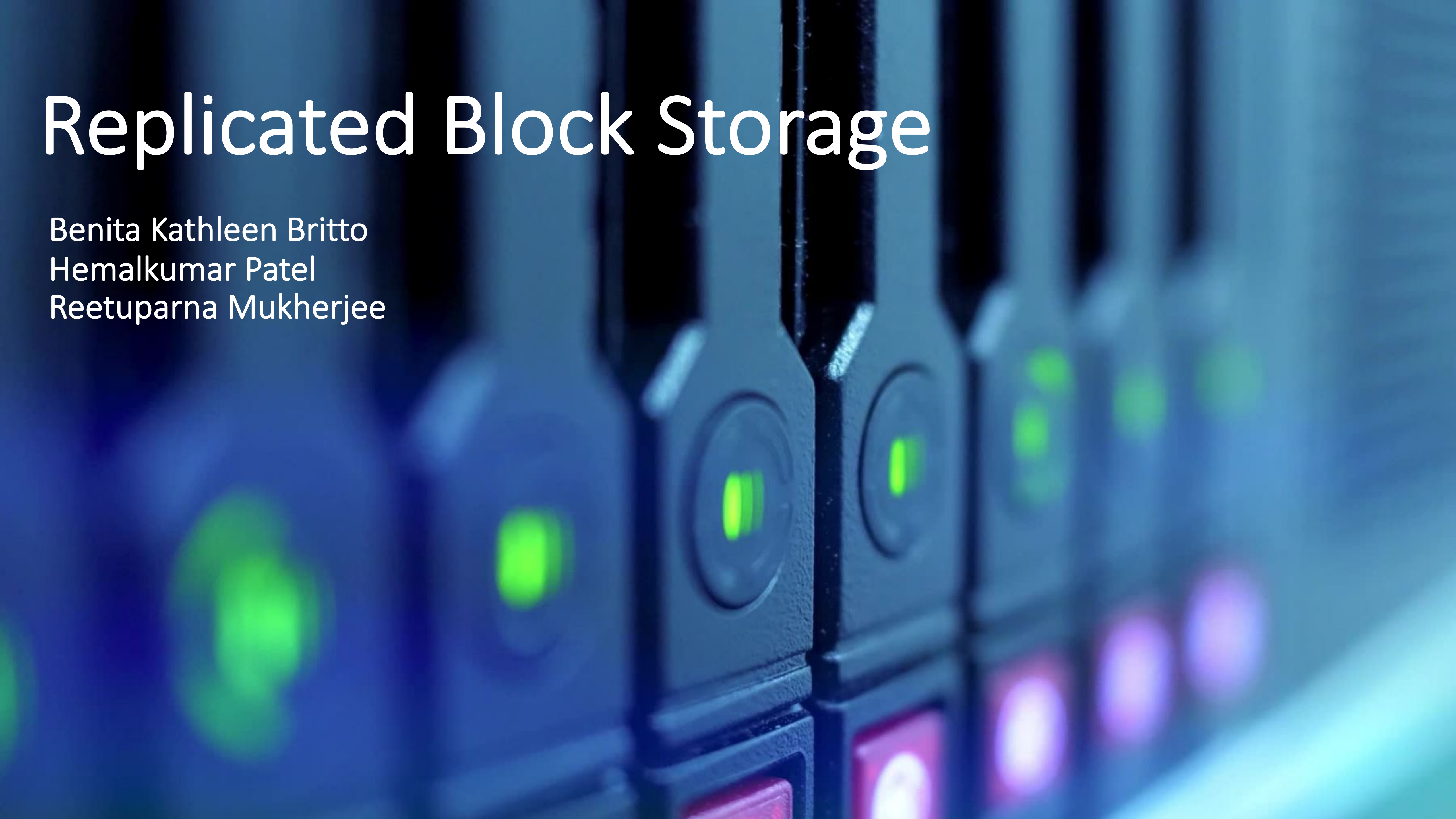
Replicated Block Storage

Benita Kathleen Britto
Hemalkumar Patel
Reetuparna Mukherjee



Key Assumptions

Non-Byzantine Failures

Stable Storage – No disk corruption

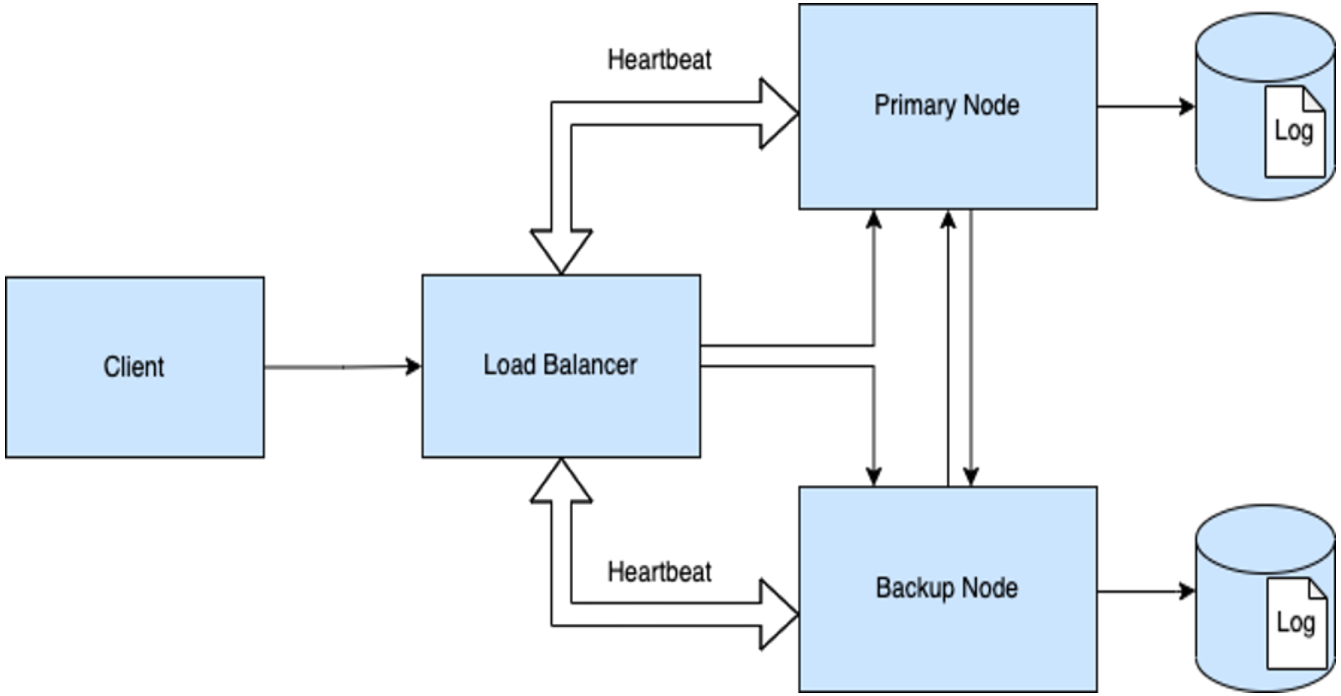
Stable Network – No packet drop

- No response means node has crashed

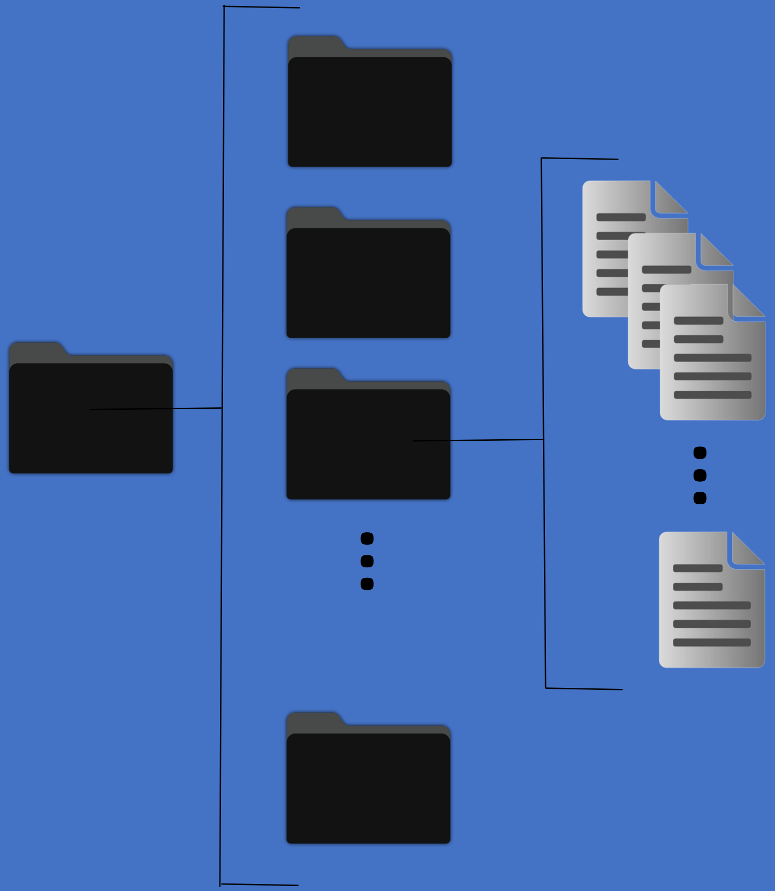
One Node Always Up

- Backup becomes primary
- No failures during crash recovery

System Architecture



Storage Structure



Block Storage Size: 1GB

1024 directories, each having 256 files of 4KB size

Easy scale up/down (not included in this project)

