Strongly Consistent Redundancy-based Utilitarian Block Store (SCRUBS)

Kalyani Unnikrishnan, Basava Kolagani, Adil Ahmed, Vyom Tayal

Goals

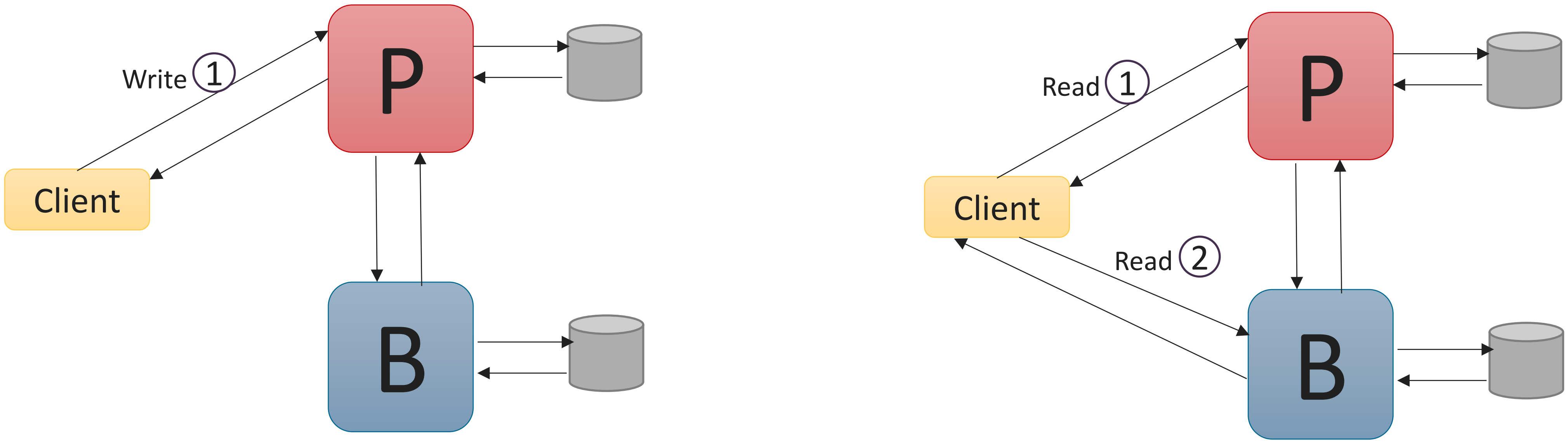
1. Provide Strong Consistency and Durability
    - Primary / Backup inspired design. Failure logs and write queue to implement consistency and ordering
2. Optimize for Performance
    - Reads are distributed across the primary and backup
    - Extraneous network calls are reduced using “piggybacking”
    - Writes to primary and backup are performed concurrently
3. Primary-backup selection mechanism robust to human error / server failure
    - A server “handshake” protocol assigns the primary / backup that is robust to crashes and concurrent operations
4. Don’t trust the client library!
    - Servers achieve consistency after crashes despite undefined client behavior
5. Failures are hidden from client
    - Client API abstracts the details from the user

Assumptions

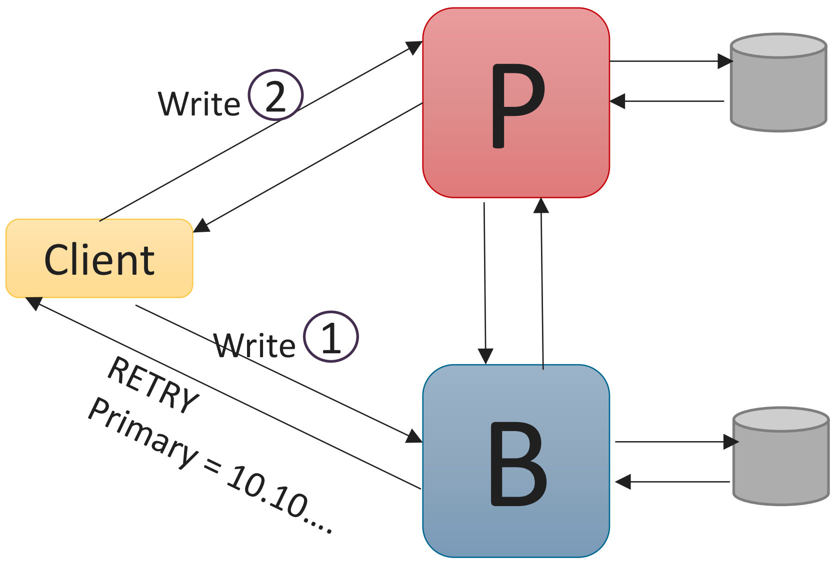
- The client library knows the IPs of the servers. No master or view service is required.
- Both servers will not fail at the same time.
- A server will not fail during recovery.

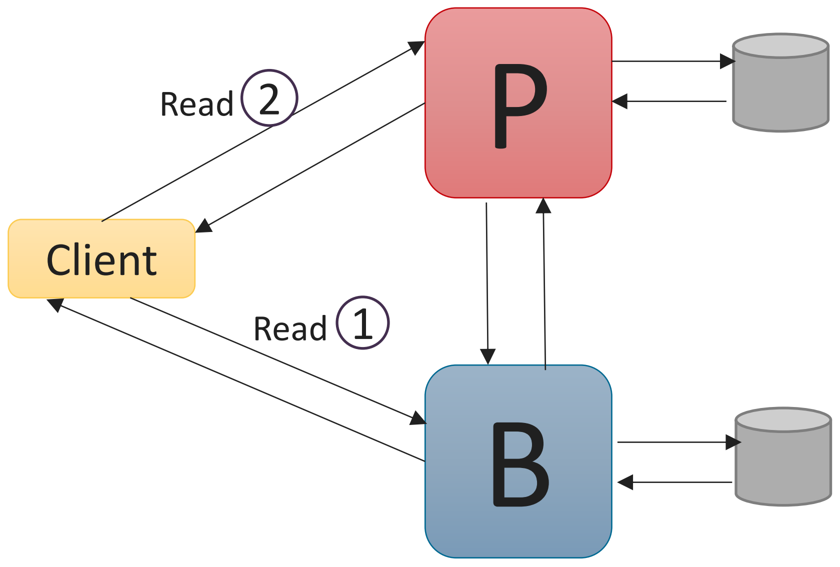
Design – High Level

- One server acts as primary and one as a replica.
- Writes are served by the primary.
- Reads are served by both servers
- Backup uses heartbeats to detect if primary is up.

