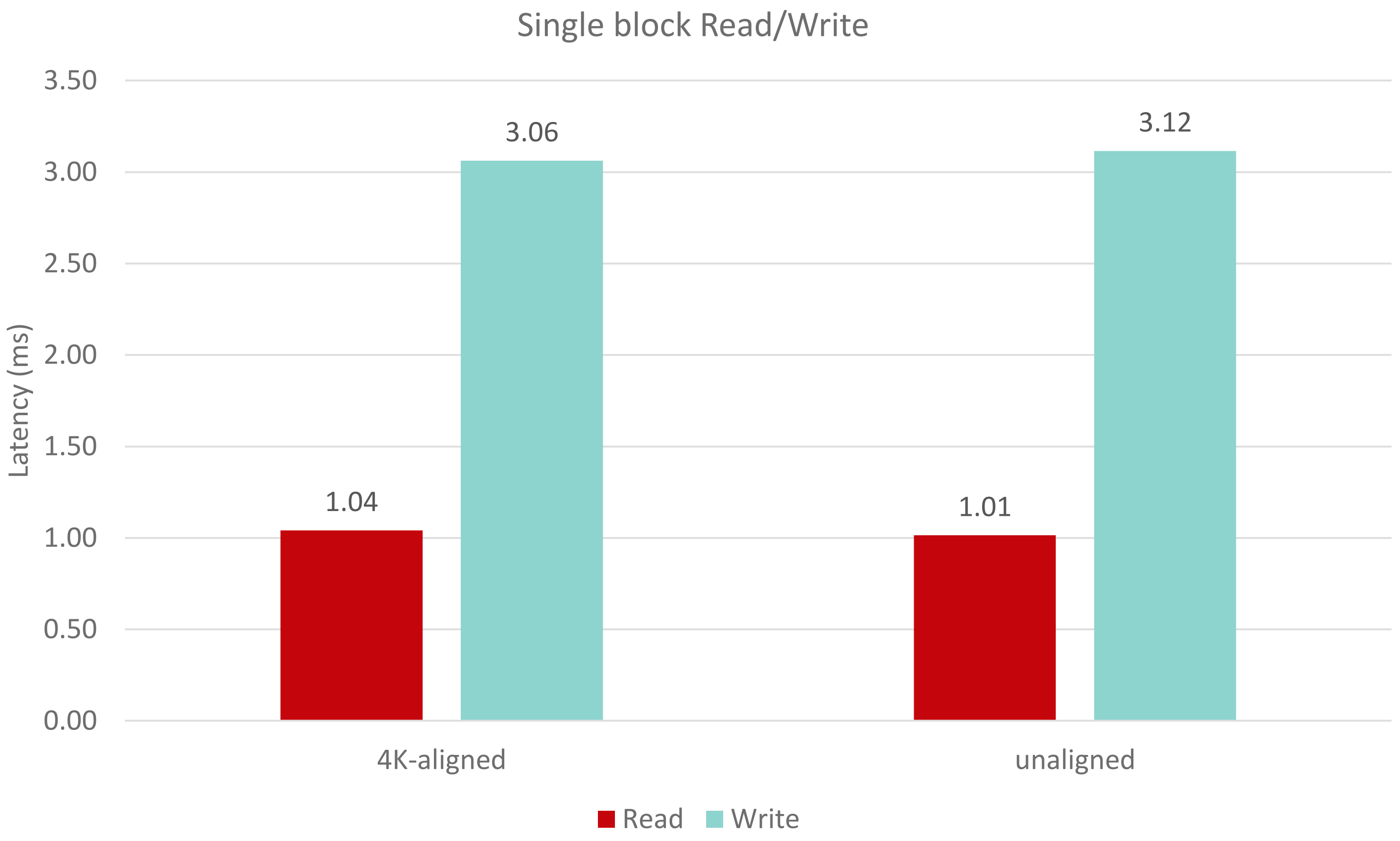
Single Block Read/Write



SQLite Select/Insert

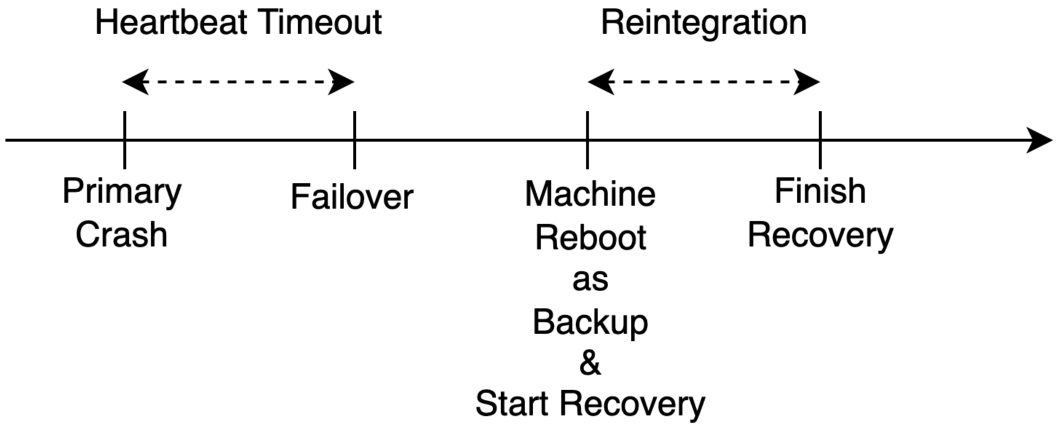
Latency - Request with Aligned Address or not

