CS 739 P3 - Block Storage w/ Primary-Backup Arch

Mini-Project 3 12
-Lincoln Spartacus James
-Himanshu Pandotra
-Pavithran Ravichandiran
-Madhav Kanbur

High Level Architecture

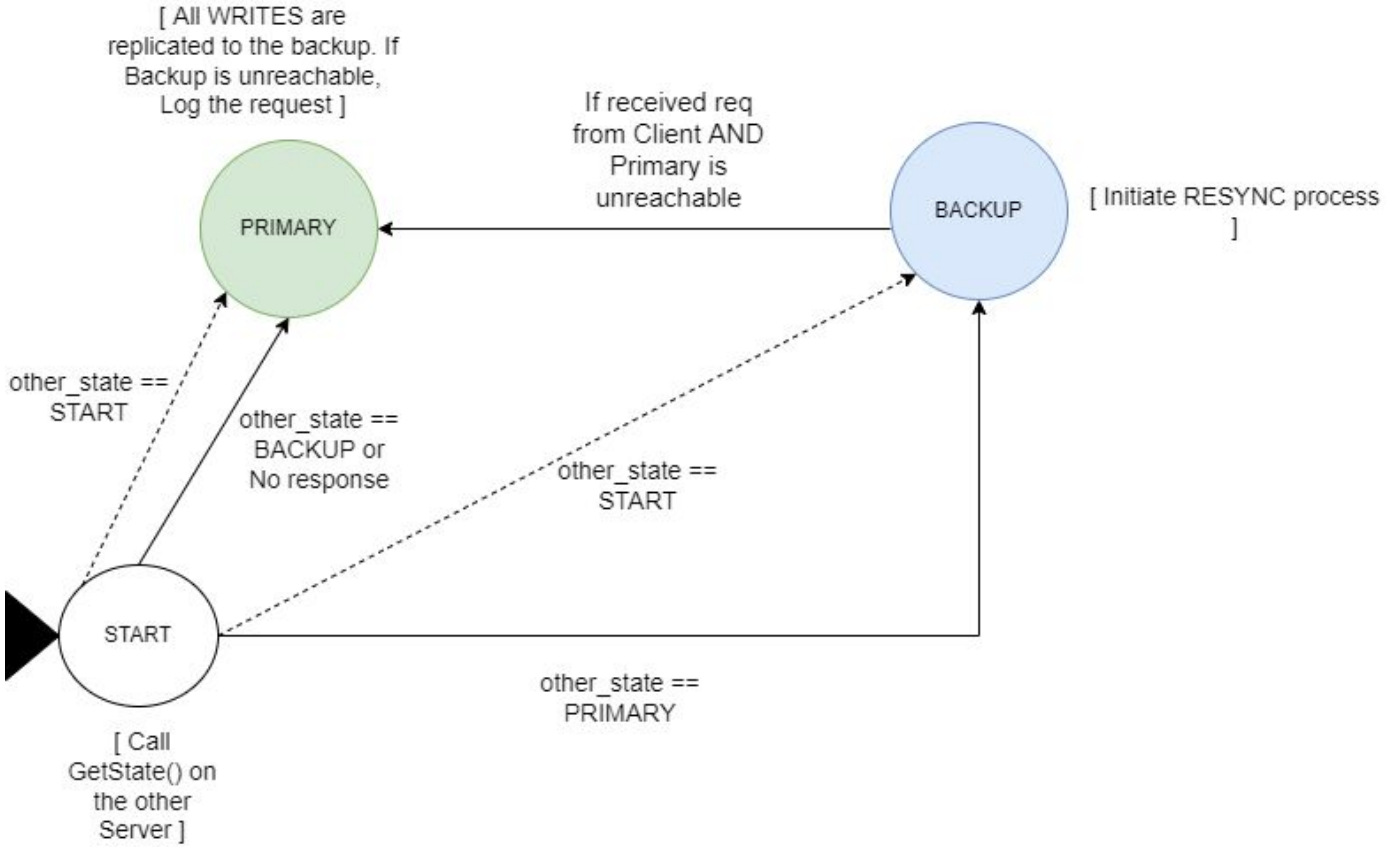
Client

- API supports only 4K reads/writes (could be unaligned)
- Only one outstanding request per client.
- Client Library handles switching between the 2 servers (transparent to client)

Primary-Backup

- All WRITES are replicated to the Backup before replying to Client
- Backup never handles requests. It redirects client to Primary (or becomes the Primary)
- No network faults between Primary - Backup (no partition, failure etc.)
- At most one crash at any given point of time