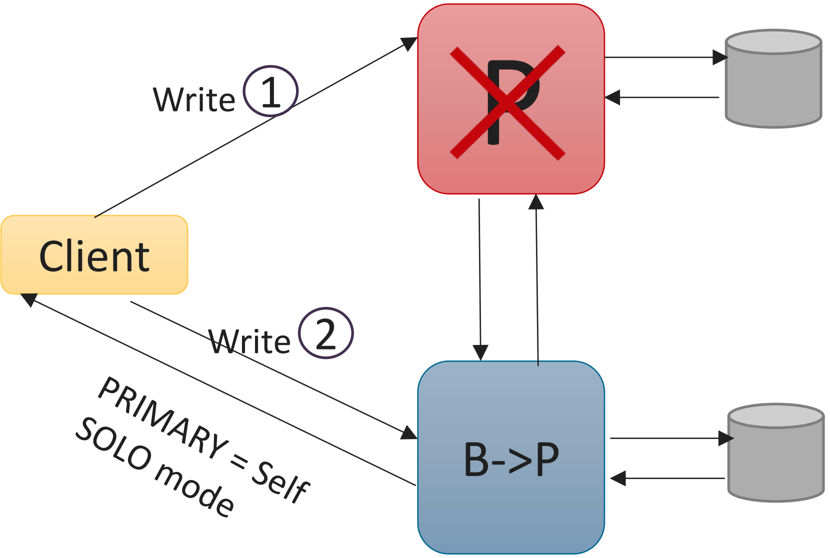


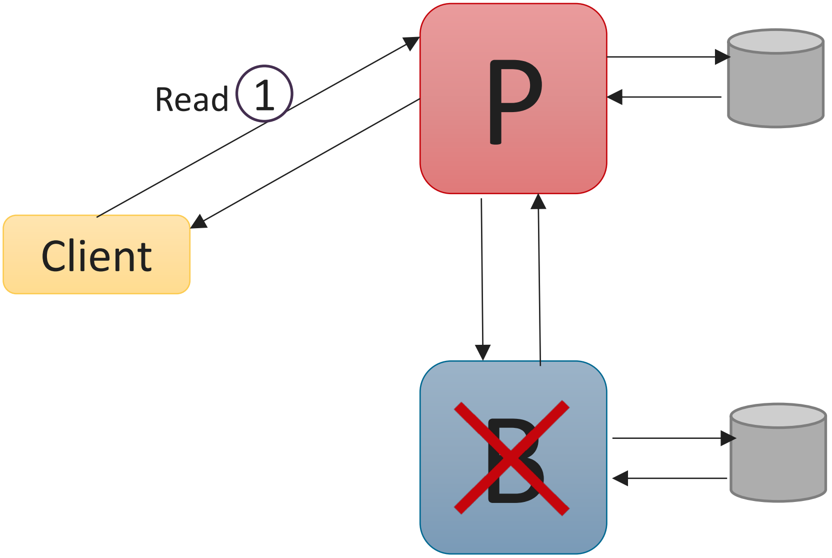
Read/Writes to Backup





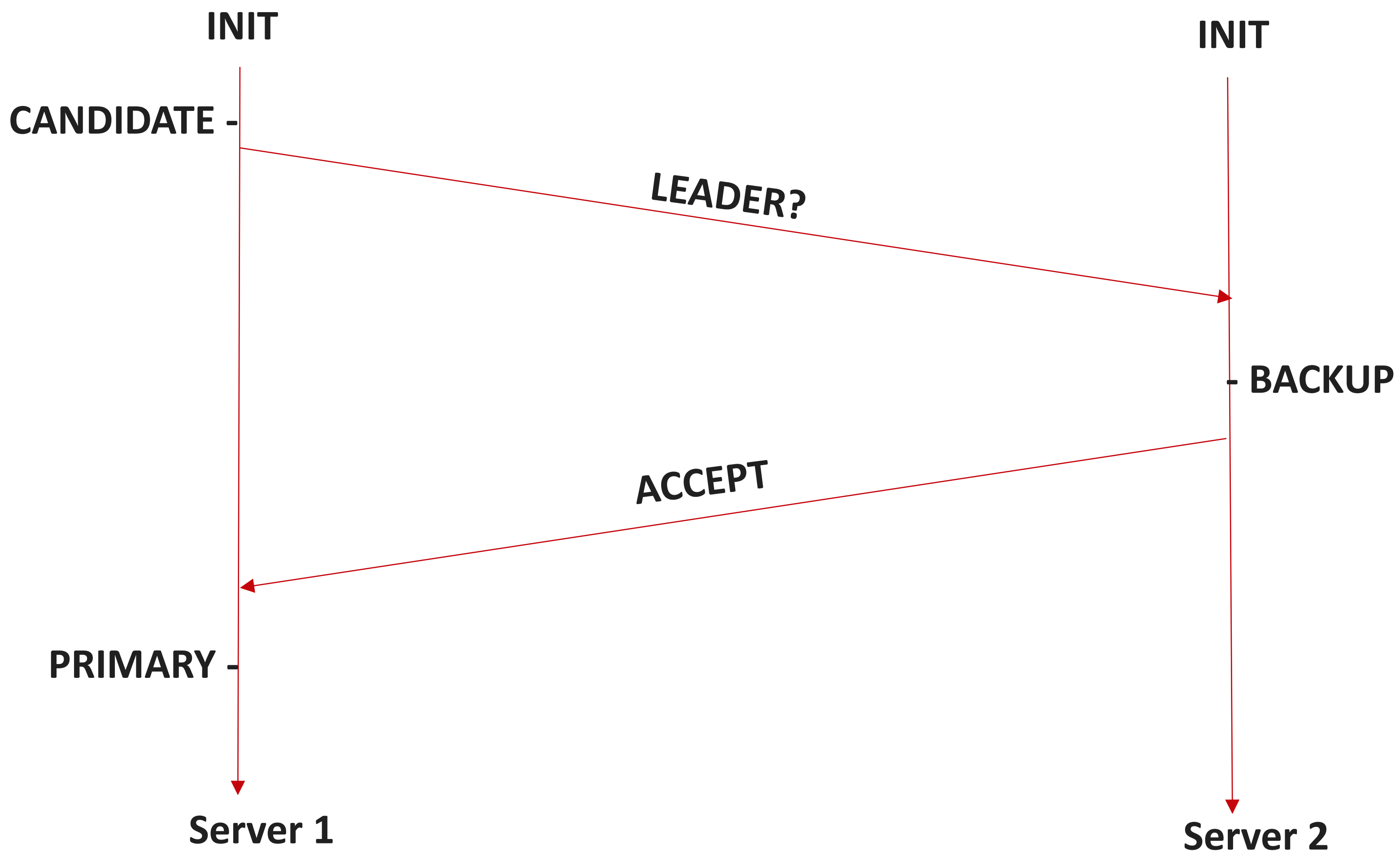
Recovery


Write 1
Client
Write 2
PRIMARY = Self