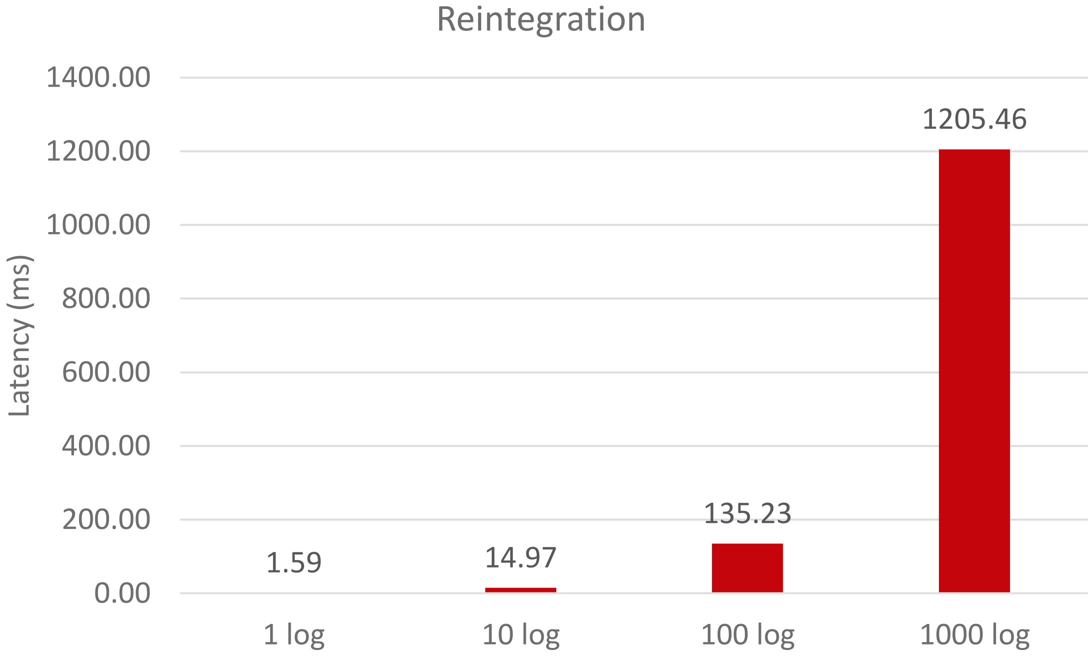
- No big difference



Recovery Time

- Heartbeat timeout – we set it to 1 second
- Reintegration time – proportional to the number of logs