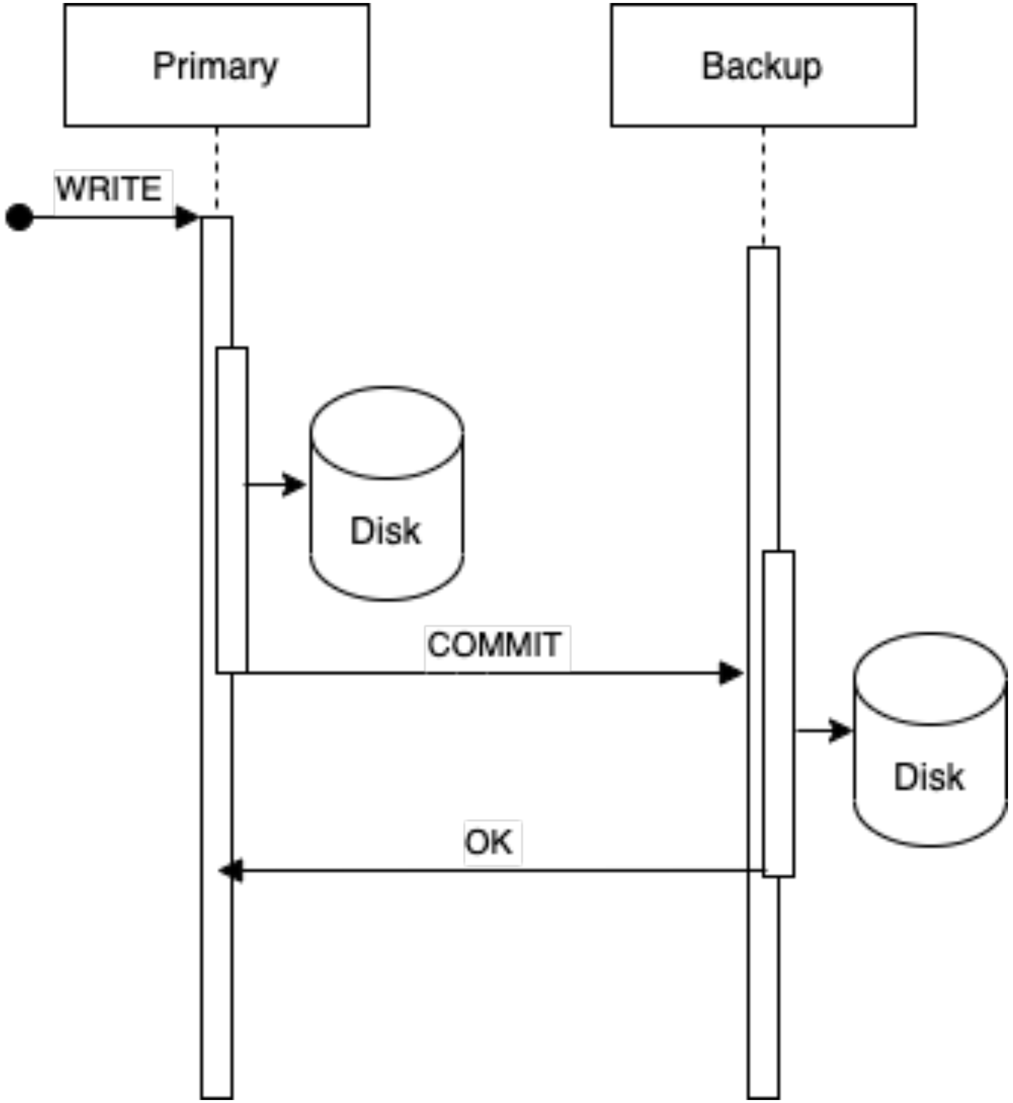
Can partition data across storage (not included in this project)

Replication Strategy

Approach #1: Primary Commits First, Backup Second

On write():

1. Primary writes (commits) data first
2. Primary contacts backup to write
3. Backup writes data



Approach #1: Primary Commits First, Backup Second

Problem:

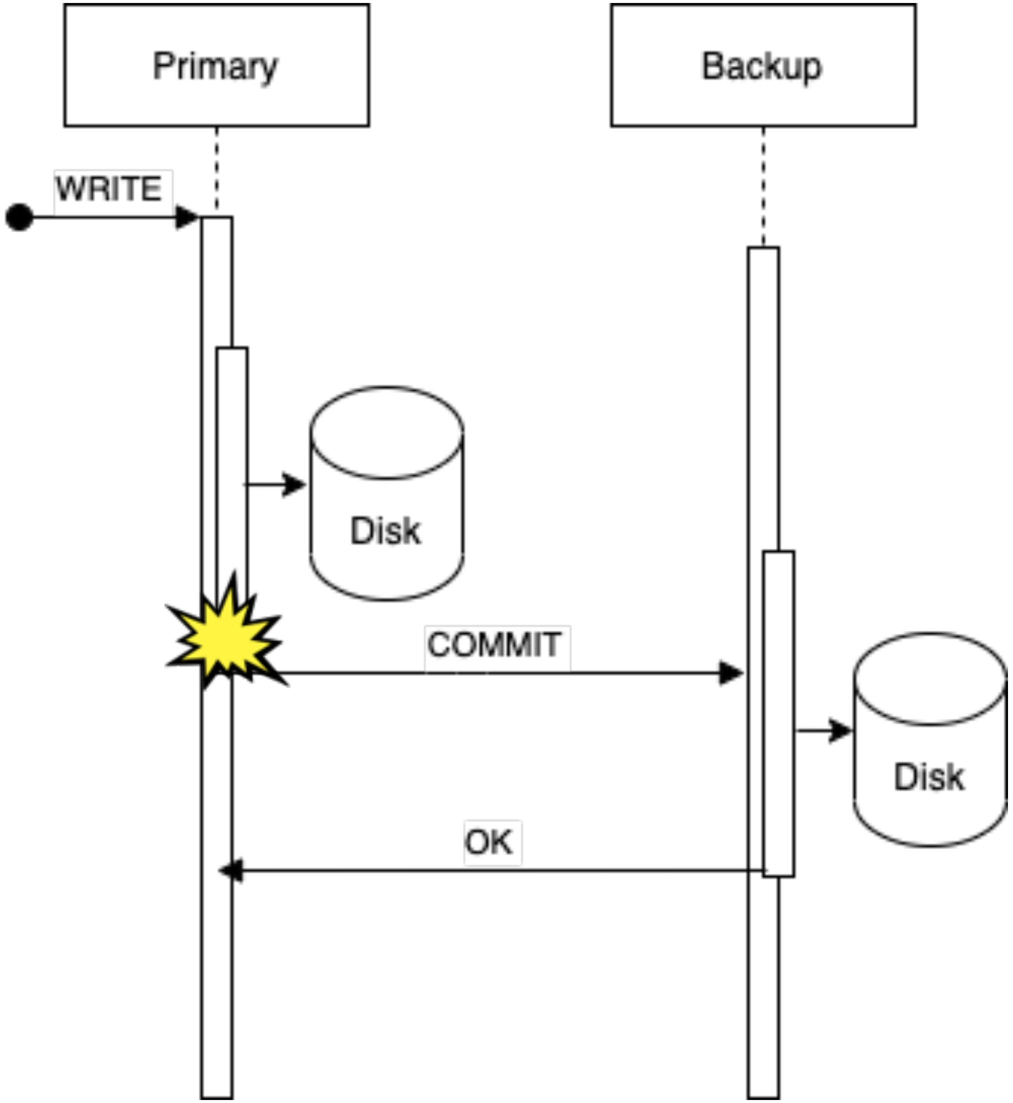
What happens if the primary crashes before contacting the backup?

- Backup has no knowledge of the request

- Client will see the old data and assume that the request failed

Solution:

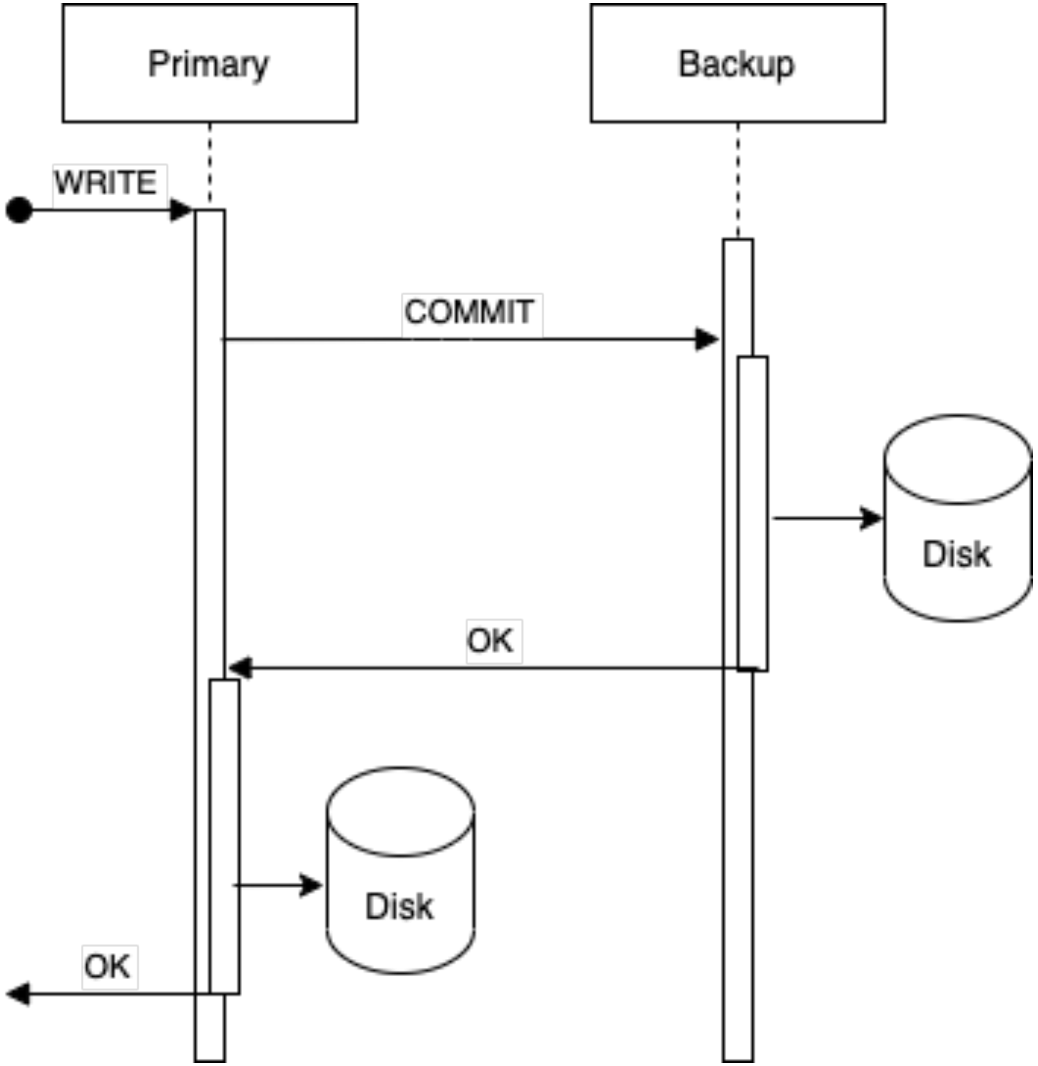
When primary reboots, it must reconcile with backup for all requests – Expensive!