State Machine

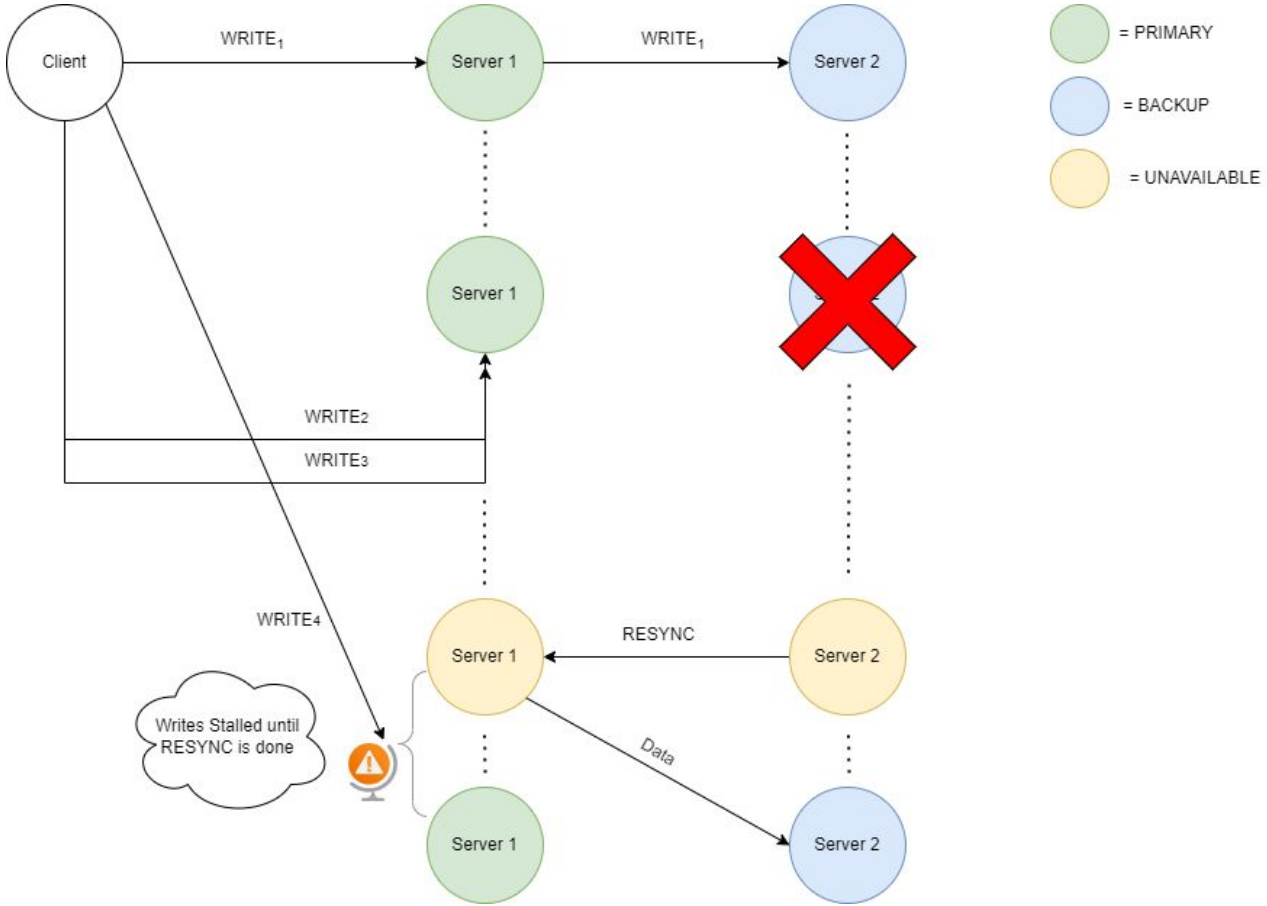


Low-level Details

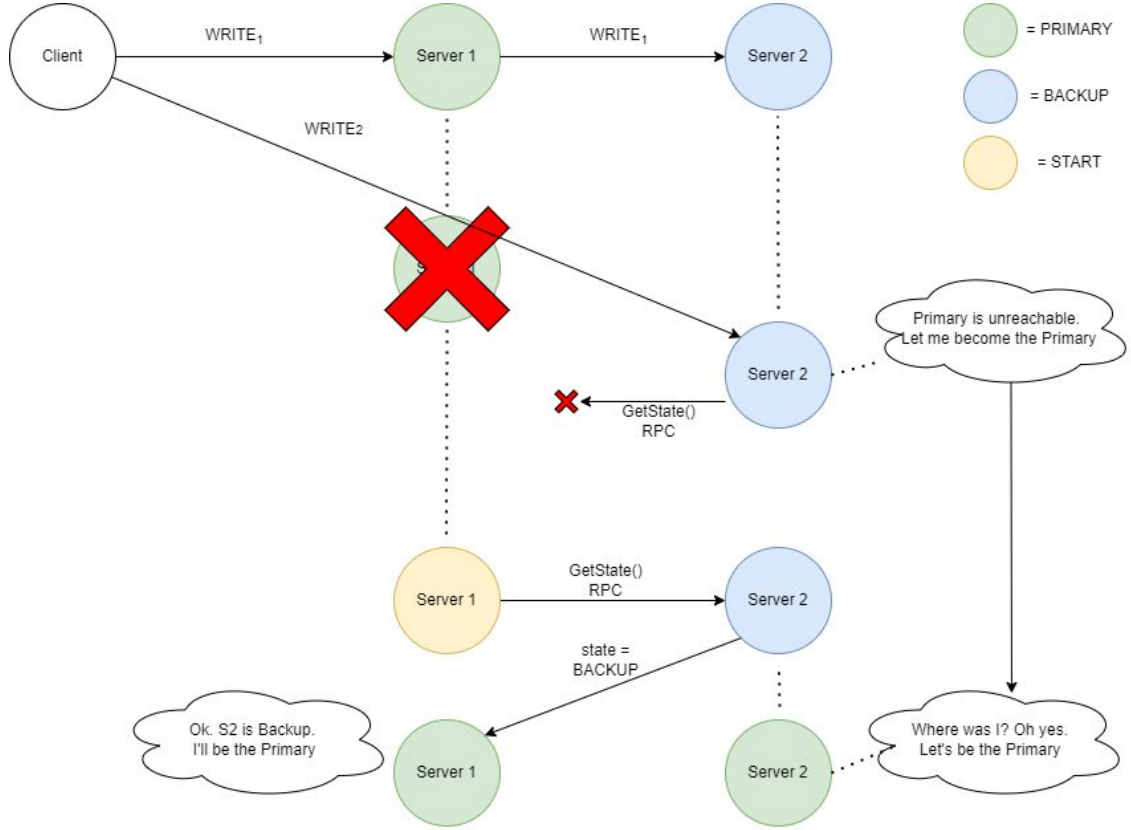
- Use pwrites() to write to one large 256GB file (on EXT4) - Thread Safe, atomic wrt each other[+]
- Fsync on each pwrite()
- Use in-memory **std::set** to track modified block numbers if Backup is dead
- Transmit blocks in this set to the Backup during RESYNC
- Does our log have to persistent? - NO (assuming only 1 crash)

+ https://stackoverflow.com/questions/5268307/thread-safety-of-read-pread-system-calls