SOLO mode
P
B->P

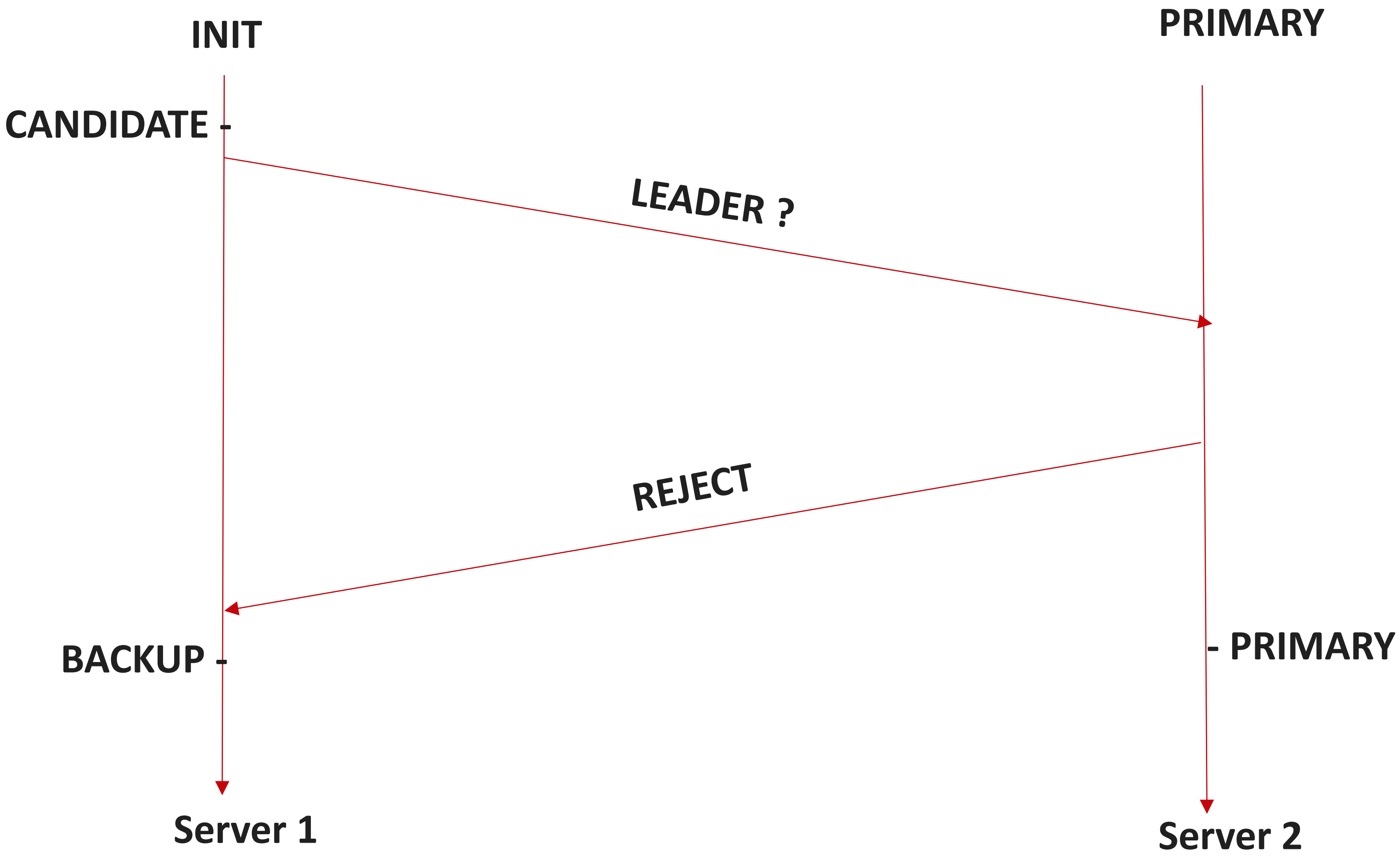

Read 1
Client
P
B

Server Handshake

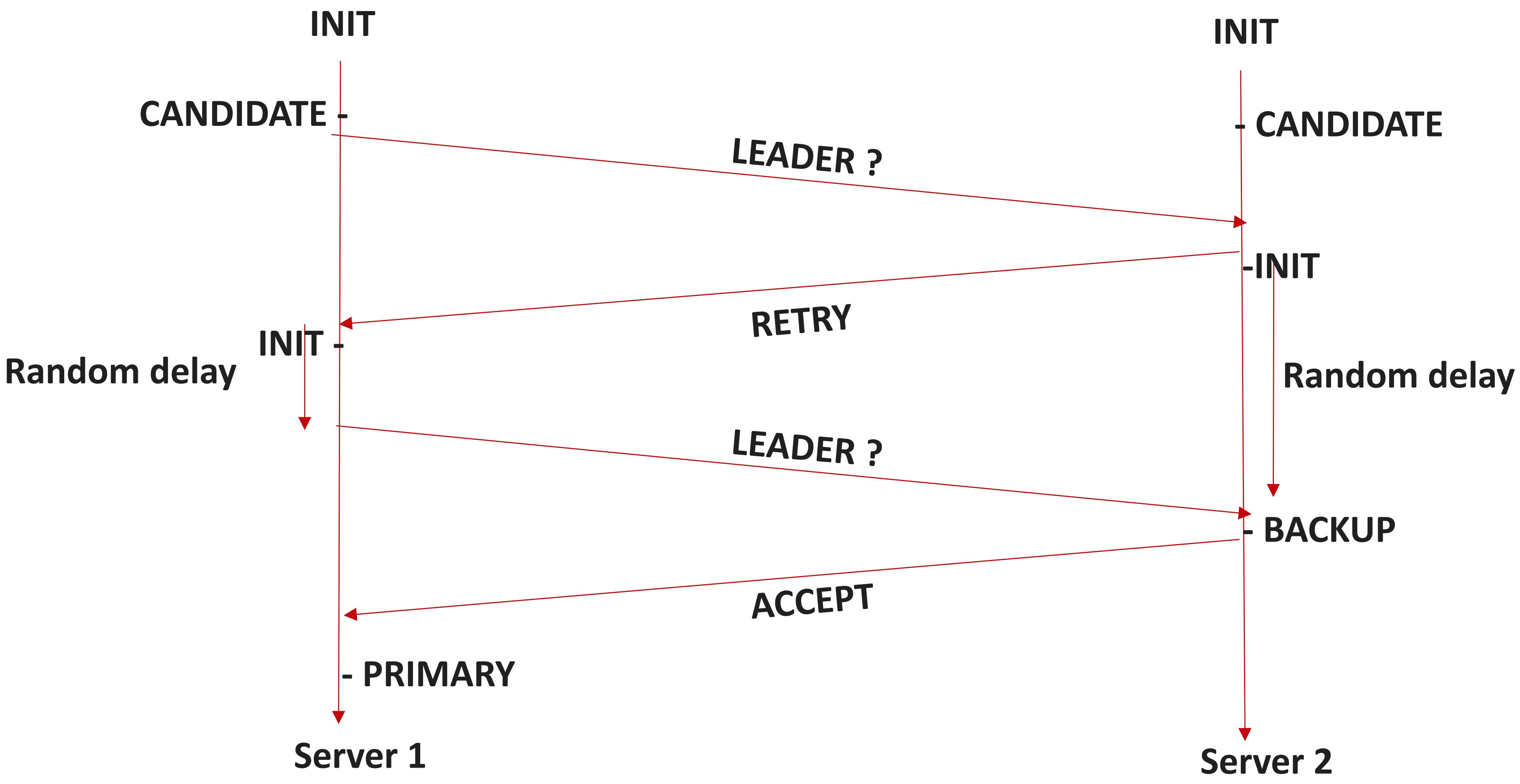
Election States: ”INIT", ”CANDIDATE", ”PRIMARY", ”BACKUP"