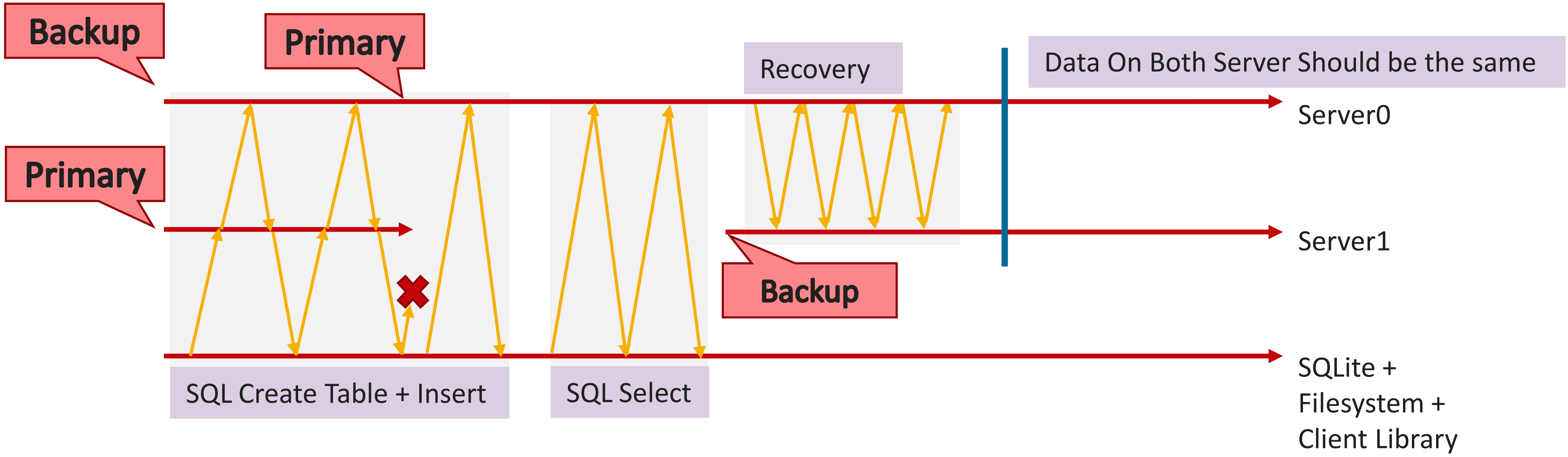


Test & Demo

- Python script to control the System
  - Launch clients and servers
  - Terminate servers at various timepoints
  - Check if output is as expected
- Test Suite
  - Simple operations with 1 or 2 servers
  - Normal FUSE operations
  - Backup die and revive
  - Backup die again during recovery
  - Primary die and revive
  - Primary die during FUSE SQLite workload
  - Network failure
  - …

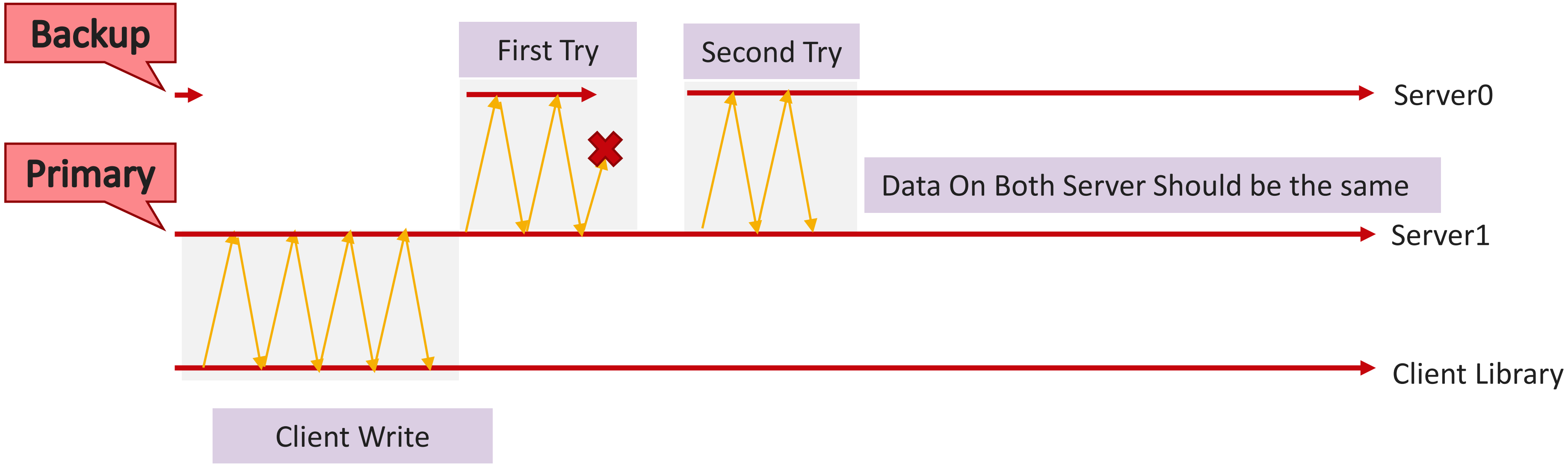
Demo Case 1: Primary fails during SQLite workload

- Goal
  - Failure should be hidden from the client
  - After the failed server restarts, data on both server should be consistent



Demo Case 2: Backup fails again during recovery

- Goal
  - In case when the backup fail again during recovery
  - Primary should remember remaining log
  - Complete the recovery when backup comes up again



Backup
Primary
First Try
Second Try
Server0
Data On Both Server Should be the same
Server1
Client Library
Client Write