Approach #2: Backup Commits First, Primary Second

On write():

1. Primary contacts backup to write
2. Backup writes data
3. Primary writes (commits) data on receiving an OK



Approach #2: Backup Commits First, Primary Second

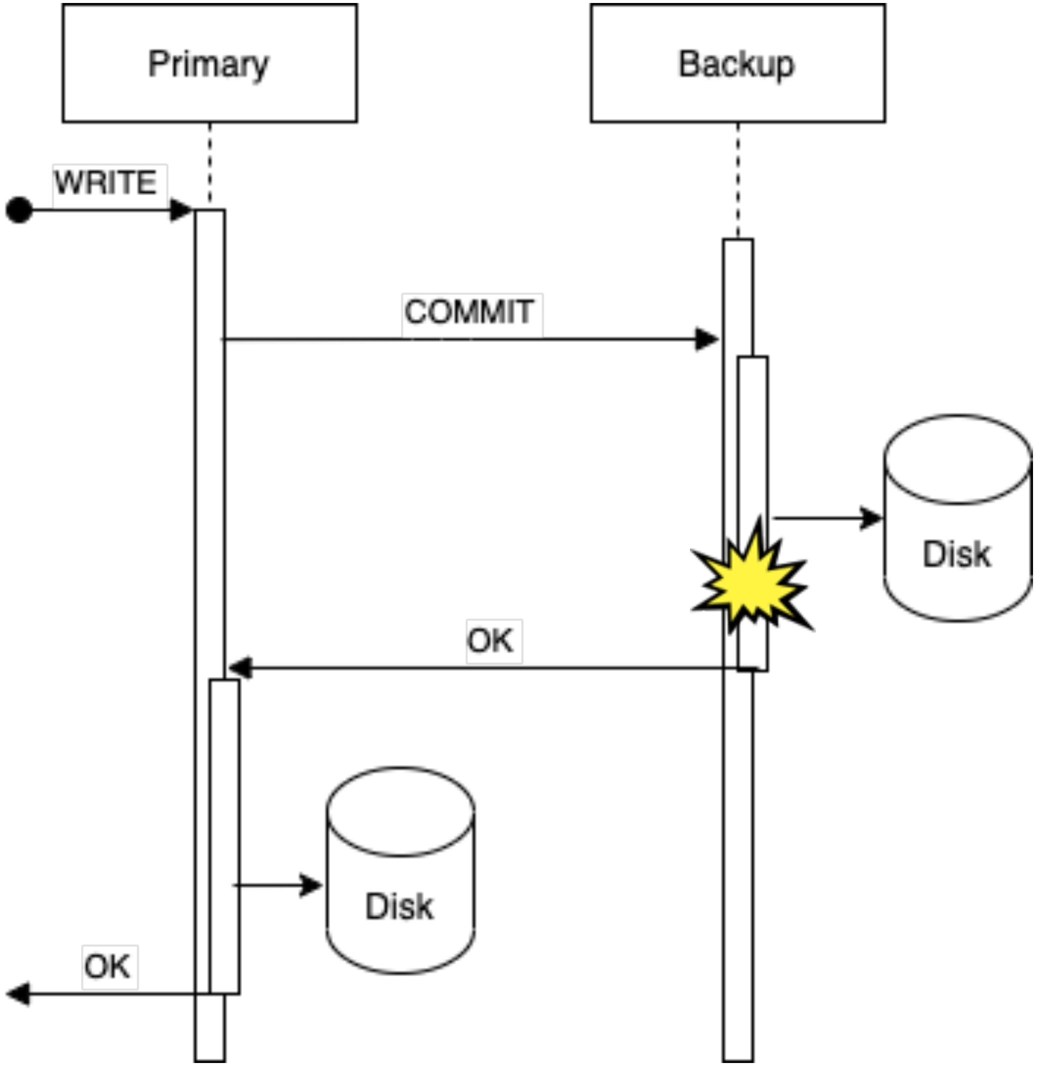
Problem:

What happens if backup crashes before responding to the primary?

- Primary can commit/abort the txn

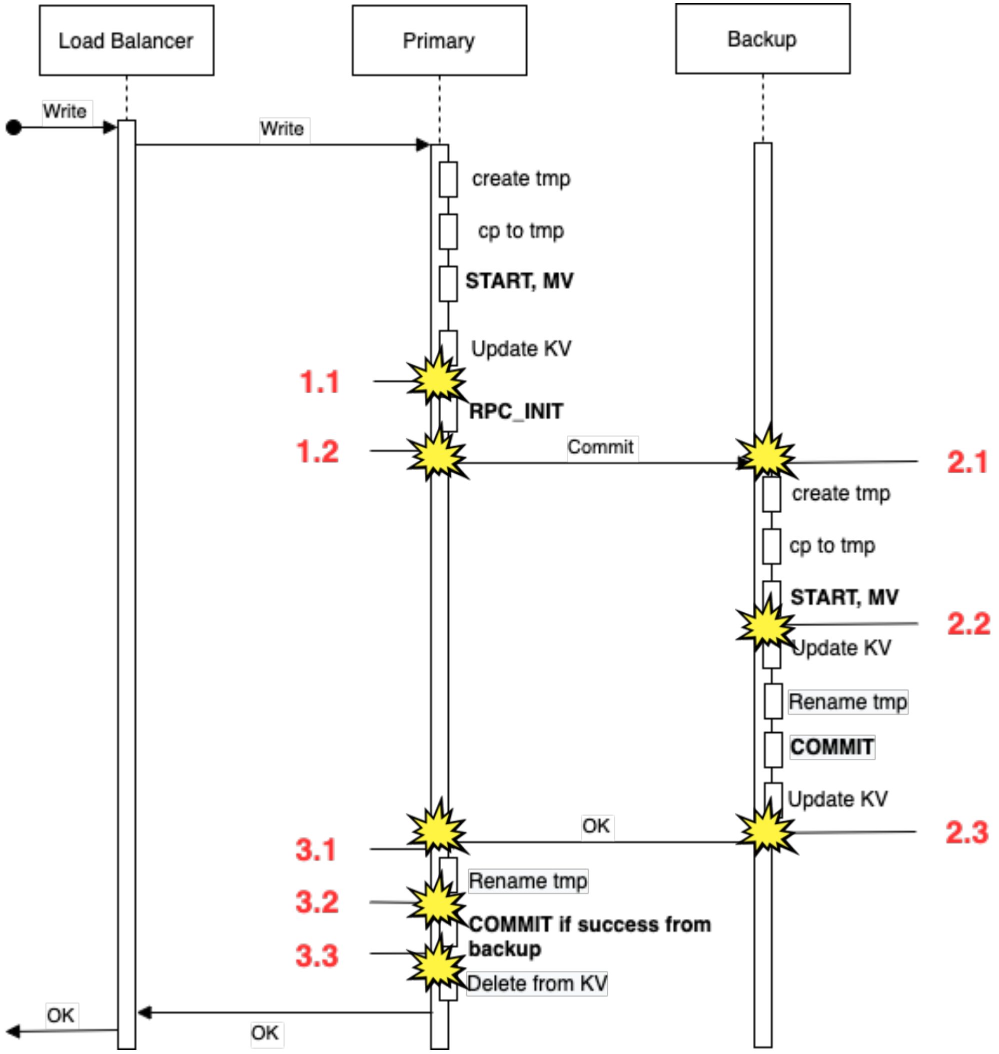
Solution:

When backup reboots,
it must reconcile with primary
for all txns – Expensive!