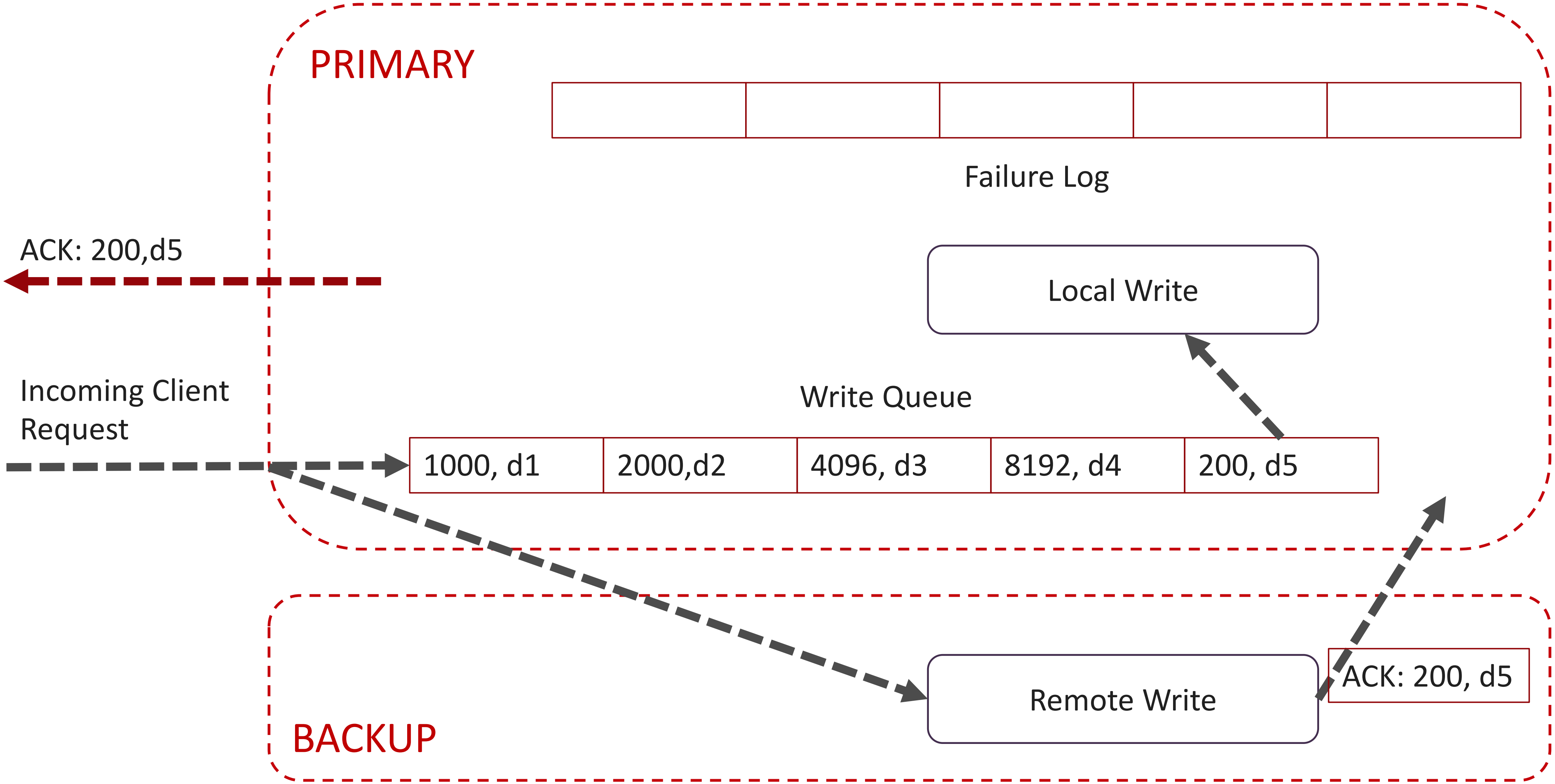
Server Handshake



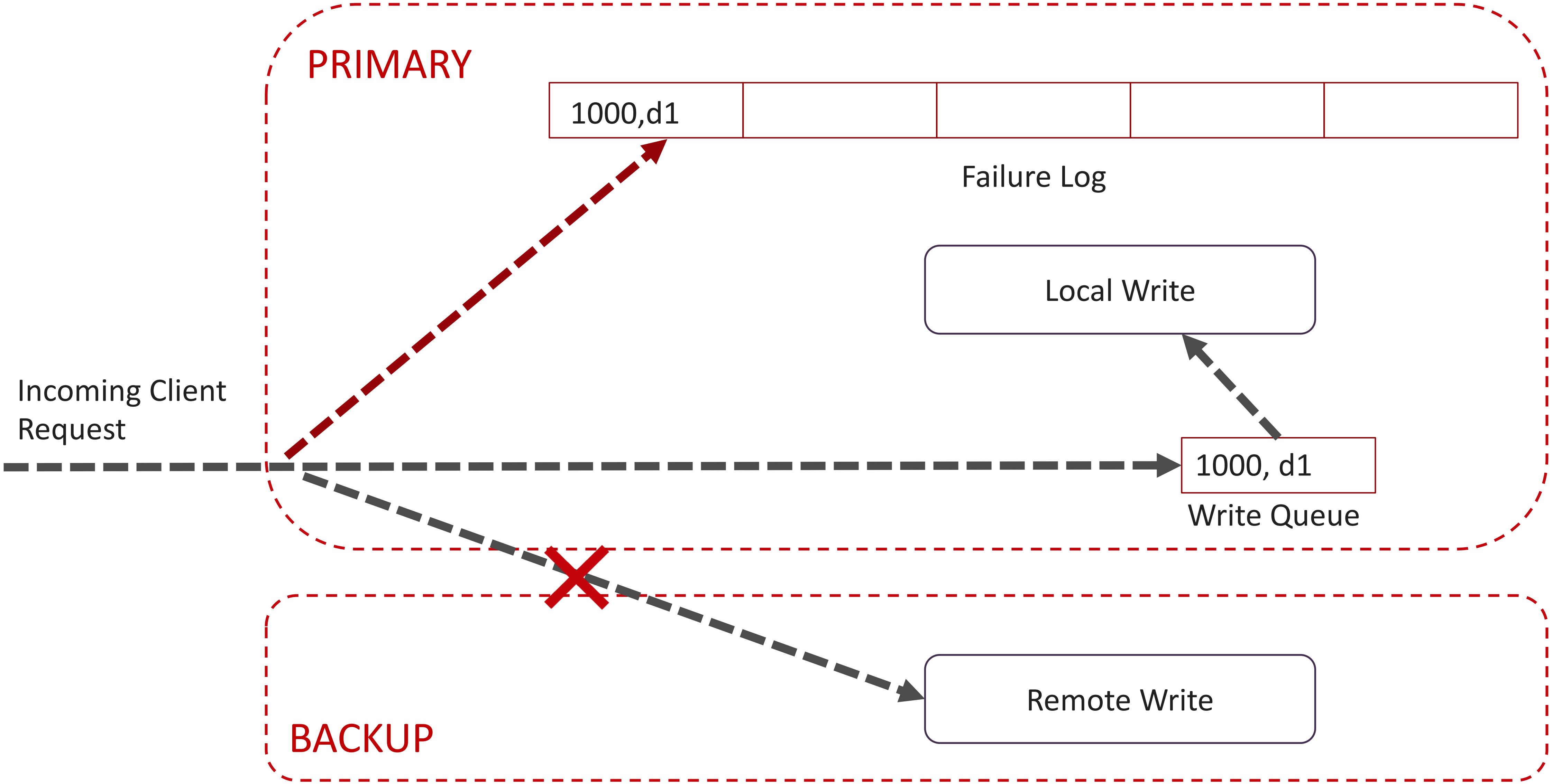
Server Handshake