RESYNC Process - Stall WRITES

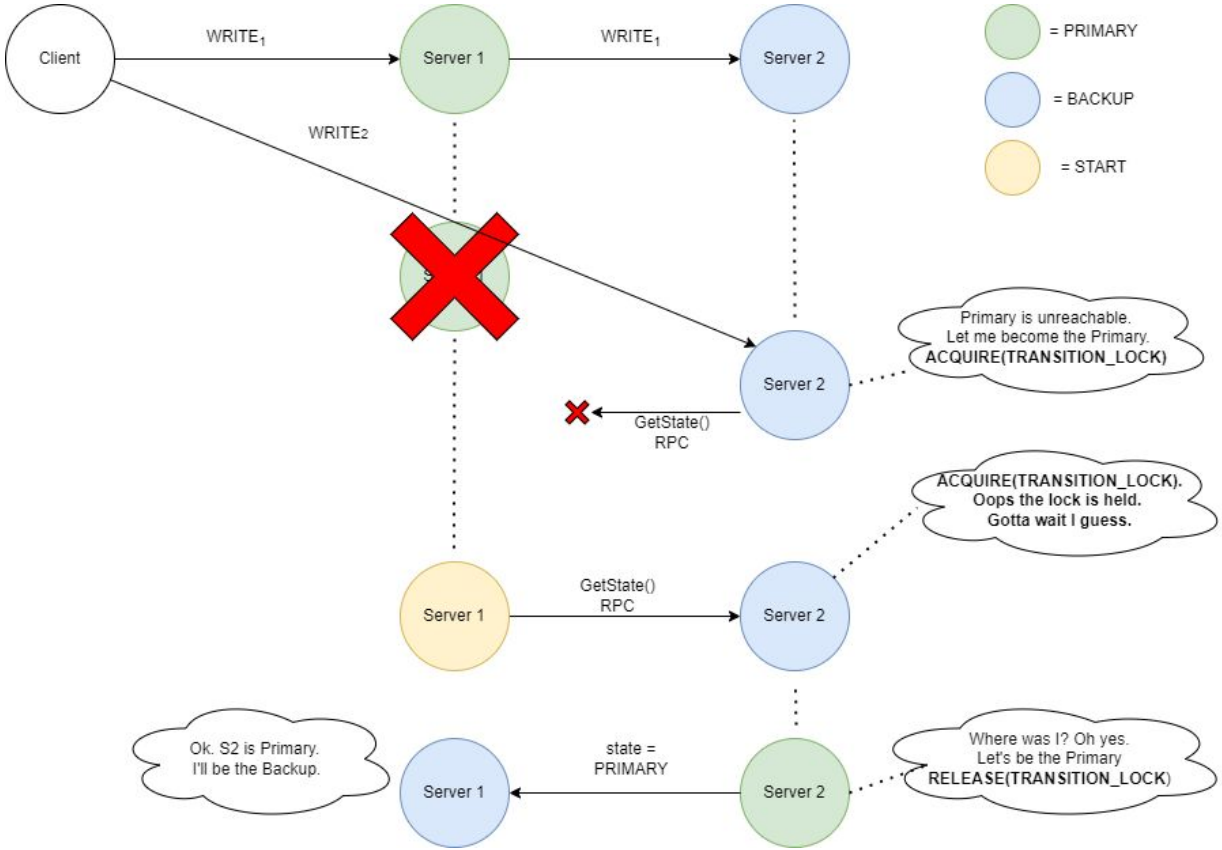


Backup-Transition Race Condition



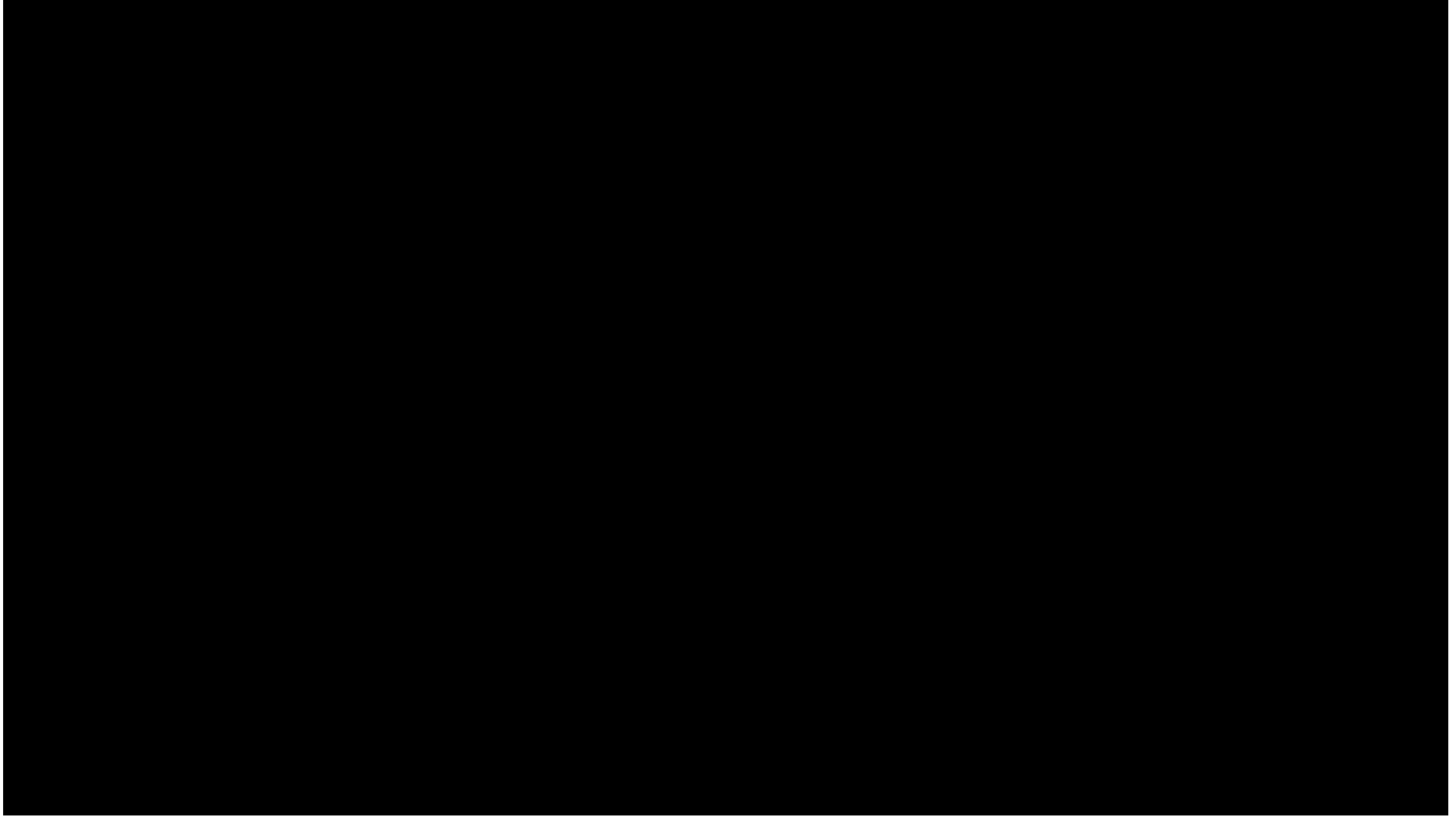
Oops! We have 2 Primaries!

Solution? - Mutex Lock



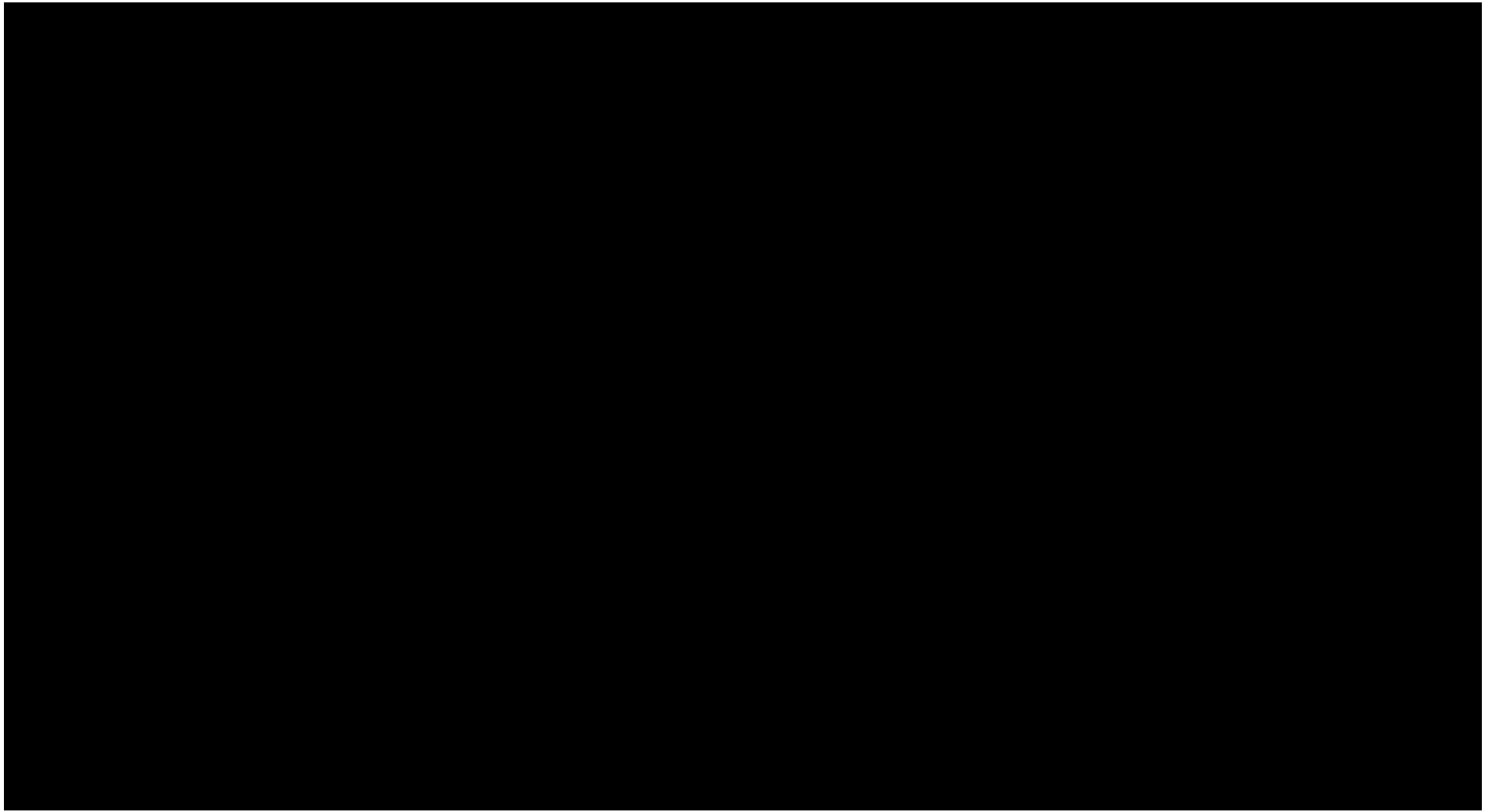
Primary failure - Backup takeover

- Whenever the primary server crashes, the backup detects this failure when the client sends a request
- In the video, we'll see that backup takes over as the primary and starts servicing the client request.
    - Backup also logs writes that needs to be sent to the other server once it comes back up.



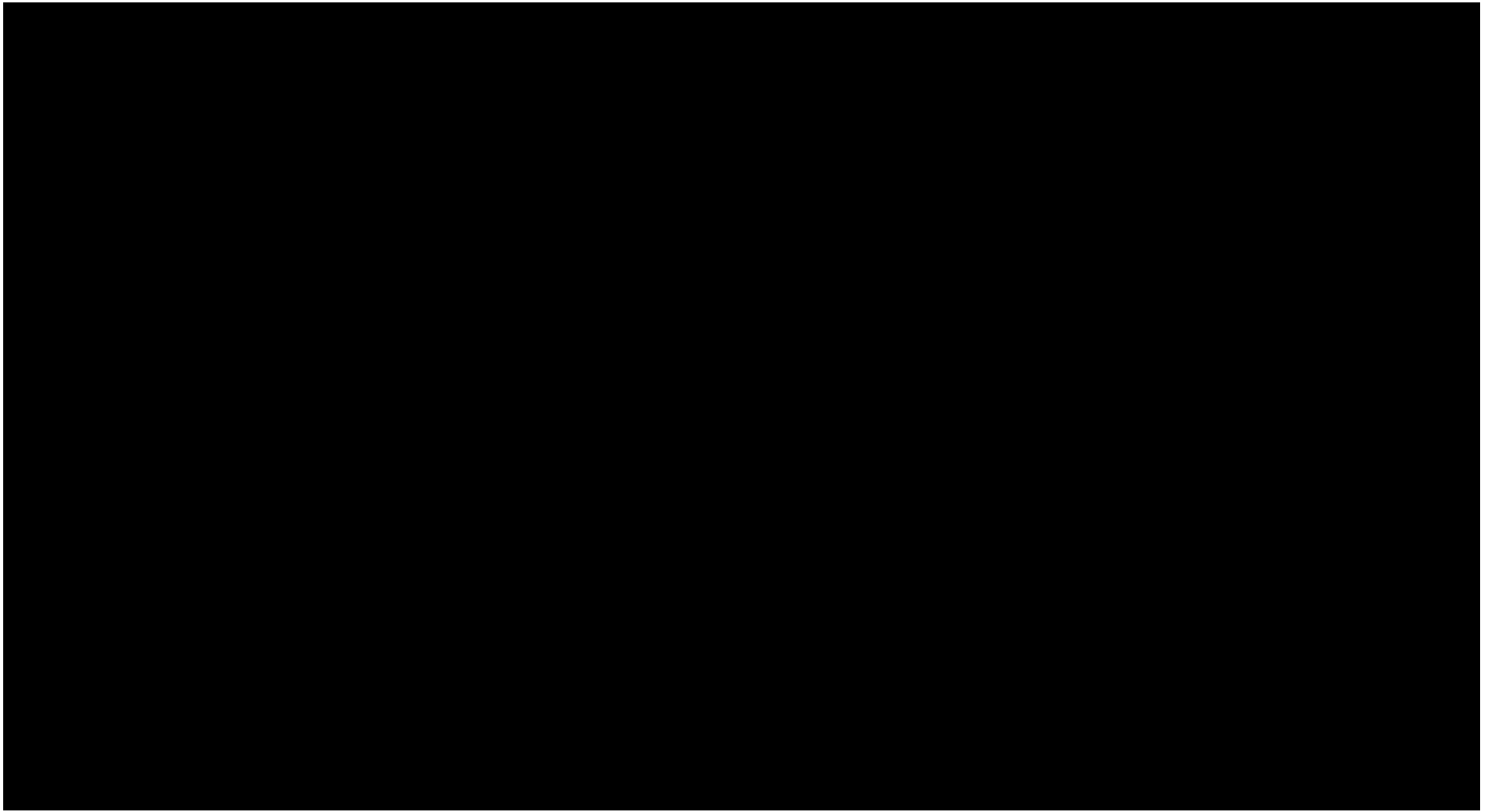
Resync

- Resync happens every time a crashed backup server comes back alive.
- When the backup is down, we store the affected block numbers that the client writes to during the backup's downtime in the logs.
    - One optimization is that we send only the last written data blocks to the backup even if that is written multiple times
- Primary server streams all the pending writes to the backup server.



Clients doesn’t see any crash

- Crash to any of the servers is not visible to the client.
- Client library retries every 1 second till it gets a response from the server.
    - Switches between primary and backup servers on each retry attempt.



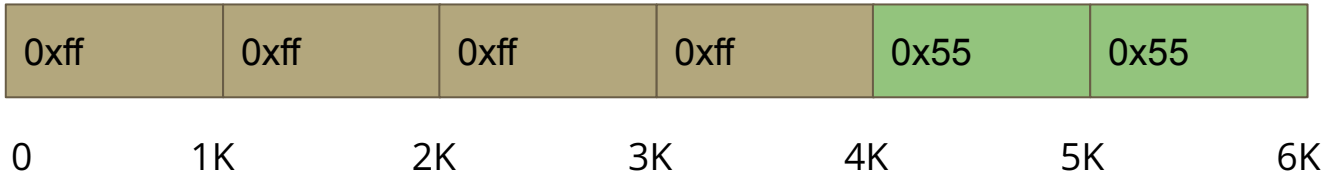
Multiple Clients

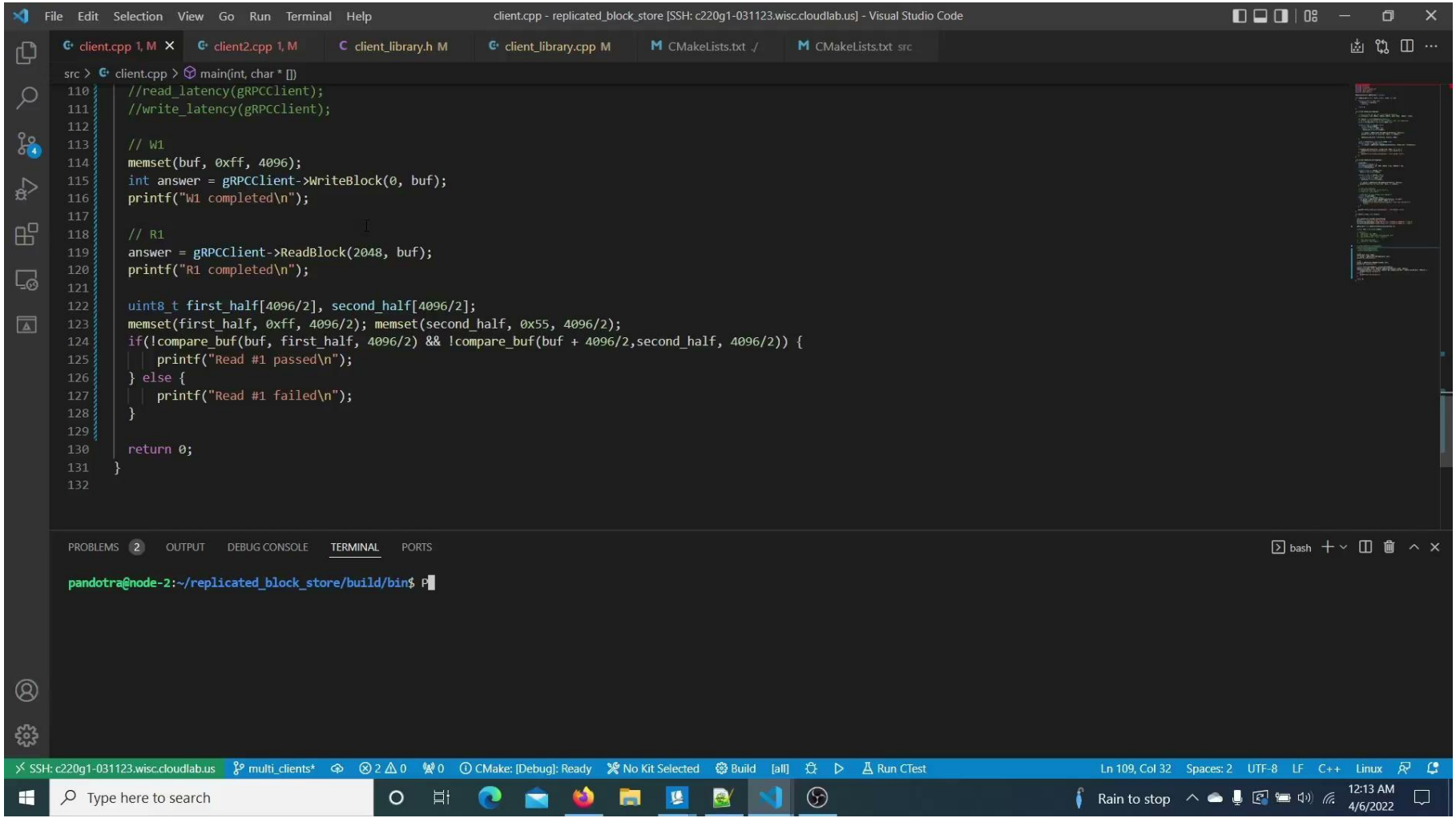
Client 1

1. W1 (0K, 0xff) -> **PRIMARY CRASHED**
2. W1 (0K, 0xff) @ BACKUP
3. R1 (2K) @ BACKUP

Client 2

1. W2 (2K, 0x55) @ BACKUP
2. R2 (0K) @ BACKUP




File Edit Selection View Go Run Terminal Help
client.cpp - replicated_block_store [SSH: c220g1-031123.wisc.cloudlab.us] - Visual Studio Code
client.cpp 1, M
client2.cpp 1, M
client_library.h M
client_library.cpp M
CMakeLists.txt ./
CMakeLists.txt src
src > client.cpp > main(int, char * [])
//read_latency(gRPCClient);
//write_latency(gRPCClient);
// W1
memset(buf, 0xff, 4096);
int answer = gRPCClient->WriteBlock(0, buf);
printf("W1 completed\n");
// R1
answer = gRPCClient->ReadBlock(2048, buf);
printf("R1 completed\n");
uint8_t first_half[4096/2], second_half[4096/2];
memset(first_half, 0xff, 4096/2); memset(second_half, 0x55, 4096/2);
if(!compare_buf(buf, first_half, 4096/2) && !compare_buf(buf + 4096/2,second_half, 4096/2)) {
printf("Read #1 passed\n");
} else {
printf("Read #1 failed\n");
}
return 0;
}
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
bash
pandotra@node-2:~/replicated_block_store/build/bin$ P
SSH: c220g1-031123.wisc.cloudlab.us
multi_clients*
2 0
0
CMake: [Debug]: Ready
No Kit Selected
Build
[all]
Run CTest
Ln 109, Col 32
Spaces: 2
UTF-8
LF
C++
Linux
Type here to search
Rain to stop
12:13 AM
4/6/2022

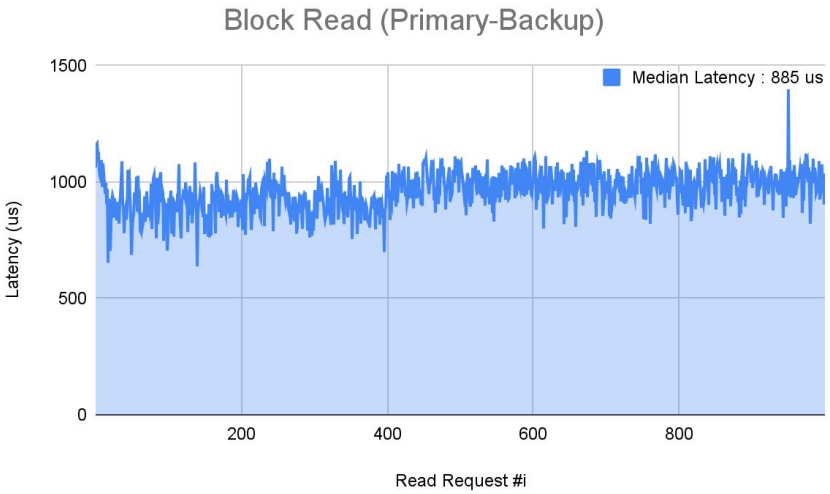
Measurements

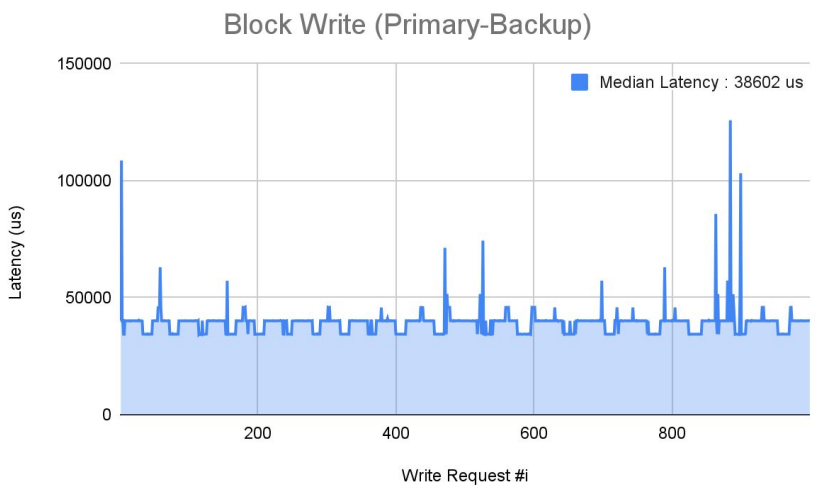
Hardware used : Cloudlab machines

Memory: 16GB

Processor: Intel Xeon Silver @2.2GHz 40 cores

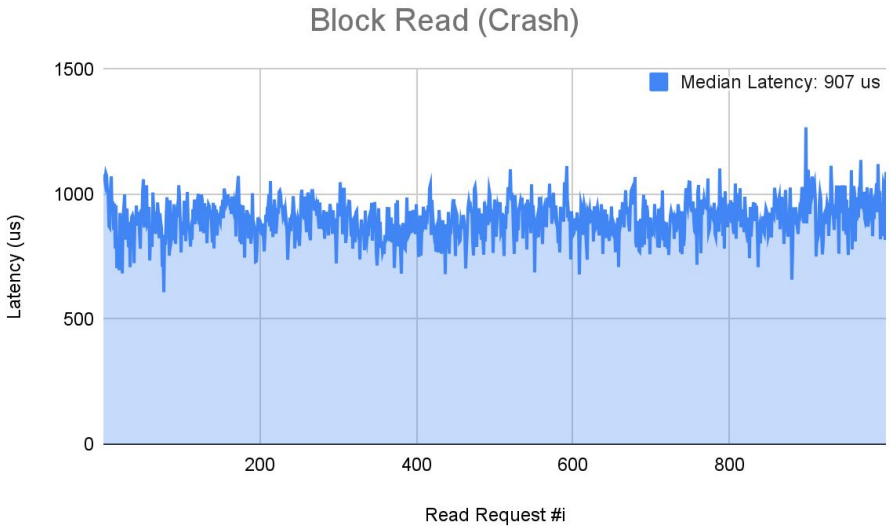
Measurement of Latency for Block Read and Write commands

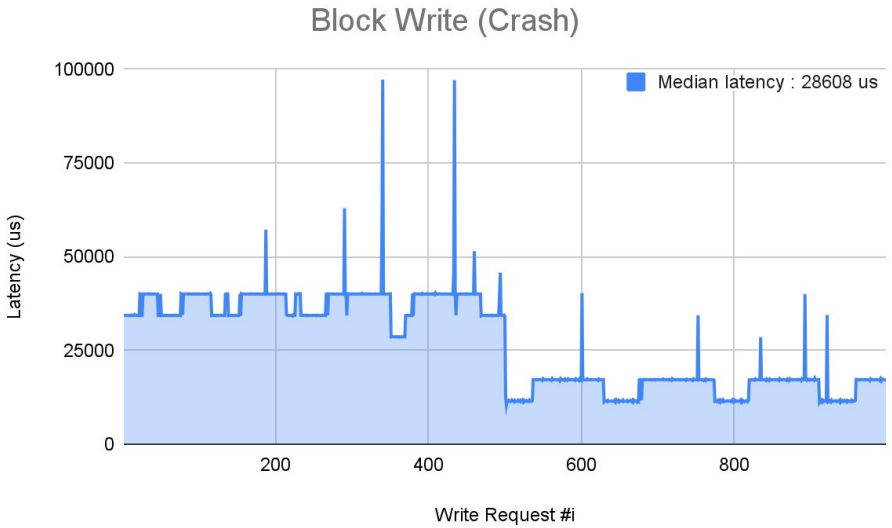




- Case when both primary and backup servers are alive.

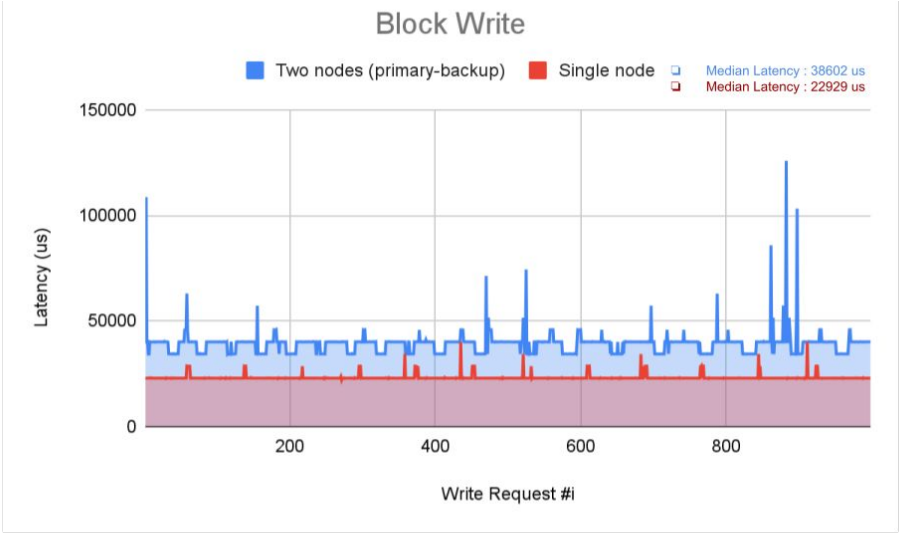
Measurement of Latency for Block Read and Write commands





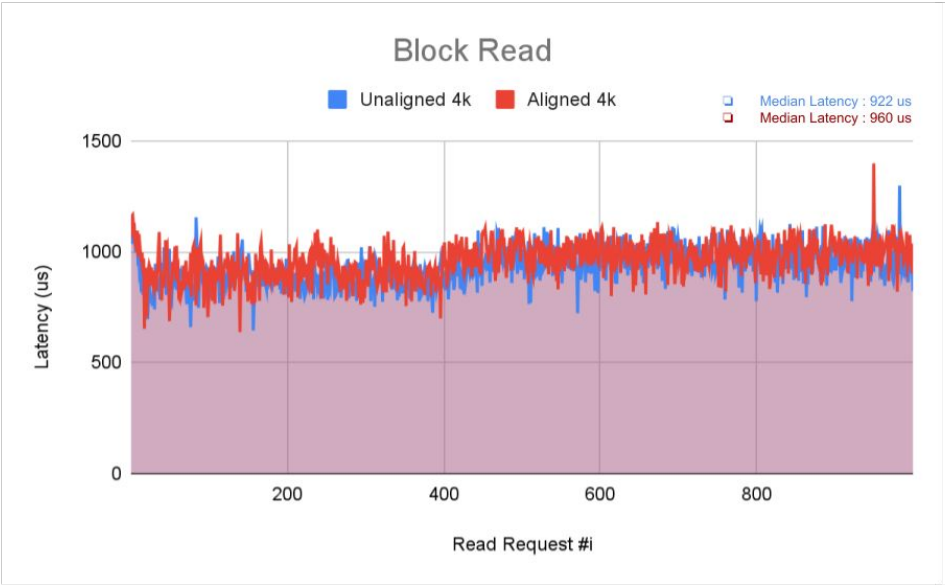
- Primary server crashes in the middle. Load is shifted to the backup server.
- Write latency reduces as the server doesn't have to send data to backup anymore.

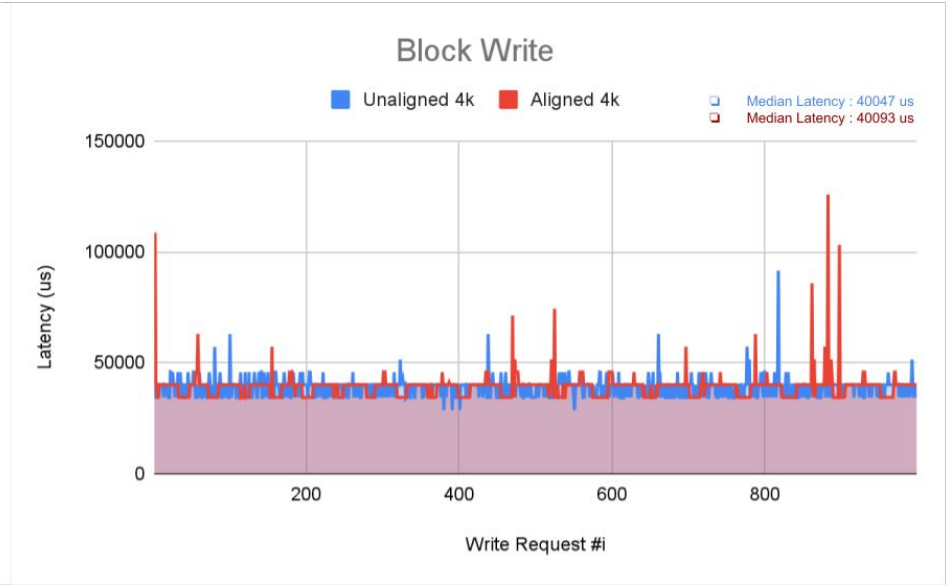
Comparison of Latency in Two nodes vs Single node



- Write latency of single node is better that two nodes as data is only persisted on one server
- Read latency is similar as we only read from the primary server.

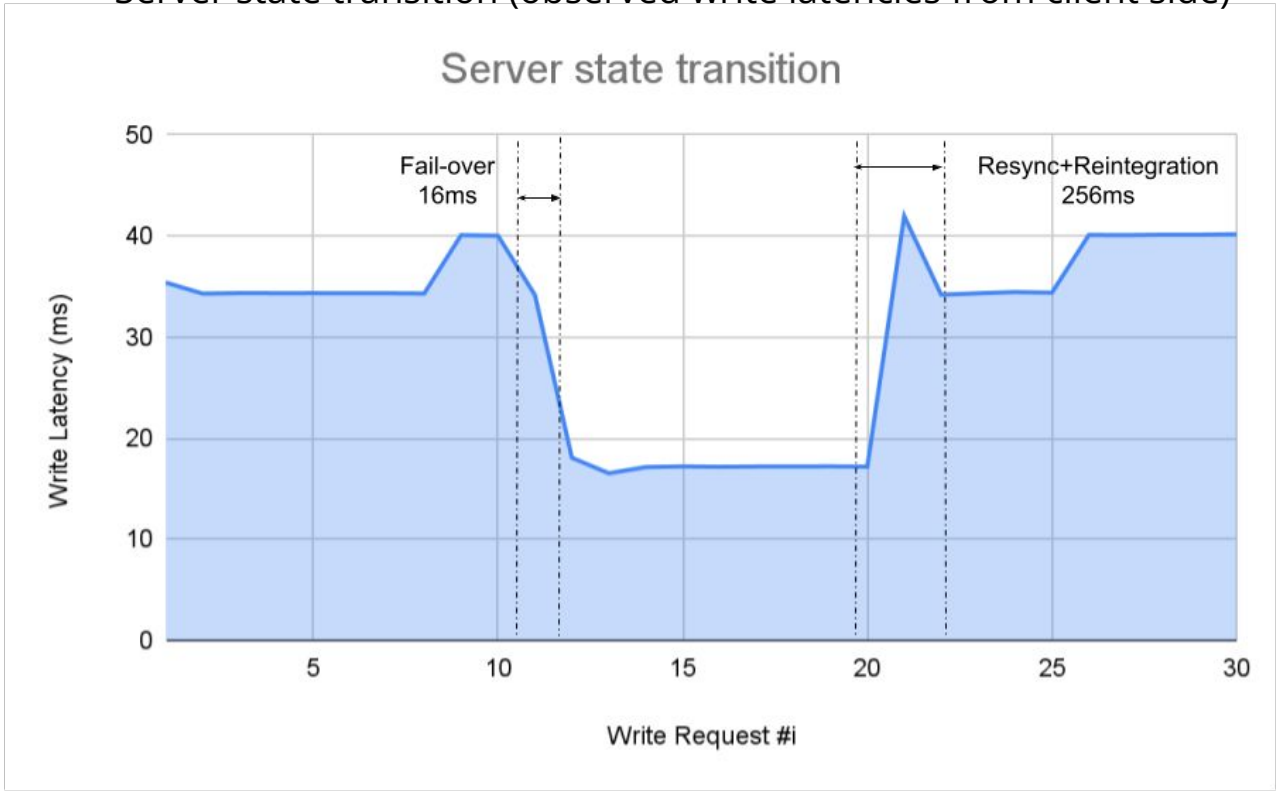
Comparison of Latency in 4K aligned and unaligned Block Read and Write commands



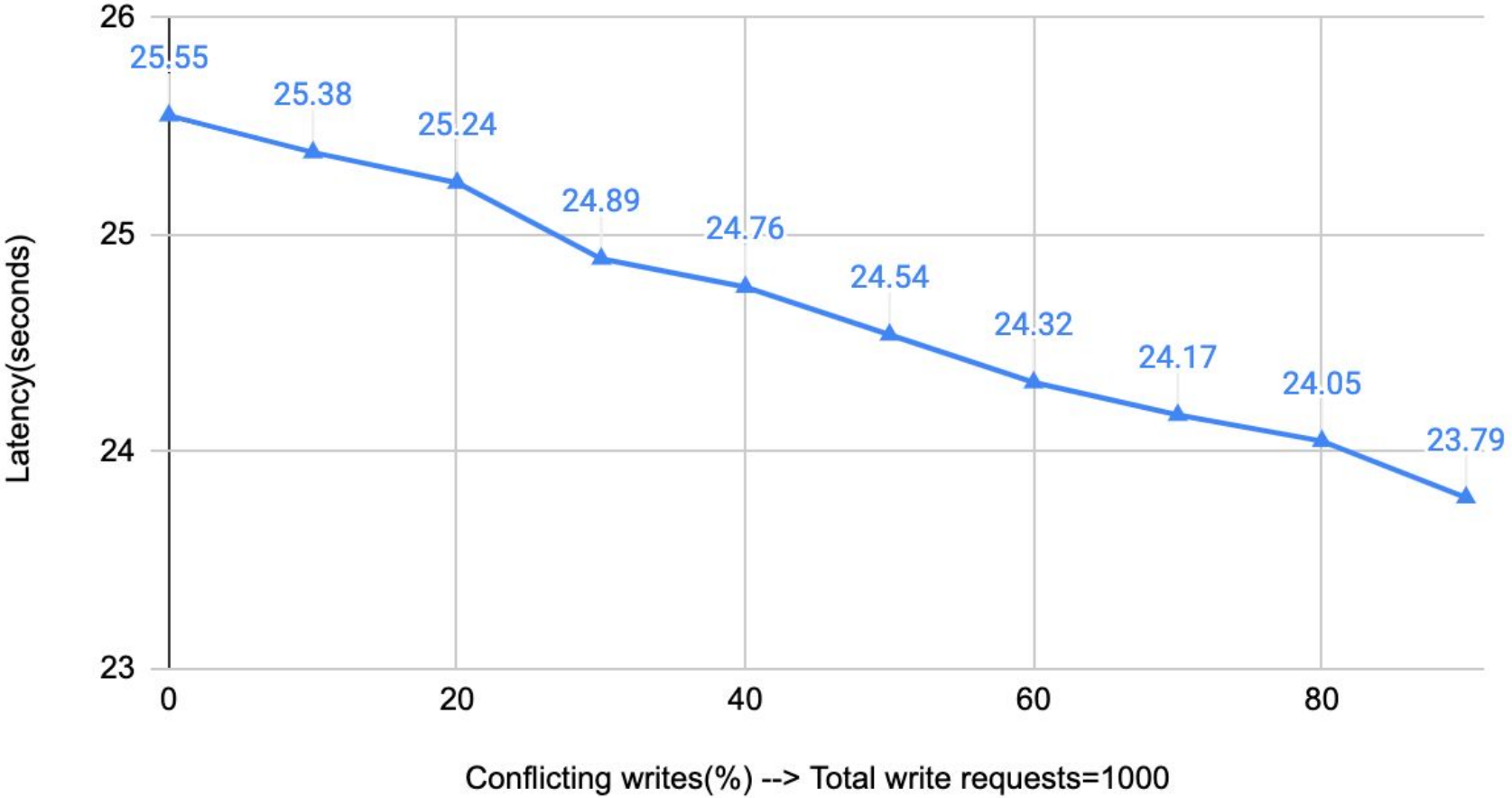


- We don’t observe a significant difference in read and write latencies as expected

Server state transition (observed write latencies from client side)



Latency vs Conflicting writes - Backup availability = 90%



Thank you!