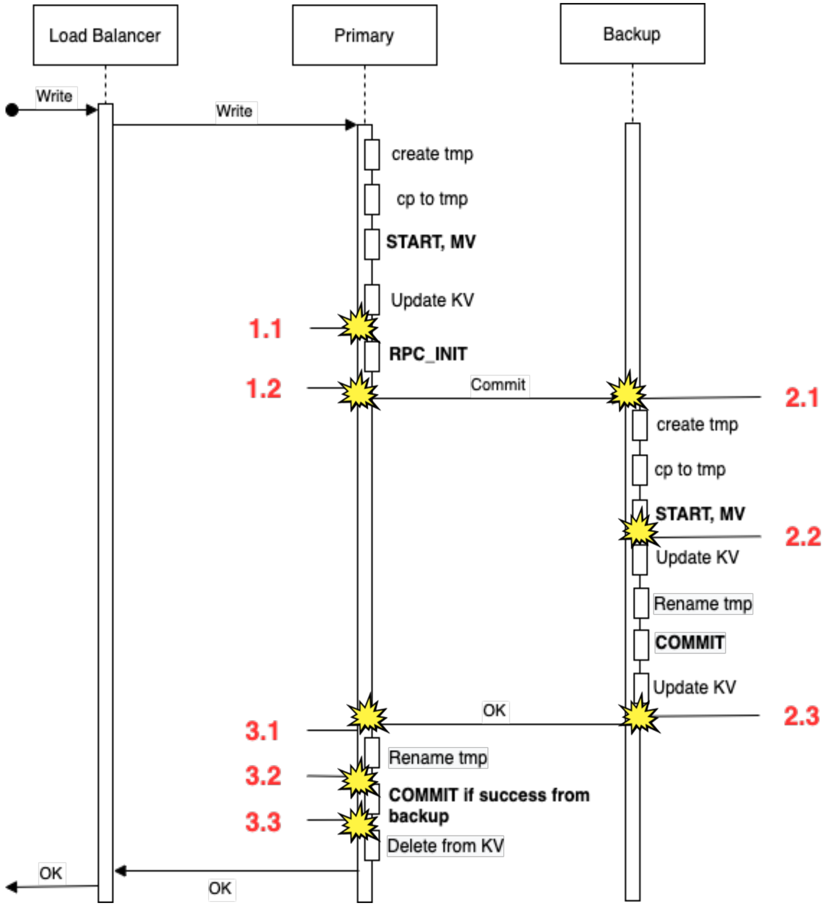
Approach #3 (Our Design): Primary Prepares, Backup Commits, Primary Commits

Crash	Primary	Backup	Client
1.1	Reconcile the txn status Fetches all pending txns from primary on reboot	Assumes Primary role	Retry
1.2			Retry



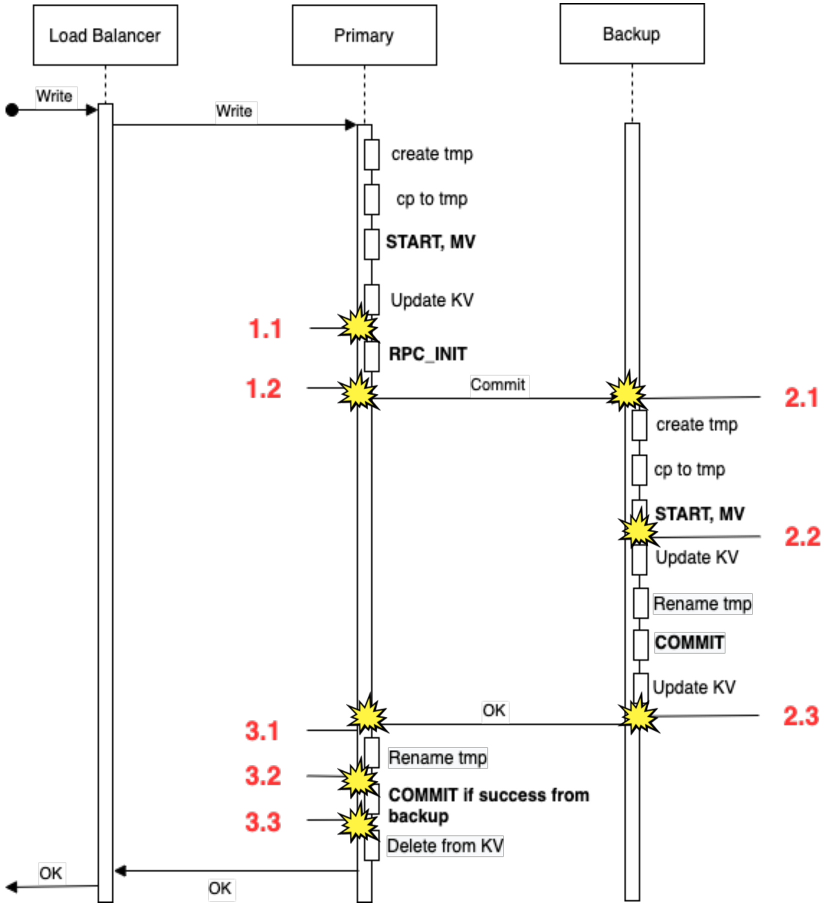
Approach #3 (Our Design): Primary Prepares, Backup Commits, Primary Commits

Crash	Primary	Backup	Client
2.1	Logs replication pending Keep txn in memory Commits txn	GetState(txnId) from primary on reboot and applies changes Apply pending txns	-
2.2			
2.3		-	



Approach #3 (Our Design): Primary Prepares, Backup Commits, Primary Commits

Crash	Primary	Backup	Client
3.1	GetState(txnId) from primary on reboot and applies changes Apply pending txns	-	-
3.2			
3.3	-	-	-



System Behavior on Crash

		Scenario	Behavior
Case- 1		When both nodes are up and no failures	• Reads: Primary/Backup • Writes: Only Primary
Case-2	i	Backup crashed	Reads/Writes: Only Primary
	ii	Primary crashed	Backup->Primary, Reads/Writes: Only Primary
Case - 3		If one node is alive and the other is recovering	Ongoing write requests are sent to the live node and future writes are rejected until the recovering node is stable, reads are sent to the live node and finally, we go back to Case #1/#2 once we reach a stable state for the recovering node. We assume that there are no failures during the recovery process

Design Decisions

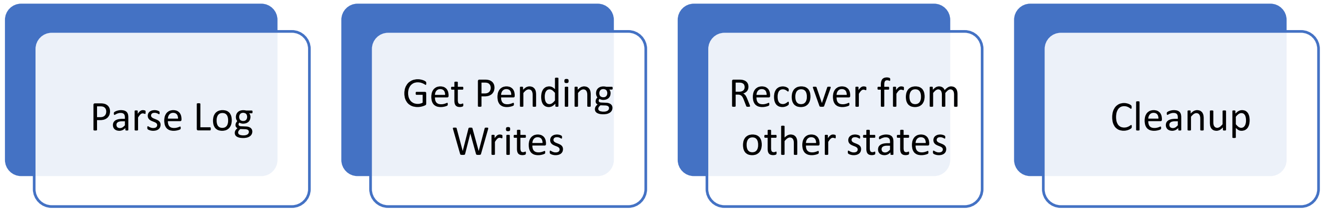
Retry mechanism on client

Lock file during writes

RPC status UNAVAILABLE is used to determine crash

Primary stops processing writes when backup is recovering

Crash Recovery Protocol



Parse Log

Get Pending Writes

Recover from other states

Cleanup

Durability (ALICE)

```
Write(Addr)
…
creat(tmp)
write(tmp)
fsync(tmp)
…
[rename(tmp,file)]
…
```

Testing Strategy

CRC for comparing block content

Macros for predefined crash points

system(kill) and dereferencing NULL to simulate a crash

Measurements

System Specifications

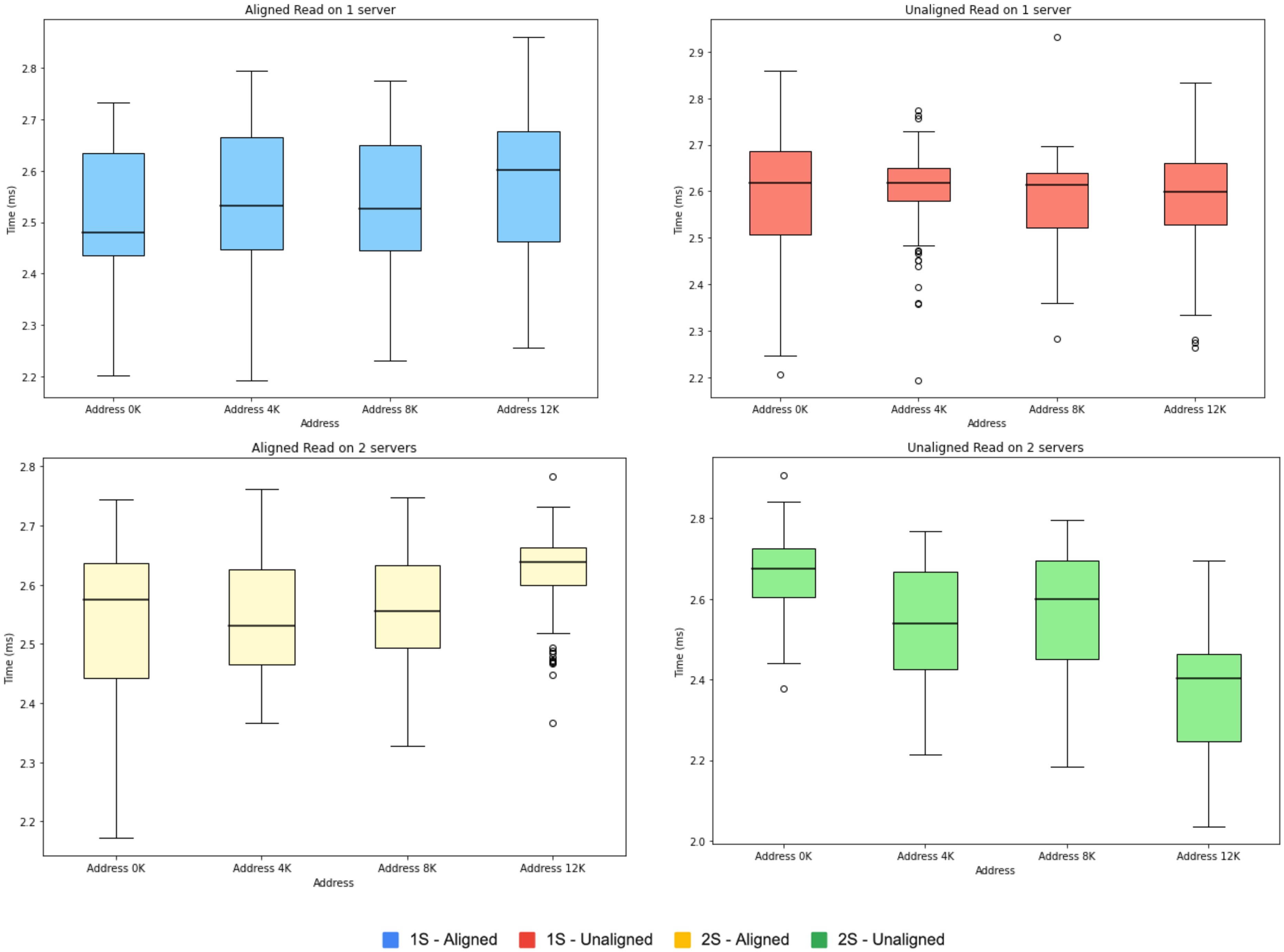
Cloud Provider: CloudLab

Architecture: x86_64

CPU(s): 40

OS: Ubuntu 18 LTS

Read Latency without Crash



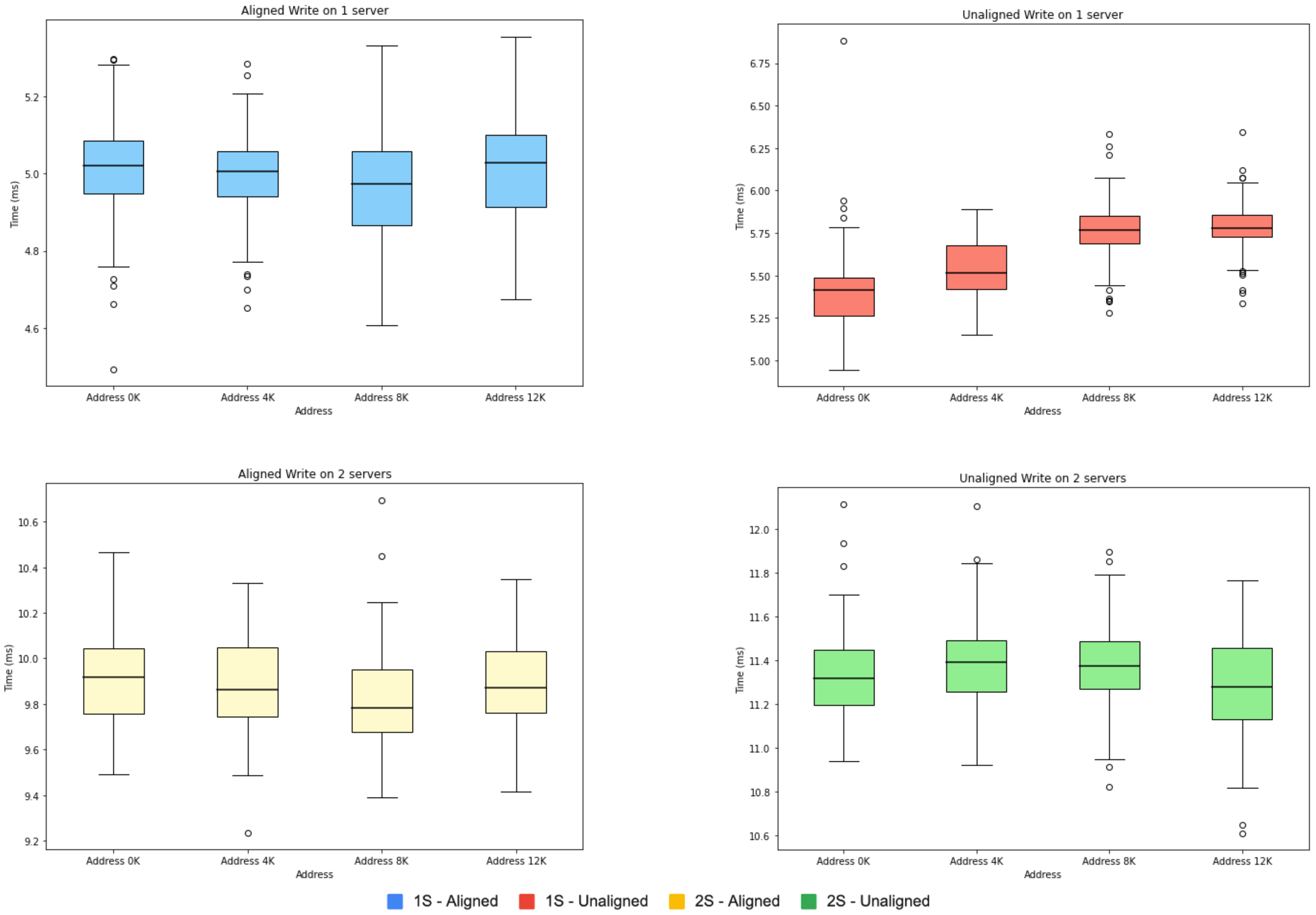
Aligned Read on 1 server

Unaligned Read on 1 server

Aligned Read on 2 servers

Unaligned Read on 2 servers

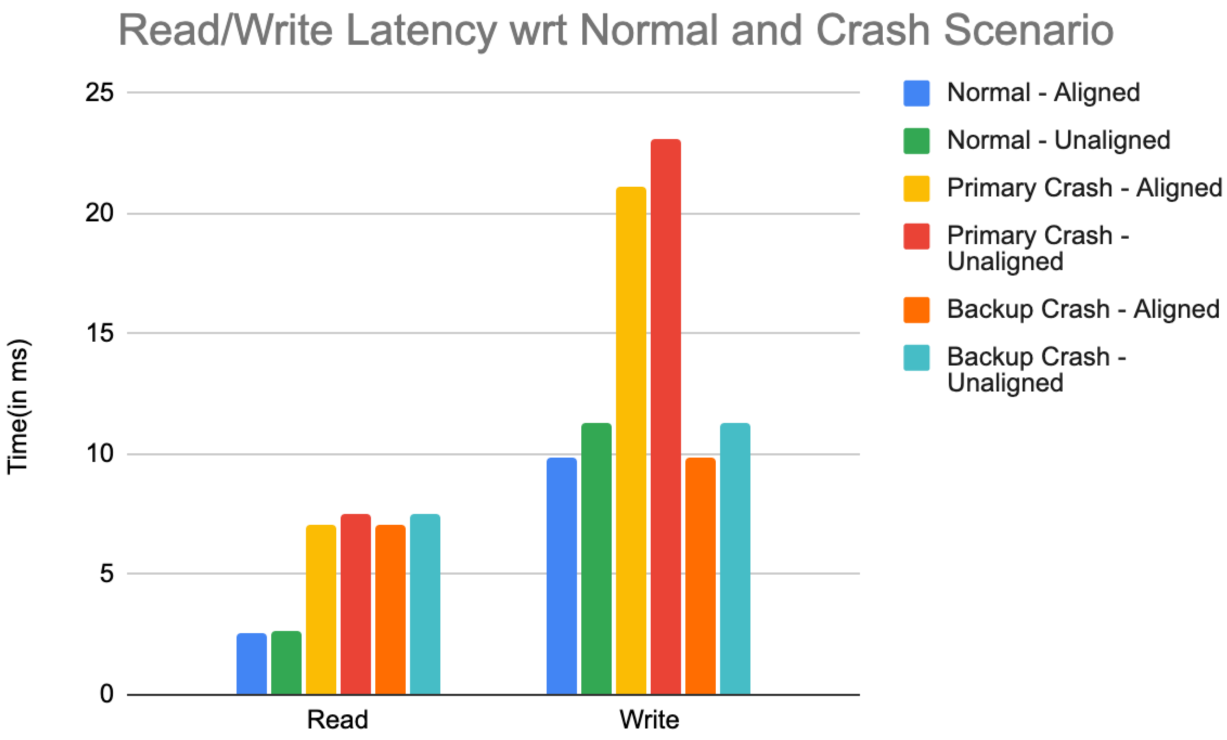
Write Latency without Crash


Aligned Write on 1 server
Time (ms)
5.2
5.0
4.8
4.6
Address 0K
Address 4K
Address 8K
Address 12K
Address
Unaligned Write on 1 server
Time (ms)
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
Address 0K
Address 4K
Address 8K
Address 12K
Address
Aligned Write on 2 servers
Time (ms)
10.6
10.4
10.2
10.0
9.8
9.6
9.4
9.2
Address 0K
Address 4K
Address 8K
Address 12K
Address
Unaligned Write on 2 servers
Time (ms)
12.0
11.8
11.6
11.4
11.2
11.0
10.8
10.6
Address 0K
Address 4K
Address 8K
Address 12K
Address
🟦 1S - Aligned
🟥 1S - Unaligned
🟨 2S - Aligned
🟩 2S - Unaligned

Read and Write Latency with Crash

Key Observations:

- Normal = 2 servers up
- For read, latency for primary crash is almost double that of reads without crash, plus 1 exponential backoff
- For writes, latency for primary crash is (two server + one server + heartbeat latency + 1 exponential backoff
- Backup crashes behave the same as normal



Configuration for Concurrent Operations

Spin up X number of worker threads. Each threads work on different directories.

Each worker thread starts from specific offset in that directory and jumps within that directory uniformly.

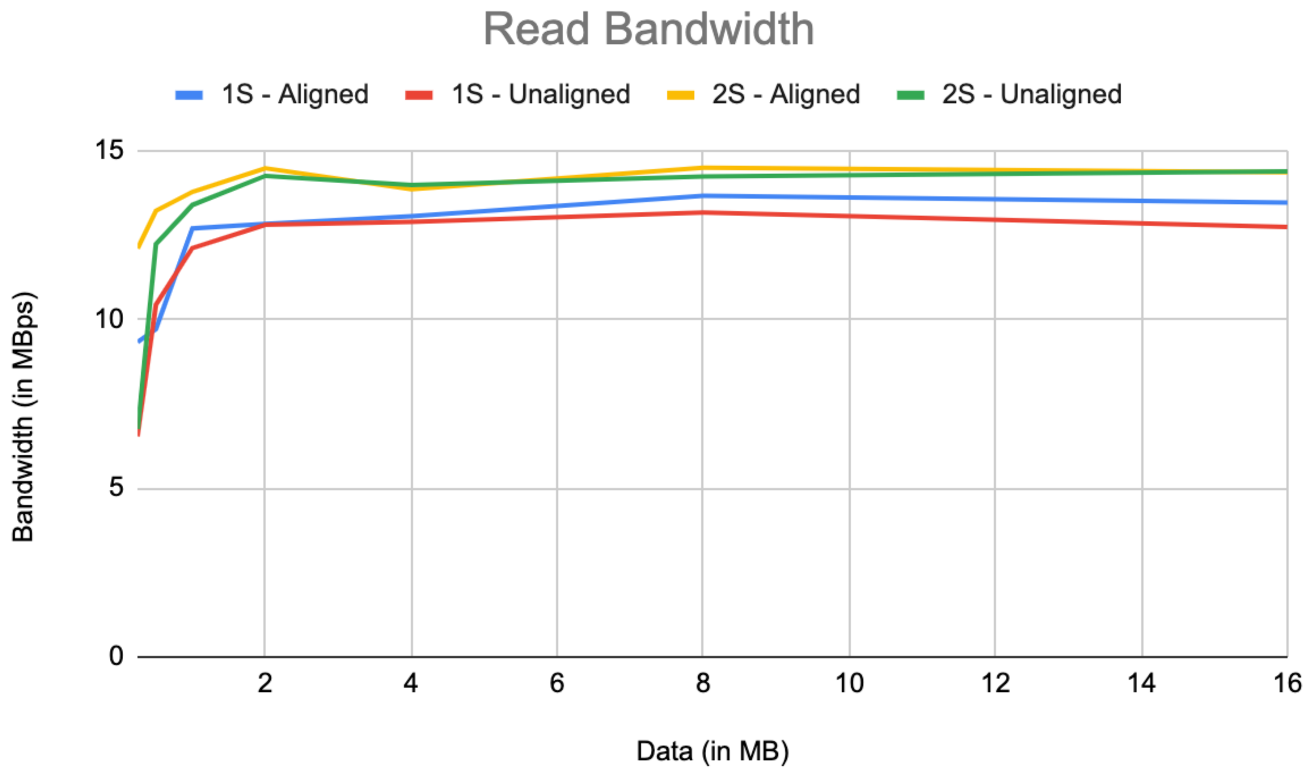
Aligned Address: Starts from 0th offset

Unaligned Address: Starts from (4K+2K)th offset

Read Scalability

Key Observations:

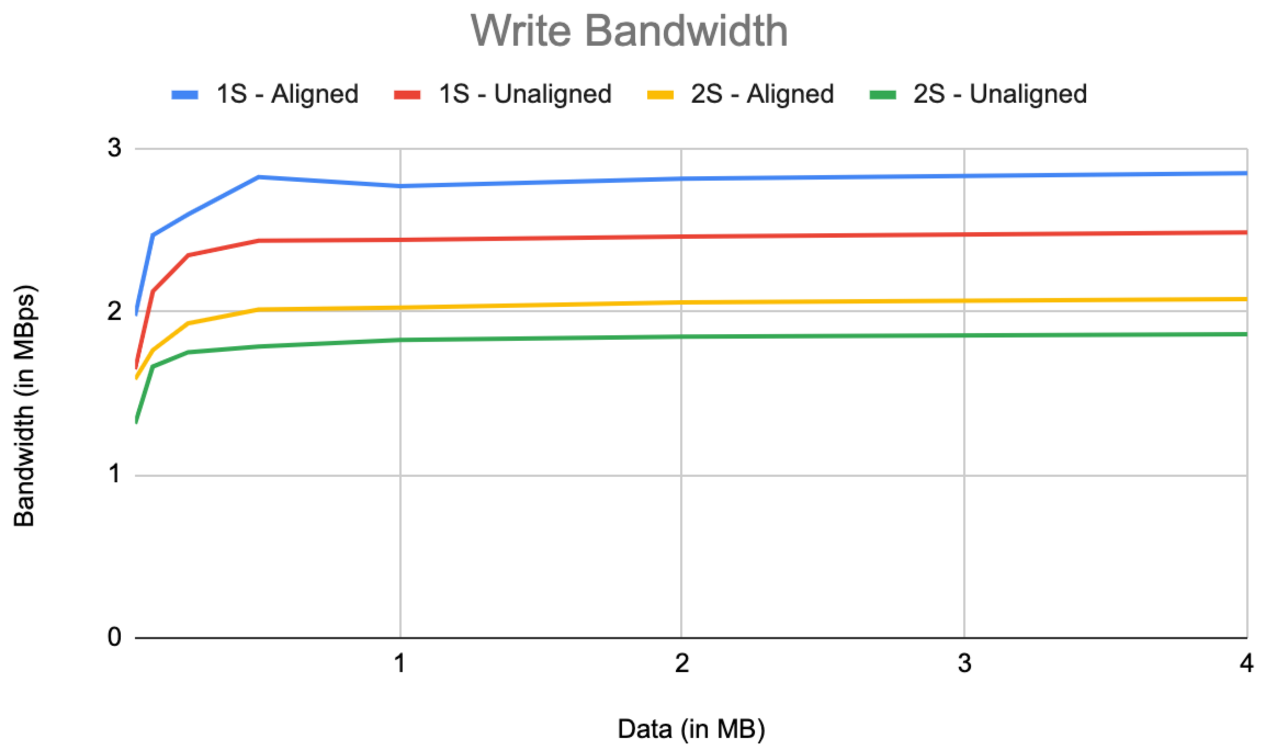
- For 1 server, Aligned vs Unaligned difference is more noticeable when number of request increases (Reason: 2x file reads per request).
- For 2 Servers, Aligned vs Unaligned difference decreases when number of request increases (Reason: Load Balancer + load distribution over 2 servers).
- Bandwidth: 2 Server > 1 Server; Aligned > Unaligned



Write Scalability

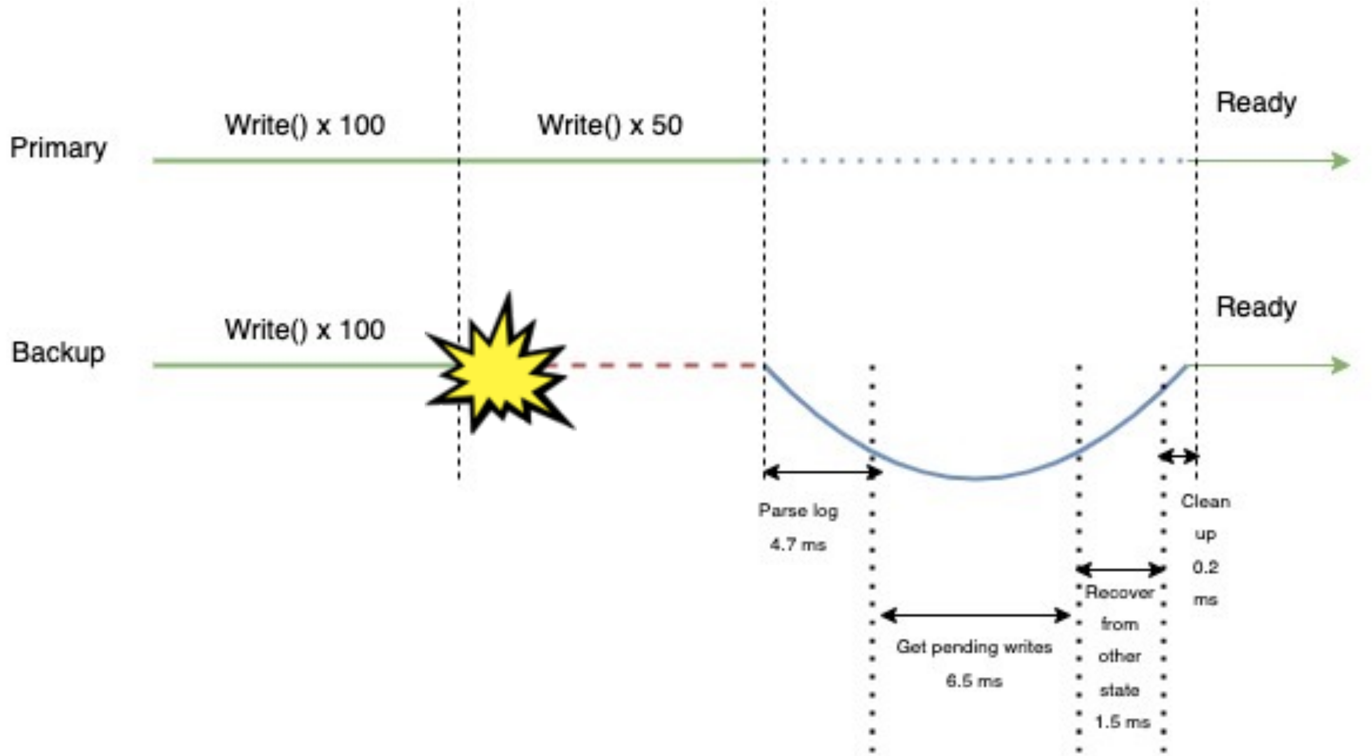
Key Observations:

- Bandwidth: 1 Server > 2 Server and Aligned > Unaligned



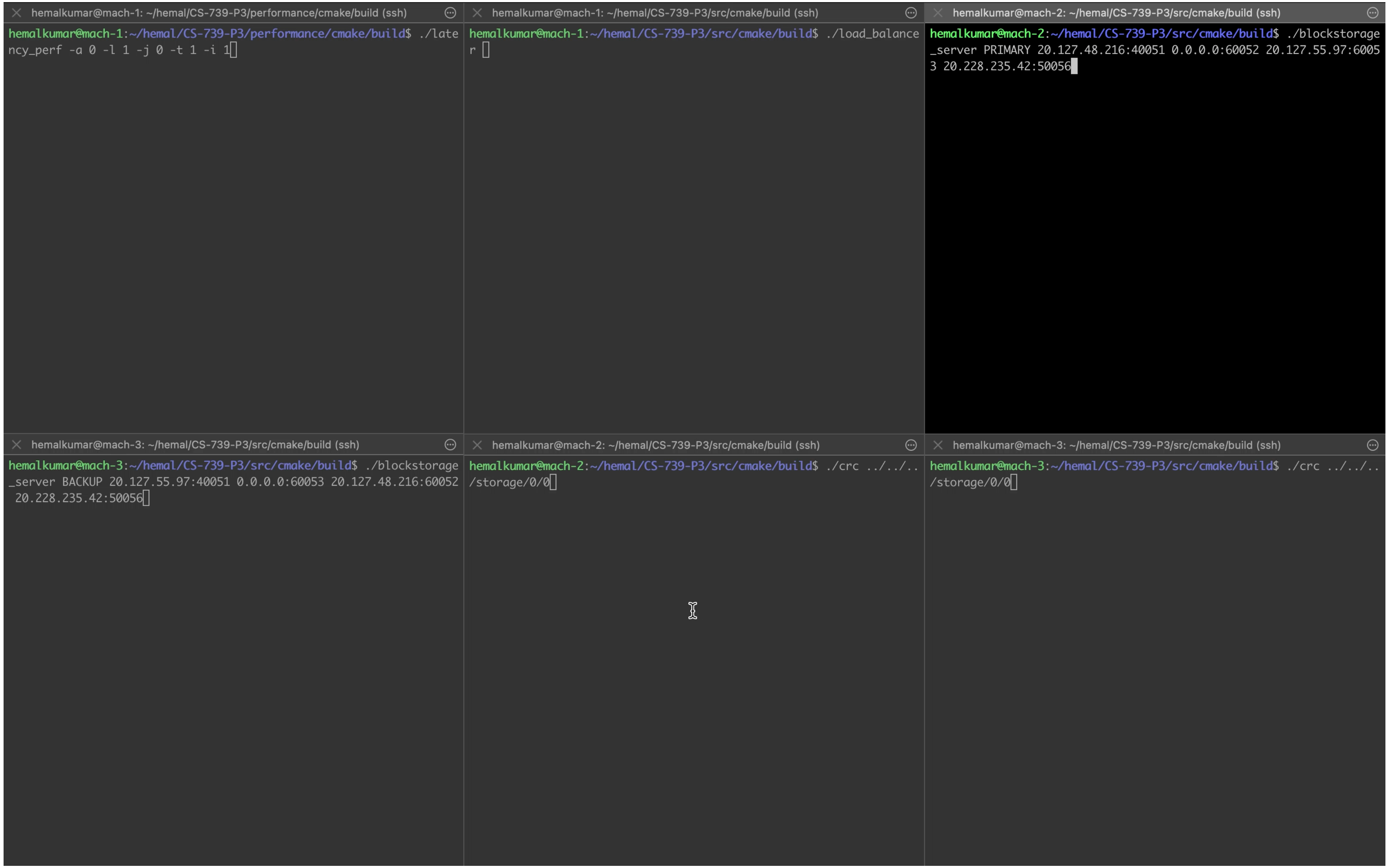
Recovery Time

- 4.7 ms for parsing 100 writes
- 6.5 ms for 50 pending writes
- 1.5 ms for recovering from other states i.e. looping through the 100 states from the 100 writes when the backup was alive
- 0.2 ms to delete the in-memory map of the log and to truncate the log file



Demos

Failover


hemalkumar@mach-1: ~/hemal/CS-739-P3/performance/cmake/build (ssh)
hemalkumar@mach-1: ~/hemal/CS-739-P3/src/cmake/build (ssh)
hemalkumar@mach-2: ~/hemal/CS-739-P3/src/cmake/build (ssh)
hemalkumar@mach-3: ~/hemal/CS-739-P3/src/cmake/build (ssh)
hemalkumar@mach-2: ~/hemal/CS-739-P3/src/cmake/build (ssh)
hemalkumar@mach-3: ~/hemal/CS-739-P3/src/cmake/build (ssh)

```
hemalkumar@mach-1:~/hemal/CS-739-P3/performance/cmake/build$ ./late
ncy_perf -a 0 -l 1 -j 0 -t 1 -i 1
```

```
hemalkumar@mach-1:~/hemal/CS-739-P3/src/cmake/build$ ./load_balance
r
```

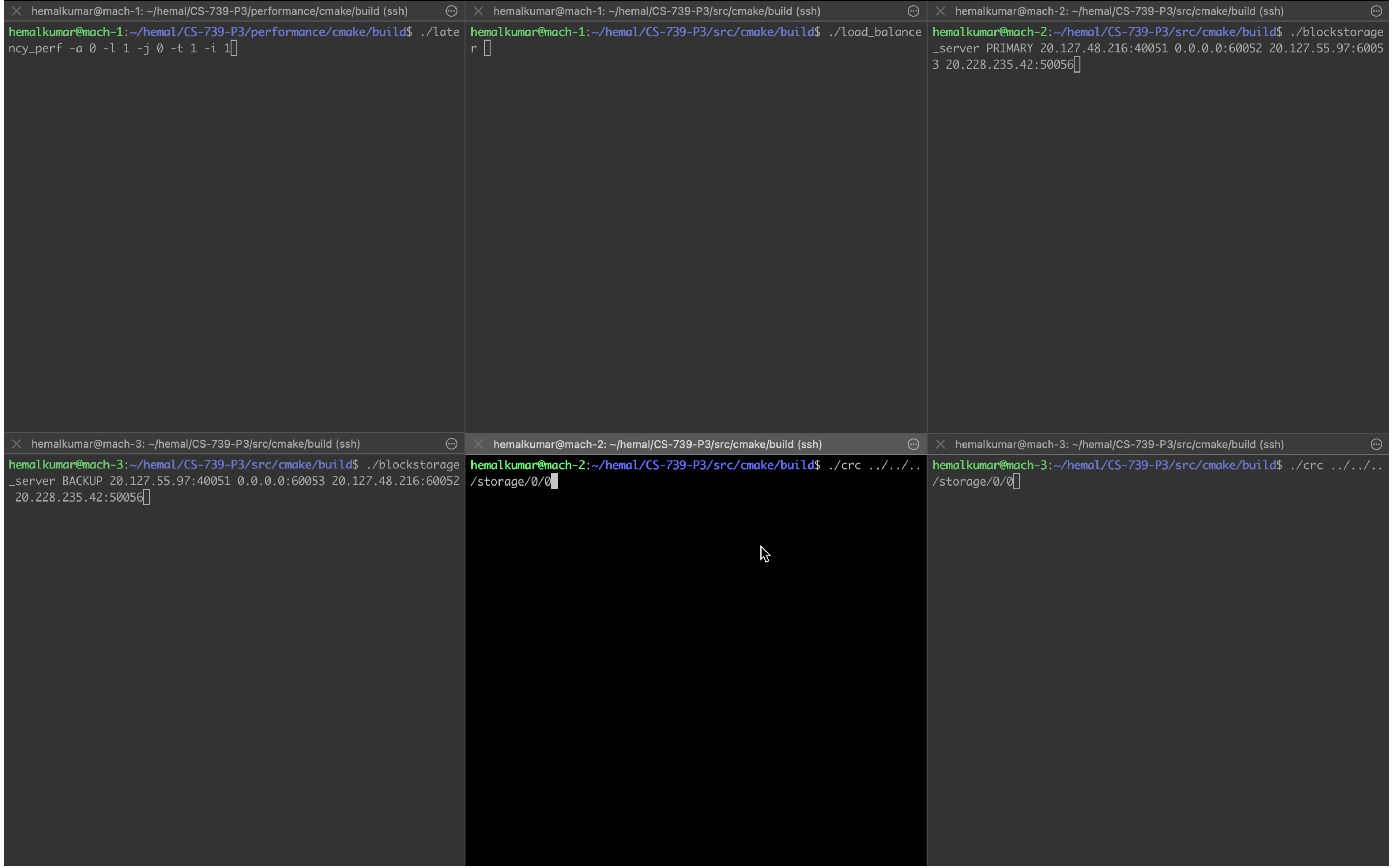
```
hemalkumar@mach-2:~/hemal/CS-739-P3/src/cmake/build$ ./blockstorage
_server PRIMARY 20.127.48.216:40051 0.0.0.0:60052 20.127.55.97:6005
3 20.228.235.42:50056
```

```
hemalkumar@mach-3:~/hemal/CS-739-P3/src/cmake/build$ ./blockstorage
_server BACKUP 20.127.55.97:40051 0.0.0.0:60053 20.127.48.216:60052
 20.228.235.42:50056
```

```
hemalkumar@mach-2:~/hemal/CS-739-P3/src/cmake/build$ ./crc ../../..
/storage/0/0
```

```
hemalkumar@mach-3:~/hemal/CS-739-P3/src/cmake/build$ ./crc ../../..
/storage/0/0
```

Backup failure during write operation



hemalkumar@mach-1: ~/hemal/CS-739-P3/performance/cmake/build (ssh)

```
hemalkumar@mach-1:~/hemal/CS-739-P3/performance/cmake/build$ ./late
ncy_perf -a 0 -l 1 -j 0 -t 1 -i 1
```

hemalkumar@mach-1: ~/hemal/CS-739-P3/src/cmake/build (ssh)

```
hemalkumar@mach-1:~/hemal/CS-739-P3/src/cmake/build$ ./load_balance
r
```

hemalkumar@mach-2: ~/hemal/CS-739-P3/src/cmake/build (ssh)

```
hemalkumar@mach-2:~/hemal/CS-739-P3/src/cmake/build$ ./blockstorage
_server PRIMARY 20.127.48.216:40051 0.0.0.0:60052 20.127.55.97:6005
3 20.228.235.42:50056
```

hemalkumar@mach-3: ~/hemal/CS-739-P3/src/cmake/build (ssh)

```
hemalkumar@mach-3:~/hemal/CS-739-P3/src/cmake/build$ ./blockstorage
_server BACKUP 20.127.55.97:40051 0.0.0.0:60053 20.127.48.216:60052
 20.228.235.42:50056
```

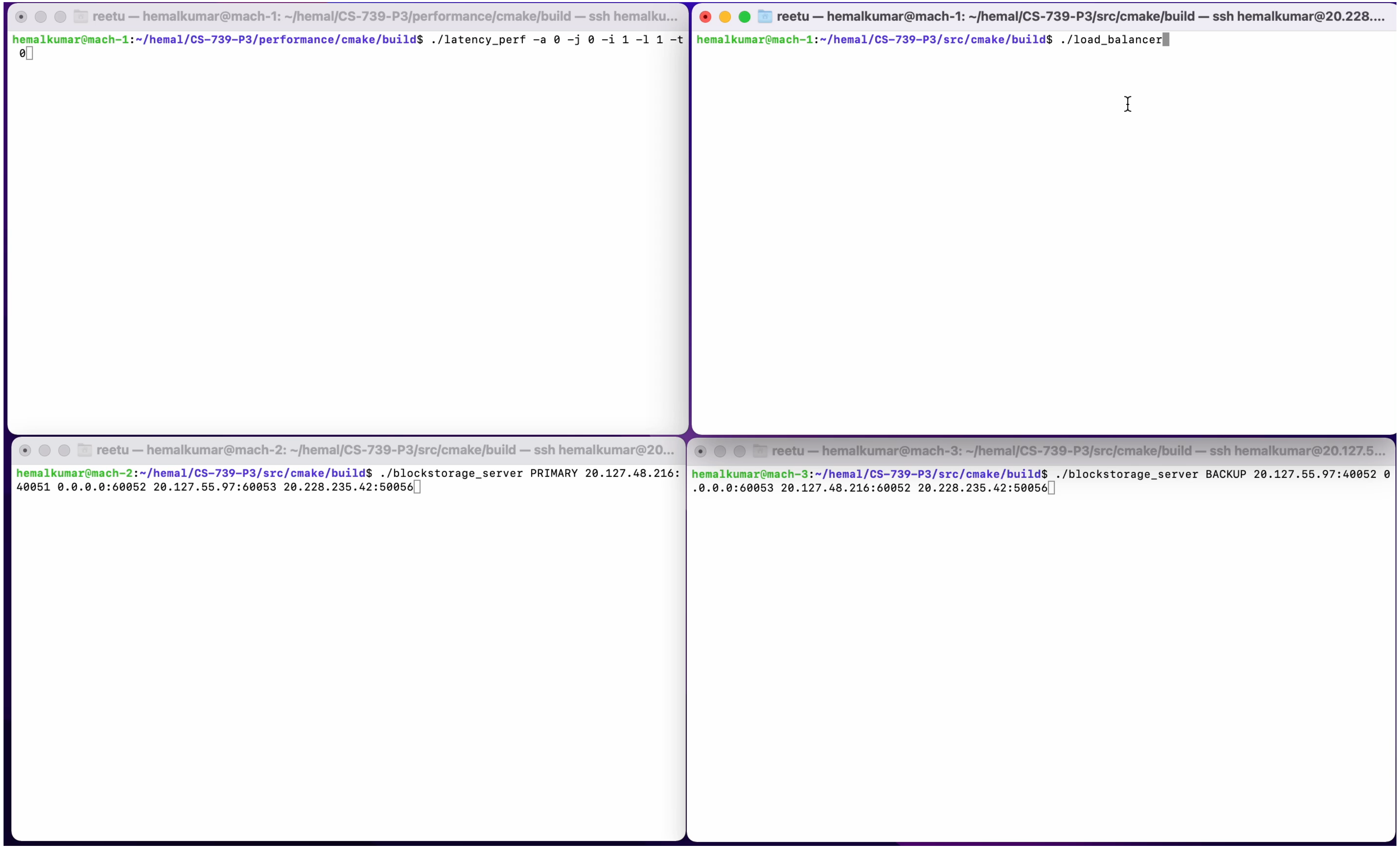
hemalkumar@mach-2: ~/hemal/CS-739-P3/src/cmake/build (ssh)

```
hemalkumar@mach-2:~/hemal/CS-739-P3/src/cmake/build$ ./crc ../../..
/storage/0/0
```

hemalkumar@mach-3: ~/hemal/CS-739-P3/src/cmake/build (ssh)

```
hemalkumar@mach-3:~/hemal/CS-739-P3/src/cmake/build$ ./crc ../../..
/storage/0/0
```

Read with crash



reetu — hemalkumar@mach-1: ~/hemal/CS-739-P3/performance/cmake/build — ssh hemalku...

```
hemalkumar@mach-1:~/hemal/CS-739-P3/performance/cmake/build$ ./latency_perf -a 0 -j 0 -i 1 -l 1 -t
 0
```

reetu — hemalkumar@mach-1: ~/hemal/CS-739-P3/src/cmake/build — ssh hemalkumar@20.228....

```
hemalkumar@mach-1:~/hemal/CS-739-P3/src/cmake/build$ ./load_balancer
```

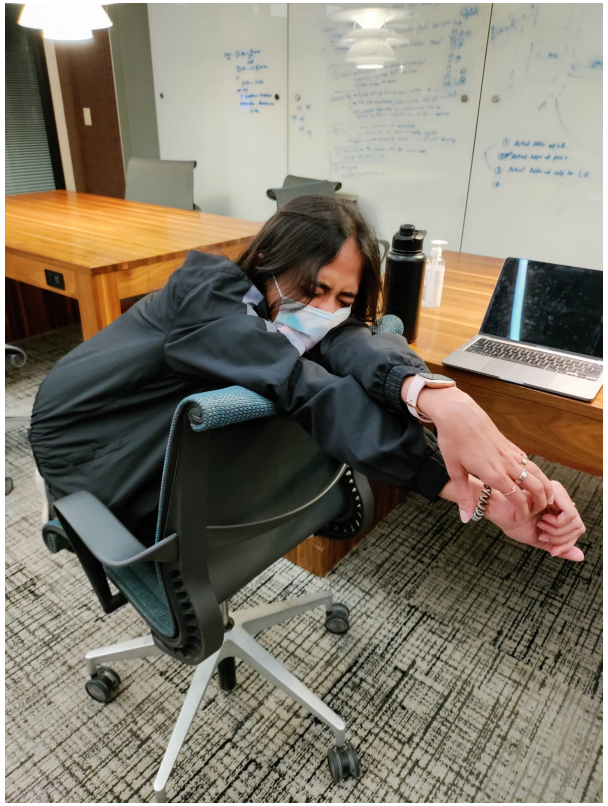
reetu — hemalkumar@mach-2: ~/hemal/CS-739-P3/src/cmake/build — ssh hemalkumar@20...

```
hemalkumar@mach-2:~/hemal/CS-739-P3/src/cmake/build$ ./blockstorage_server PRIMARY 20.127.48.216:
40051 0.0.0.0:60052 20.127.55.97:60053 20.228.235.42:50056
```

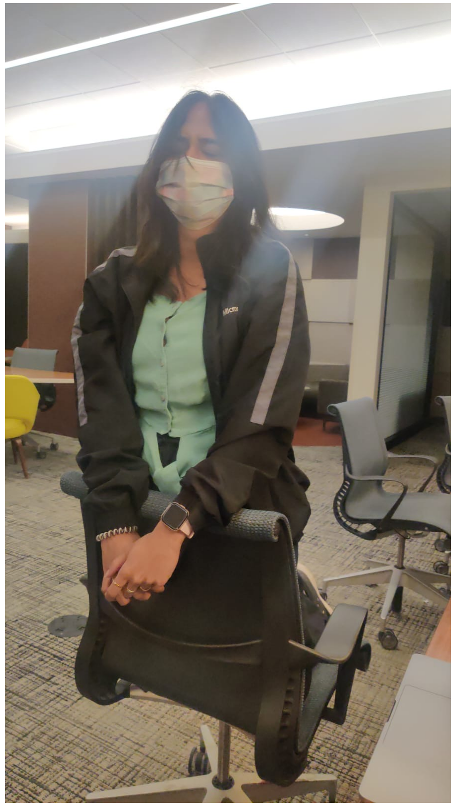
reetu — hemalkumar@mach-3: ~/hemal/CS-739-P3/src/cmake/build — ssh hemalkumar@20.127.5...

```
hemalkumar@mach-3:~/hemal/CS-739-P3/src/cmake/build$ ./blockstorage_server BACKUP 20.127.55.97:40052 0
.0.0.0:60053 20.127.48.216:60052 20.228.235.42:50056
```

Thank You!



Crash!!!!



Recovery

No request to serve...