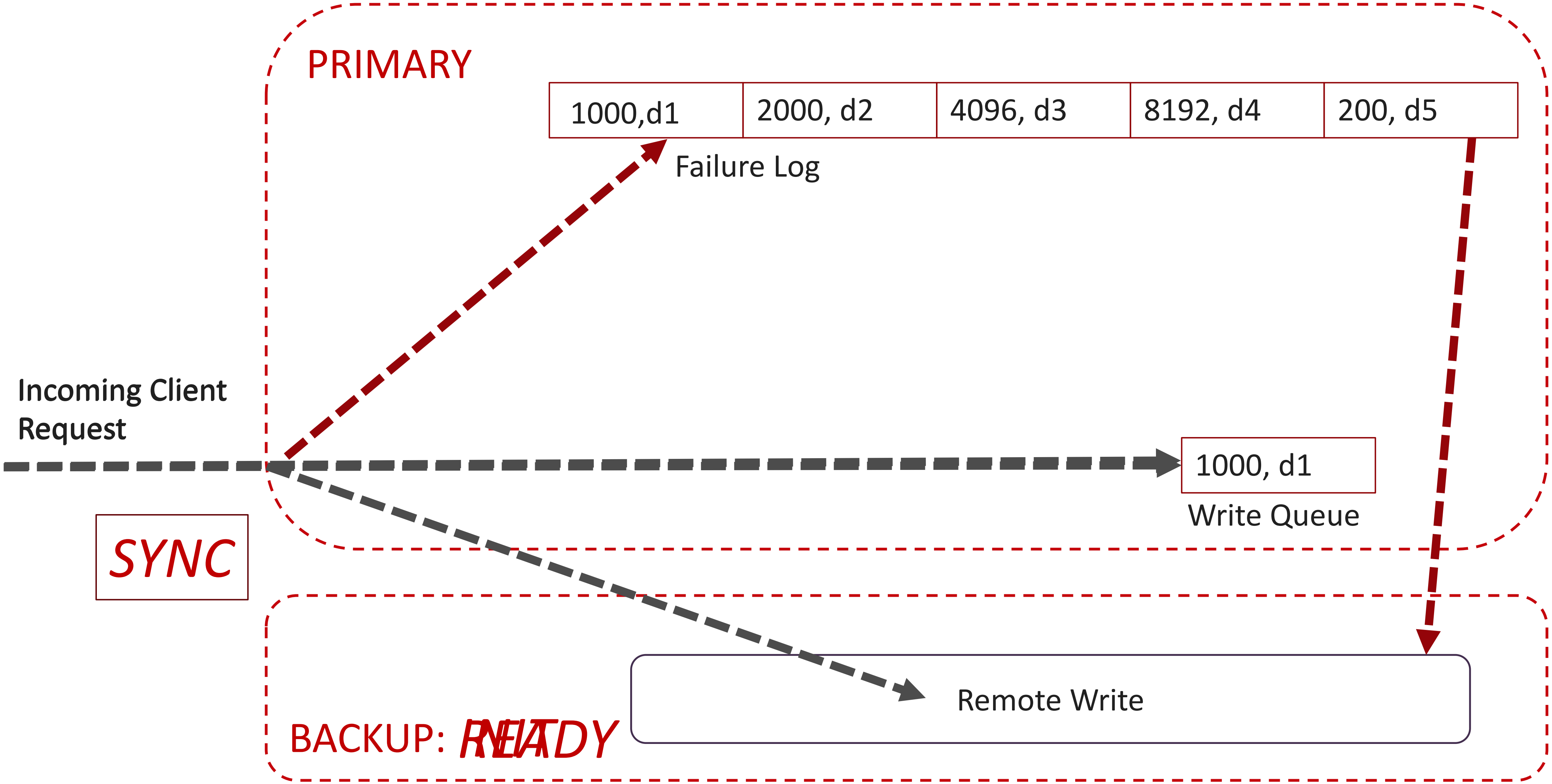
Writes at Server



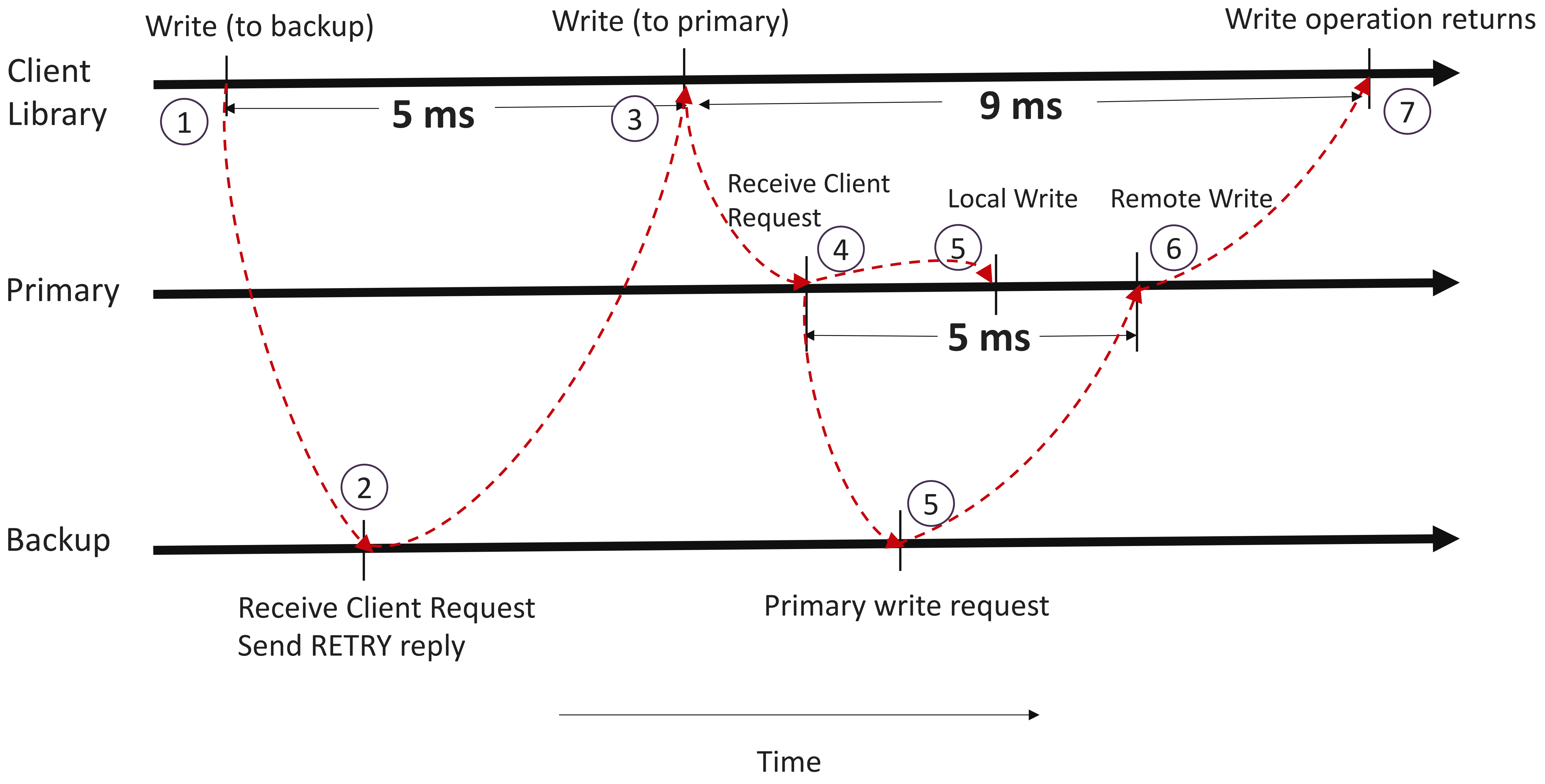
Writes at Server (when one server fails)



Recovery


PRIMARY
1000,d1
2000, d2
4096, d3
8192, d4
200, d5
Failure Log
Incoming Client Request
1000, d1
Write Queue
SYNC
Remote Write
BACKUP: READY

Timeline of Operations



Consistency matrix

All fault injection points vs effect (consistent / not consistent)

Client	Primary					Backup				State
	Hand-shake	Read	Local Write	Remote Write	Sync	Hand-shake	Read	Local Write	Sync	Consistent/ Inconsistent
-	X	-	-	-	-	✓	✓	✓	✓	✓
Read	✓	X	-	-	-	✓	✓	✓	✓	✓
Write	✓	✓	X	✓	✓	✓	✓	✓	✓	✓
Write	✓	✓	✓	X	✓	✓	✓	✓	✓	✓
Write	✓	✓	✓	✓	✓	✓	✓	X	✓	✓
-	✓	✓	✓	✓	✓	✓	✓	✓	X	X
-	✓	✓	✓	✓	X	✓	✓	✓	-	X

Demo and Evaluation

Experiment Set-up:

4 nodes on CloudLab.

Each node comprised 2 Intel Xeon Silver 4114 10-core CPUs at 2.20 Ghz with 192GB of RAM

Link bandwidth on each node was capped at 10 MBps, to study behavior during contention

Demo 1: Normal Operation

demo.py 1

write(0); read(0);

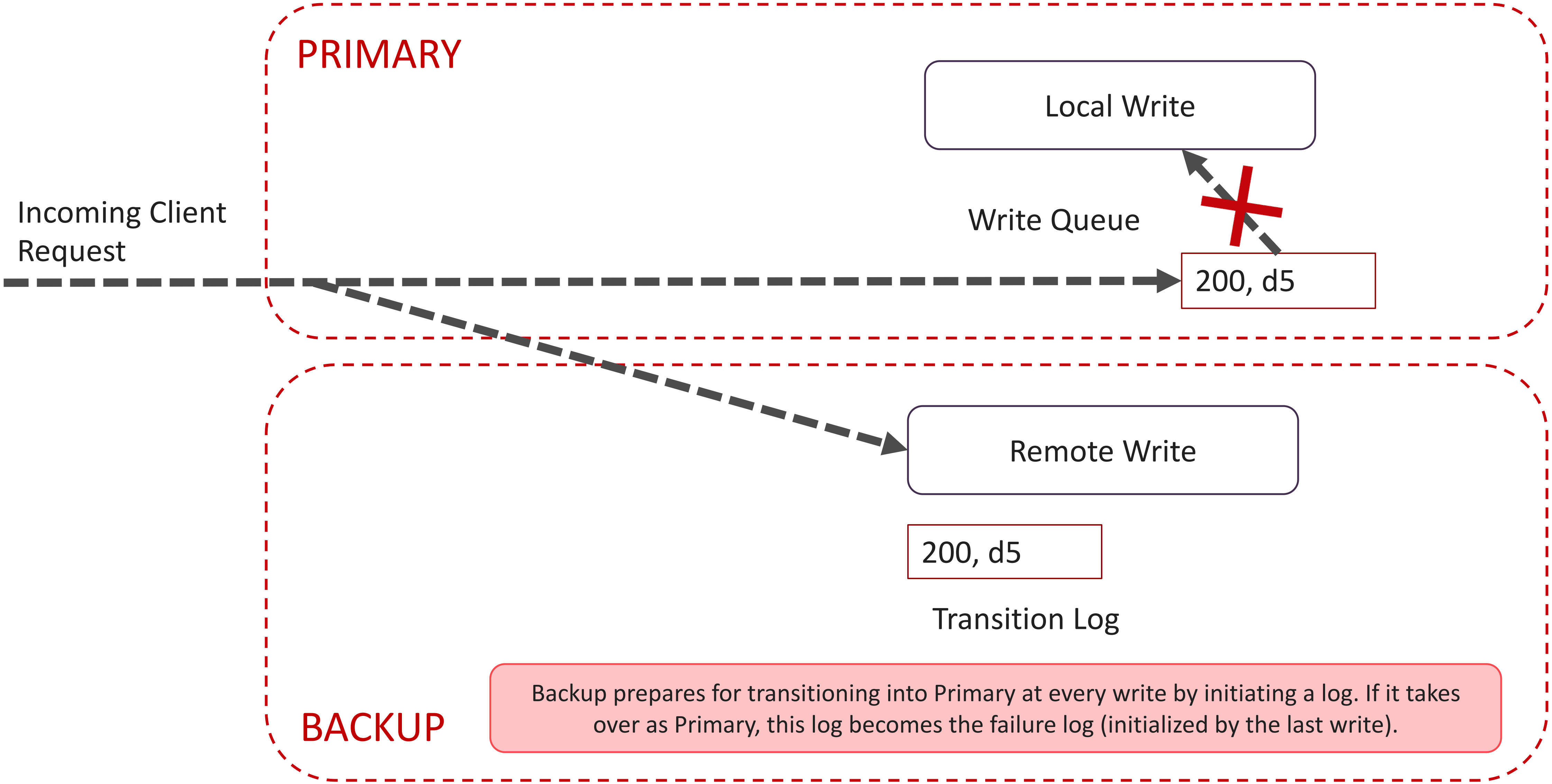
write(1); read(1);

demo.py 2

read(0); read(1);

https://drive.google.com/file/d/1LUpZla08yXFE213LeiyFmZYLmuF_N-G6/view?usp=sharing

Demo 2: Primary crashes during local write but after remote write

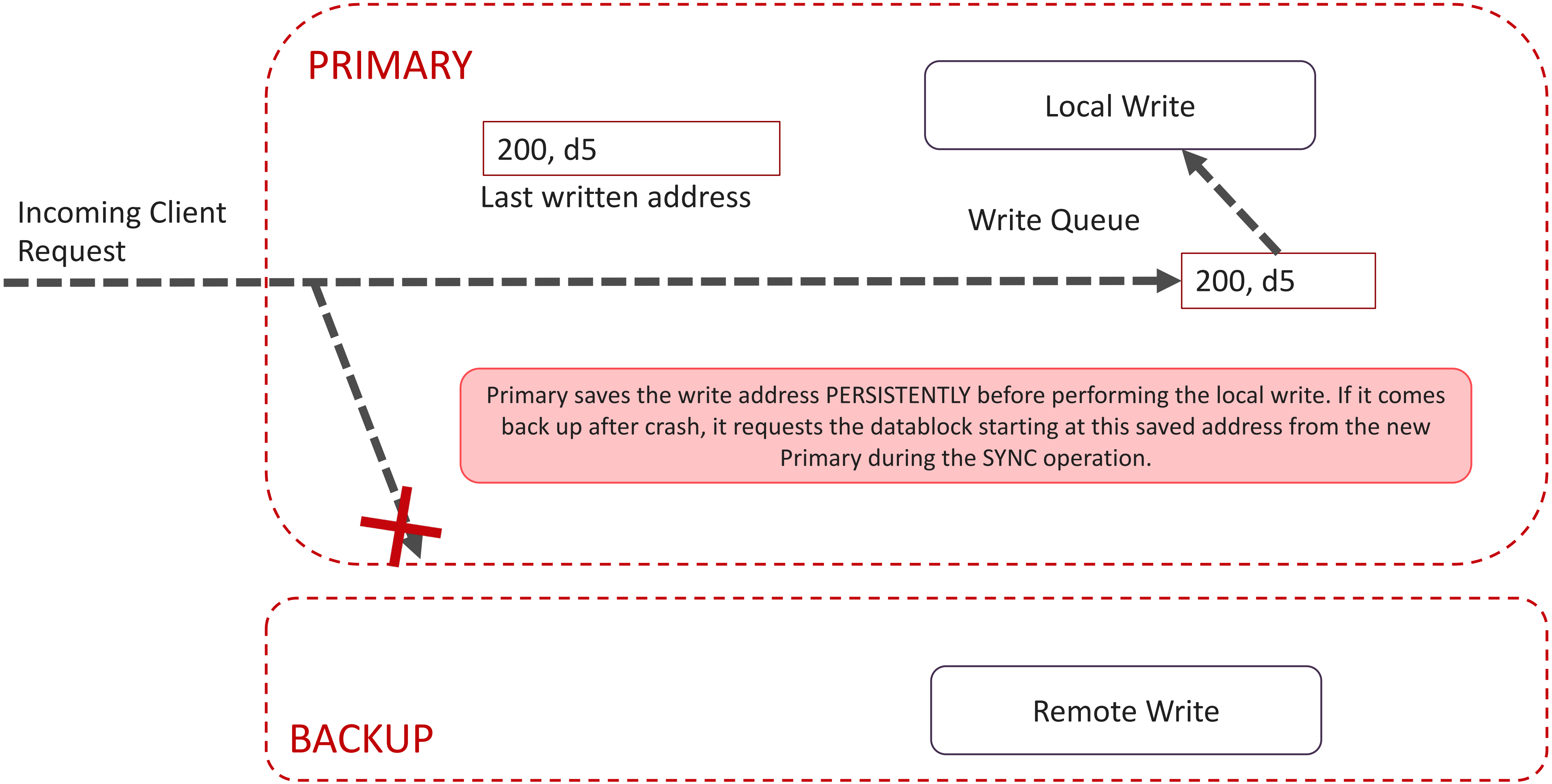


Demo 2

Crash PRIMARY_CRASH_BEFORE_LOCAL_WRITE_AFTER_REMOTE injected:
Local write blocked
Remote write request sent
Kill

https://drive.google.com/file/d/1O63ivN9zLAT6F4x3NhsXL45TBituSbk8/view?usp=sharing

Demo 3: Primary crashes during remote write but after local write



Demo 3

Crash PRIMARY_CRASH_AFTER_LOCAL_WRITE_BEFORE_REMOTE injected:
Remote write blocked
Local write thread kills server after write

https://drive.google.com/file/d/1EqGNdtLUs8e0bcLGyUKzfWKg7uEgLW4t/view?usp=sharing

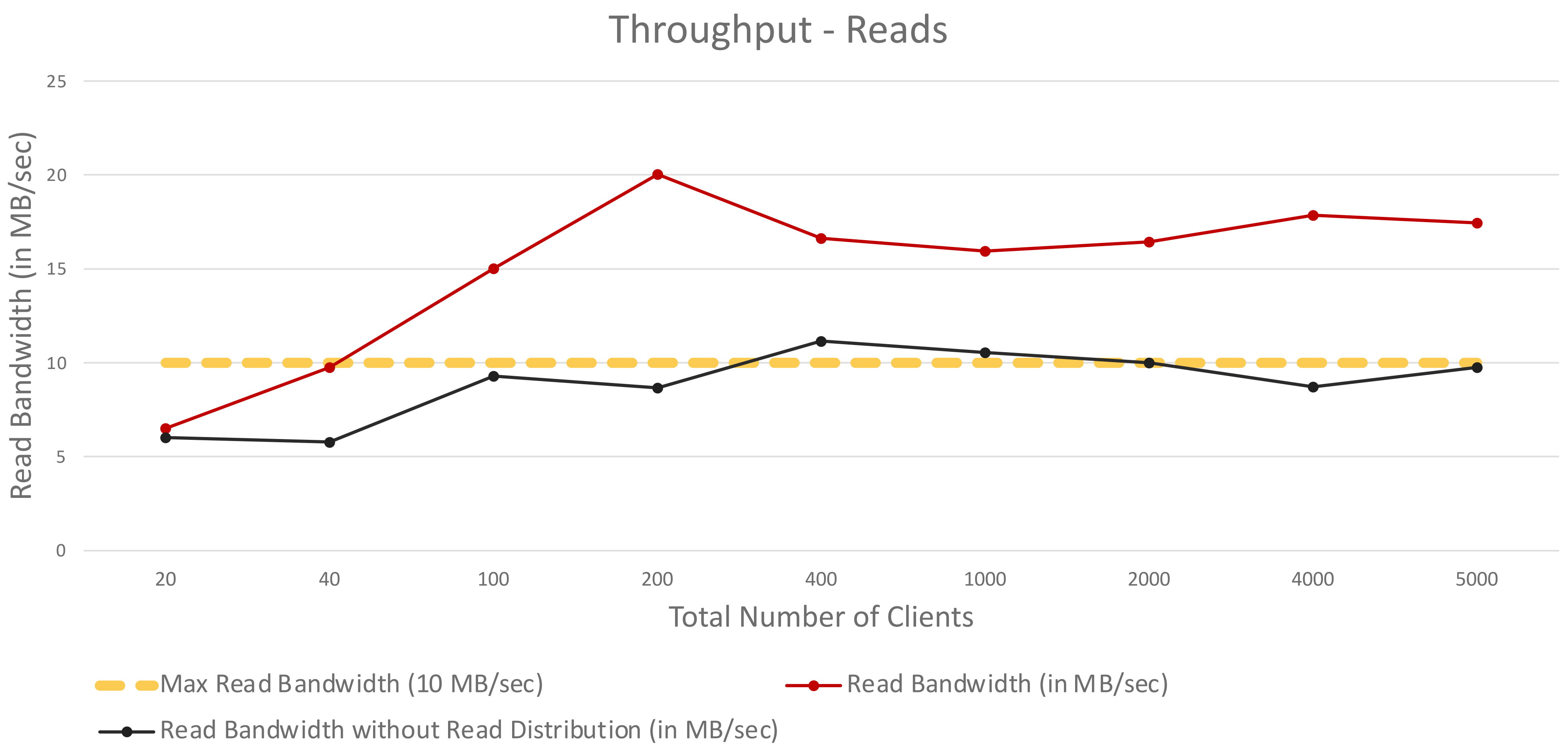
Demo 4: Backup crashes during a remote write from the primary.

https://drive.google.com/file/d/12_Q8KQjq7KUmvtE-e2nAj9u-7PjoR5vS/view?usp=sharing

Throughput – Write Bandwidth for multiple clients

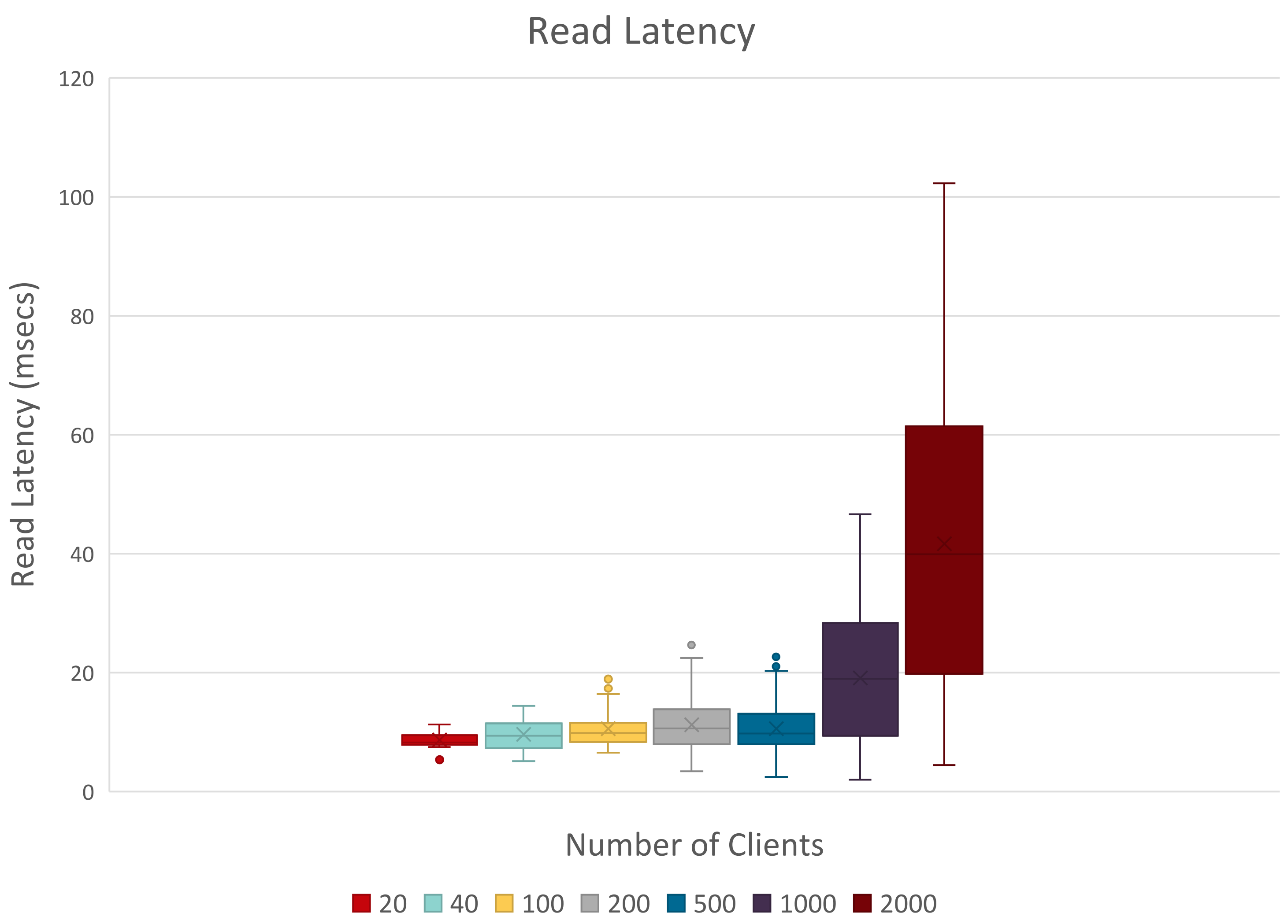
Number of Clients	Total time of writes (1 write per client)	Bandwidth (MB/sec)
1000	1.91	2.1
2000	3.39	2.4

Throughput – Reads [link b/w set to 10MBps per server]

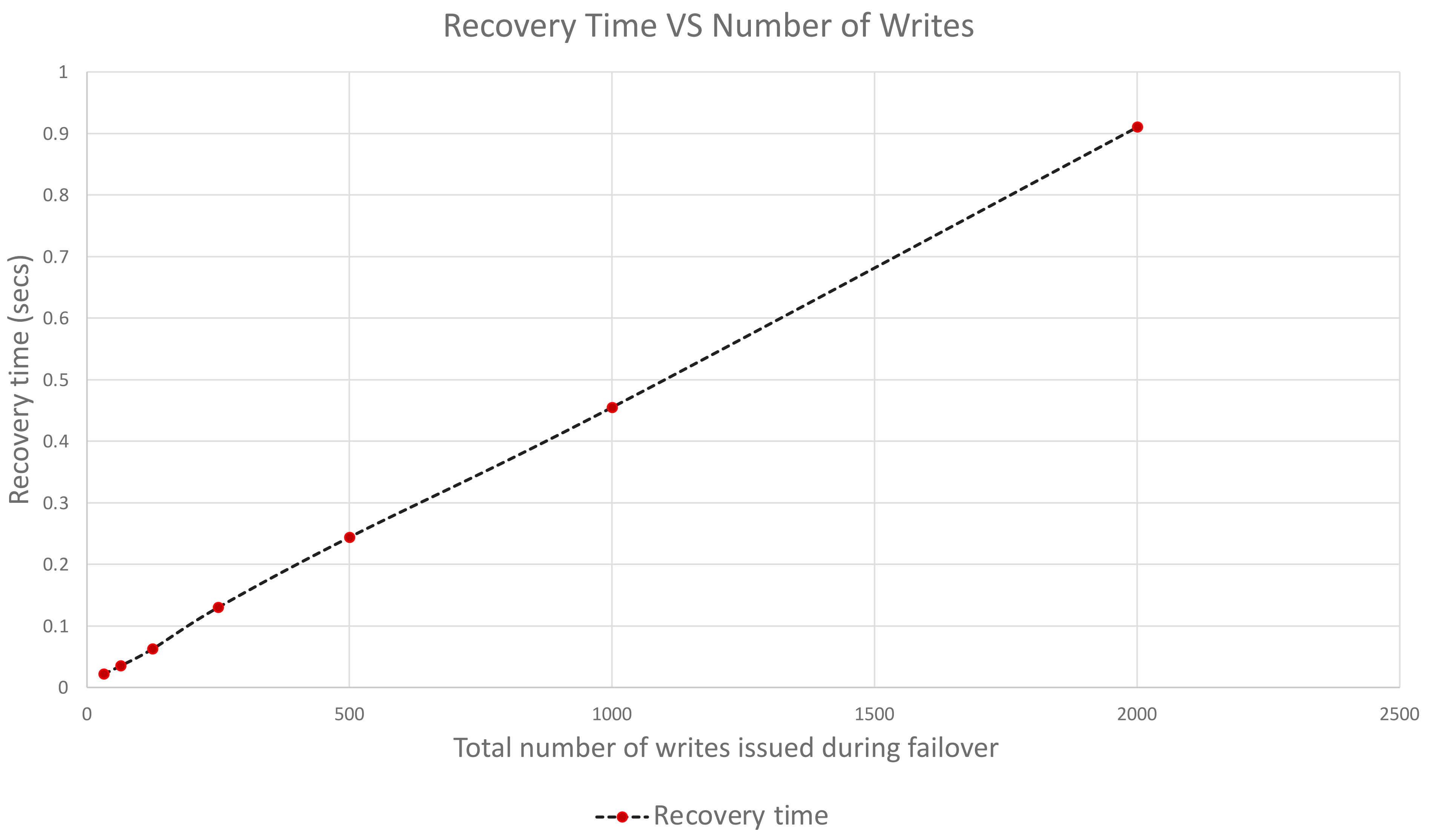


Read Latency [link b/w set to 10MBps per server]

All clients run concurrently. Each performs a single read operation.



Performance: Recovery Times



Performance: SOLO vs DUAL modes

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON