The Google File System

Jason Bury

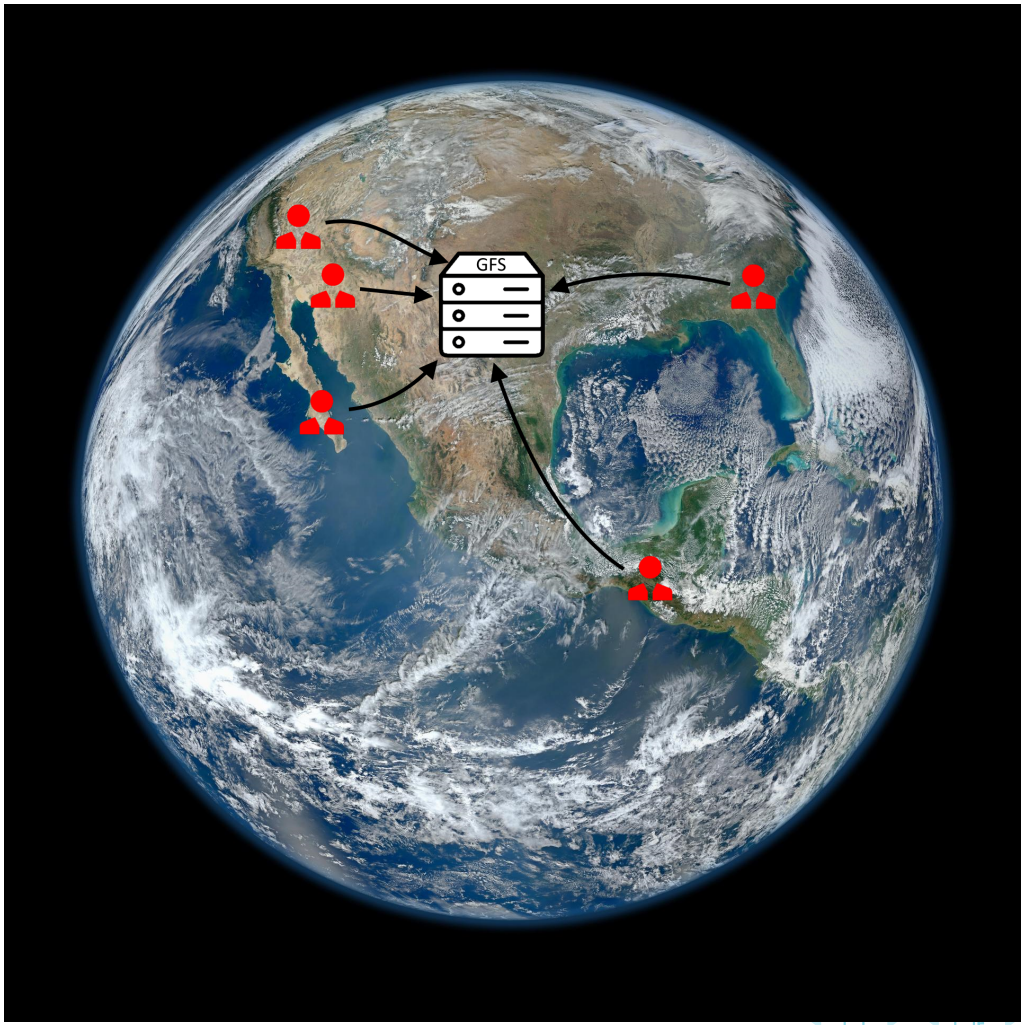
Faculté des Sciences
Université de Mons

UMONS Faculté des Sciences

April 15, 2018

The Google File System

One file system for thousands users



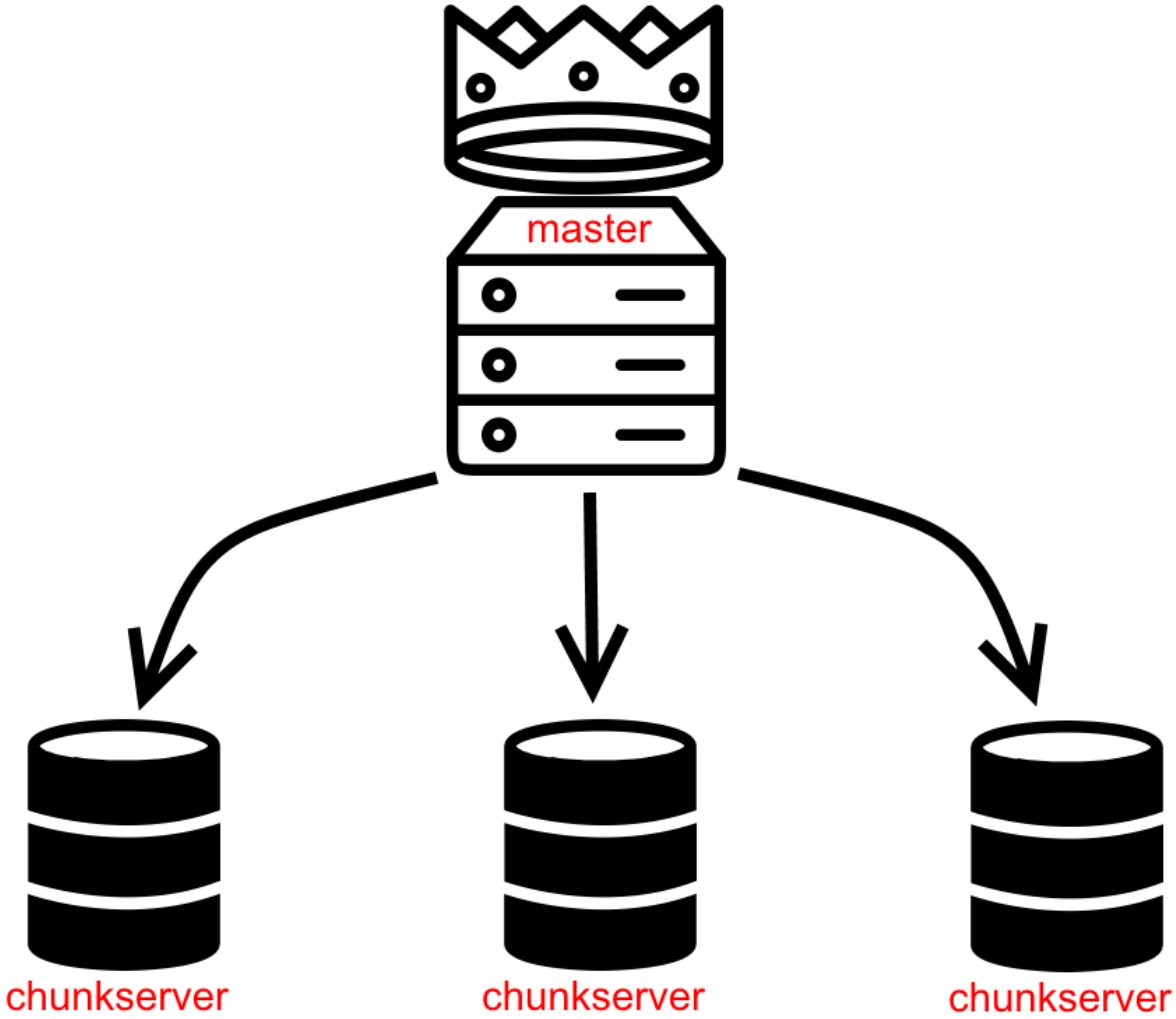
The problems

-  Growing demands of data processing
-  Component failures
-  Huge files
-  Many appends
-  Large streaming reads

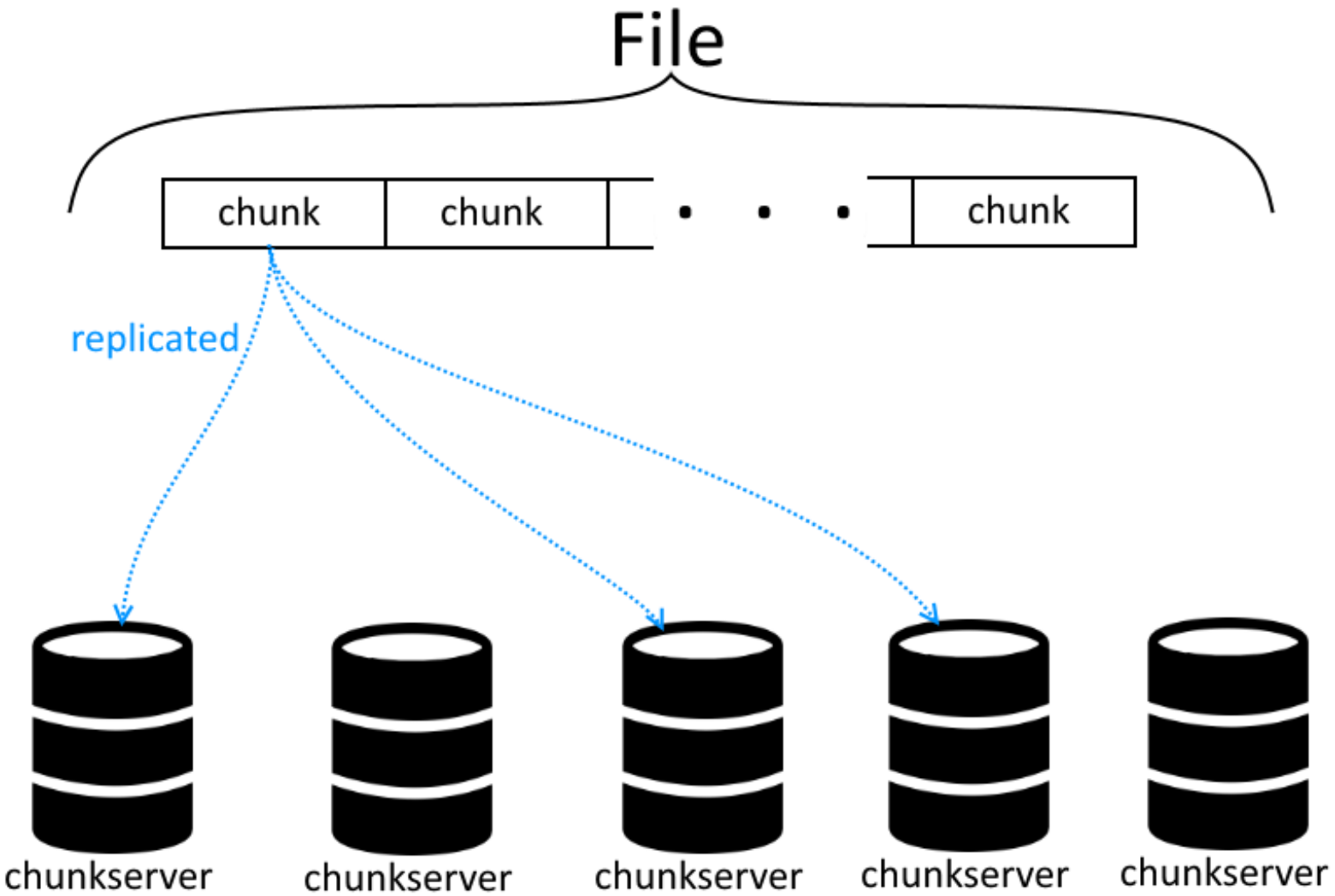
Fault tolerance

- Data replication
- Fast recovery (fast reboot)
- Log system
- Data integrity (checksum + version number)

One Master, thousands chunkservers

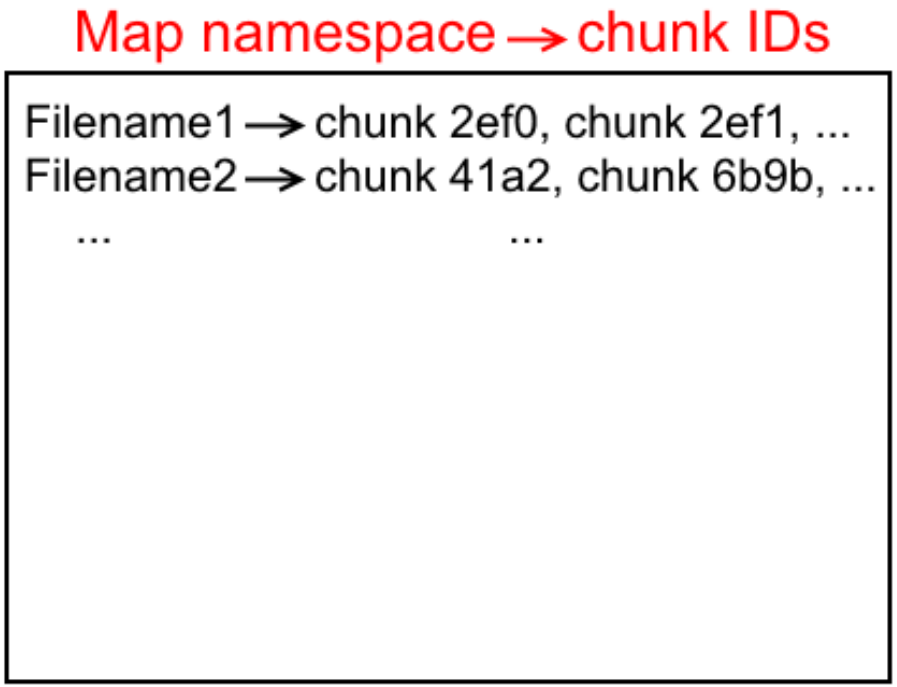


A file


File
chunk
chunk
chunk
replicated
chunkserver
chunkserver
chunkserver
chunkserver
chunkserver

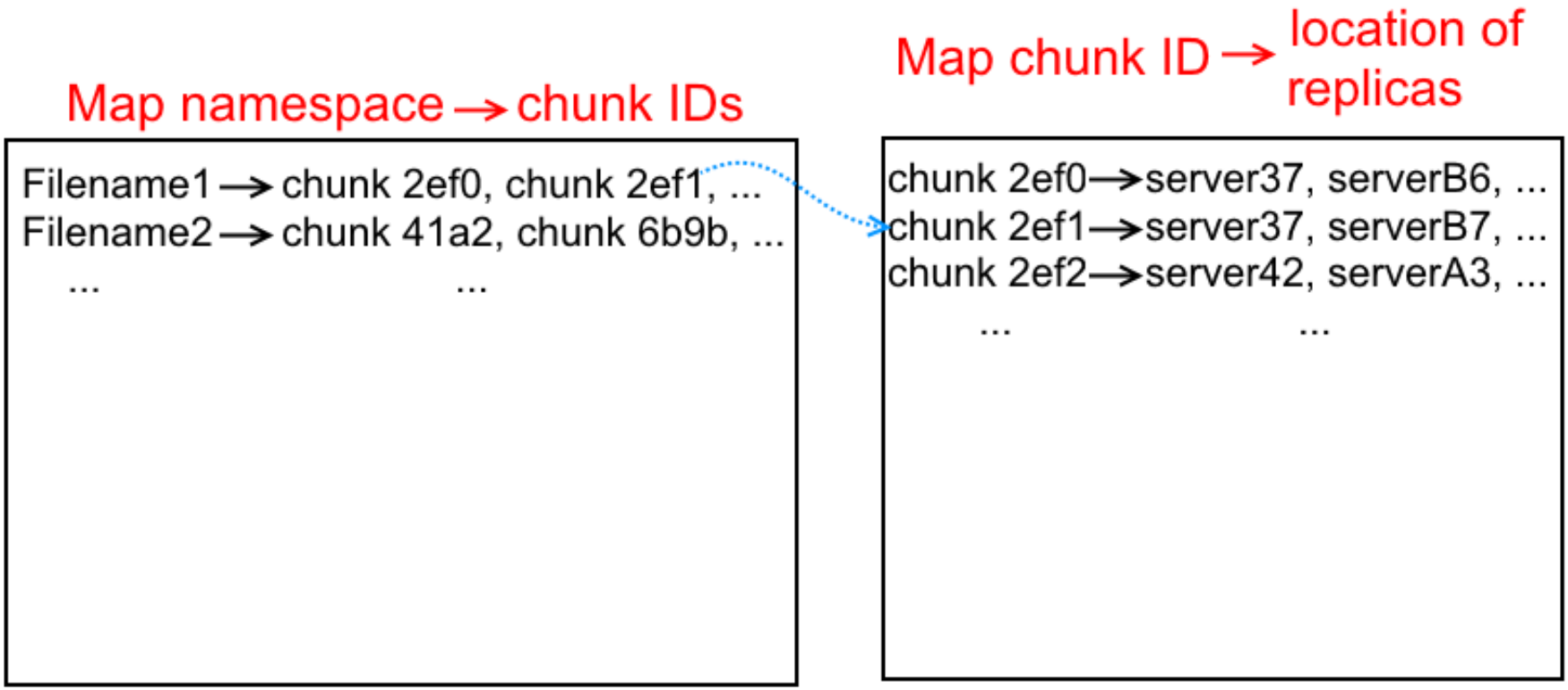
What is in the master ?

The file mapping



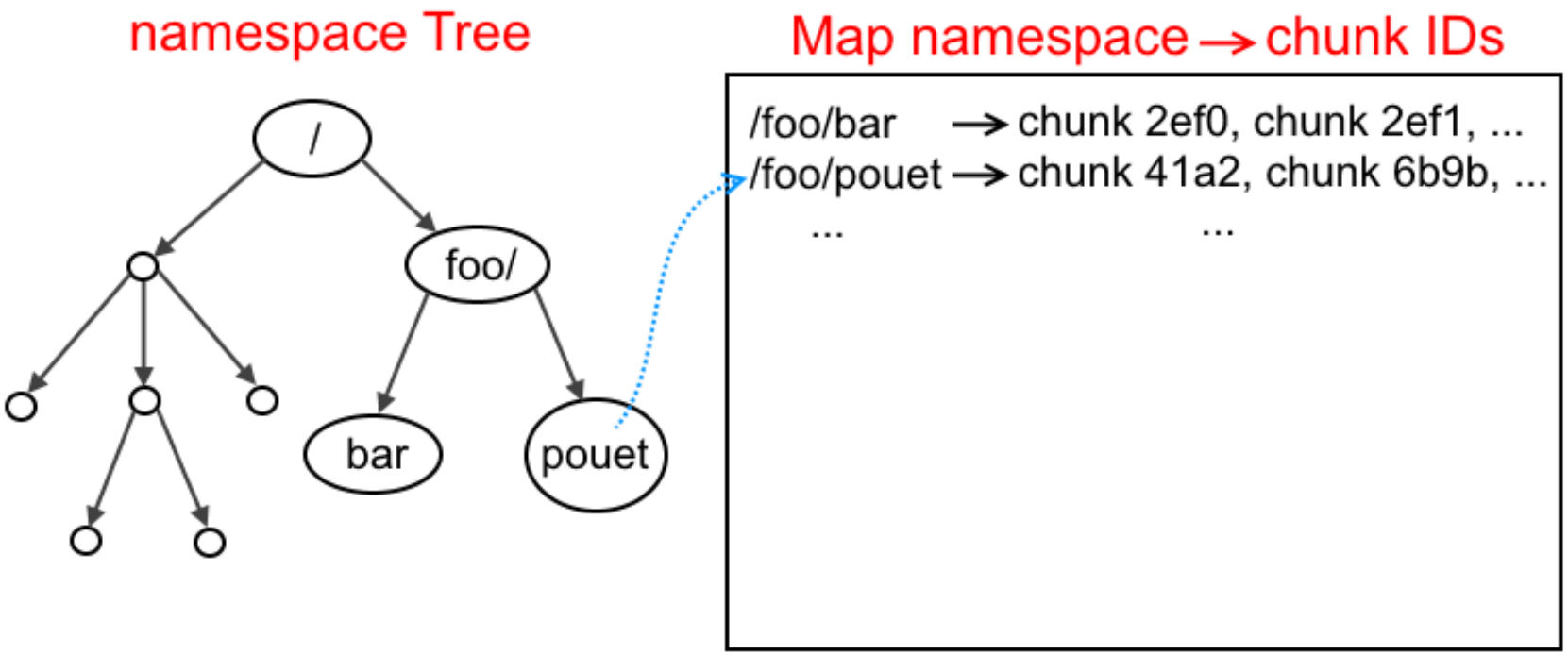
What is in the master ?

The chunk mapping

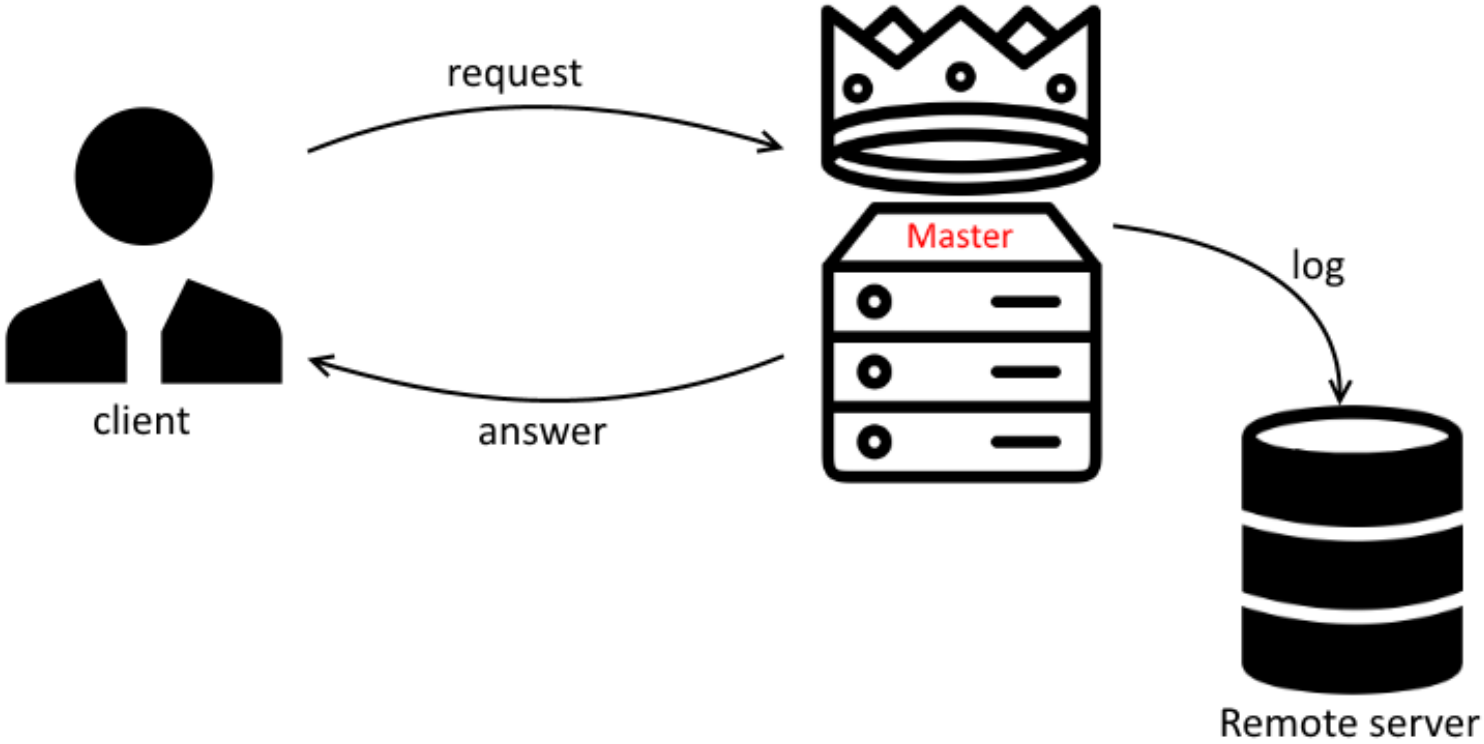


What is in the master ?

The namespace tree



The operations Log

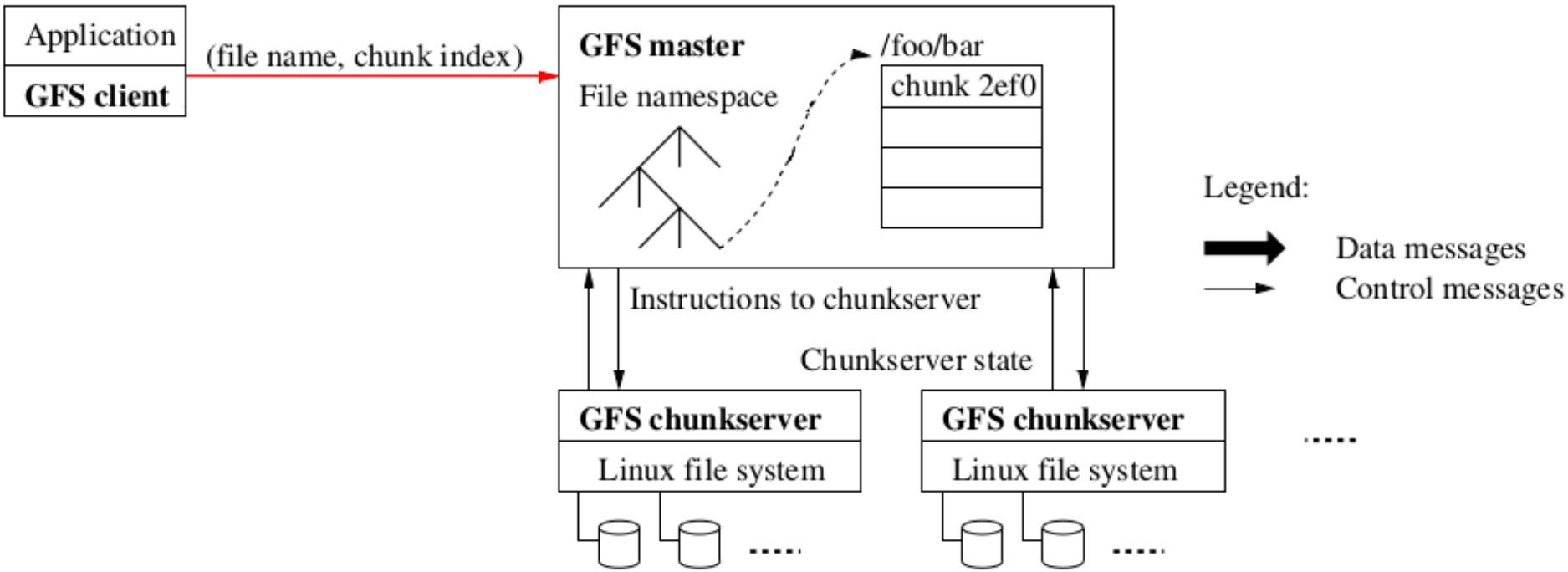


+ checkpoints

⇒ fast recovery

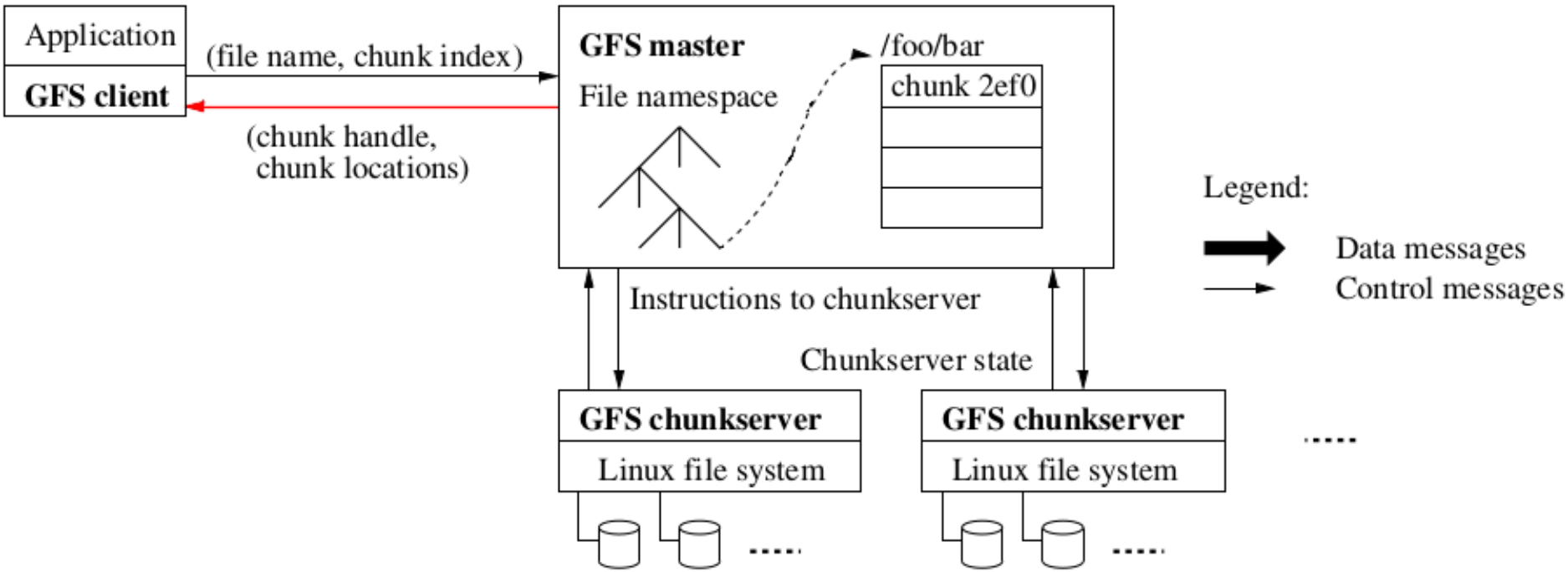
Interactions for a read

Step 1: The client asks to the master replica locations.



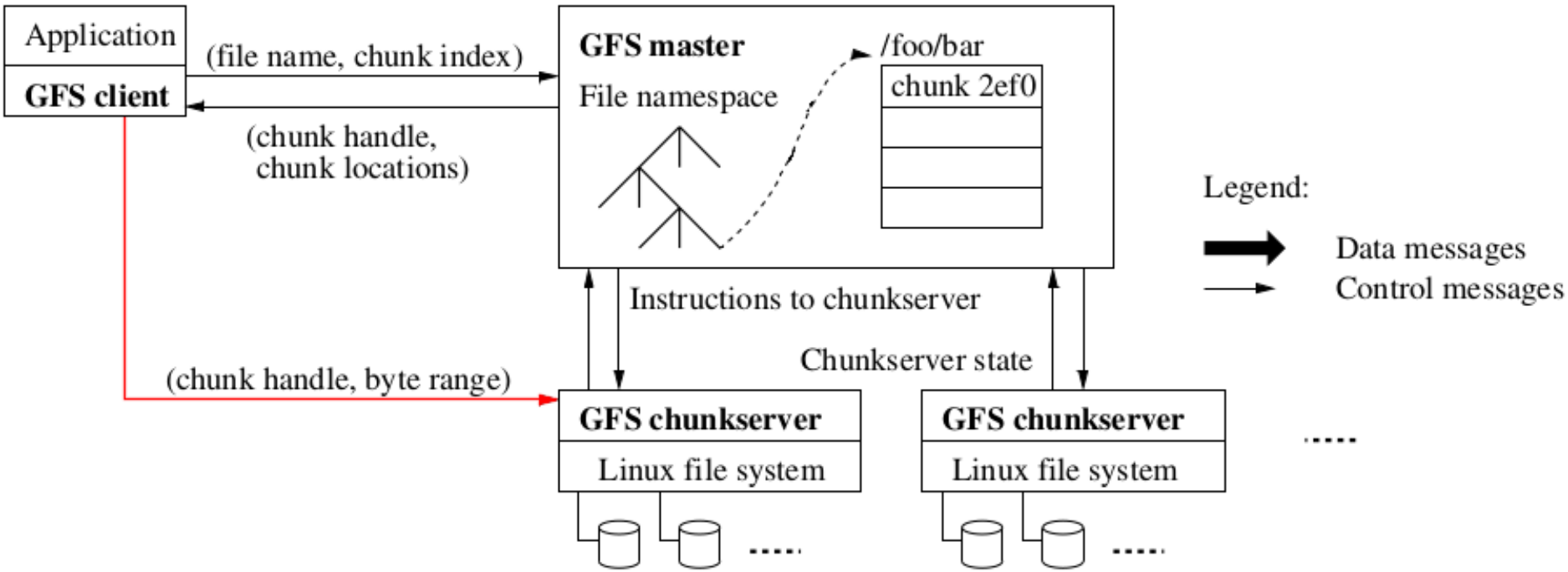
Interactions for a read

Step 2: The master gives location of all replicas.



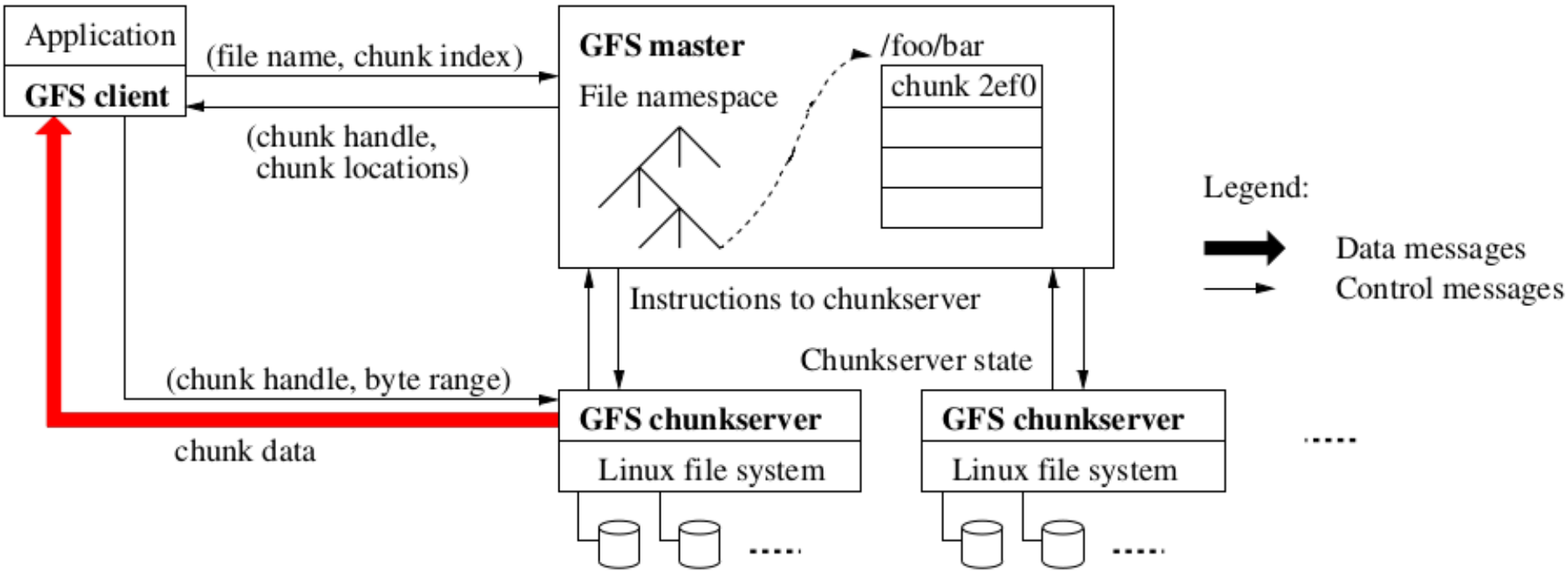
Interactions for a read

Step 3: The client sends the read request to a chunkserver.



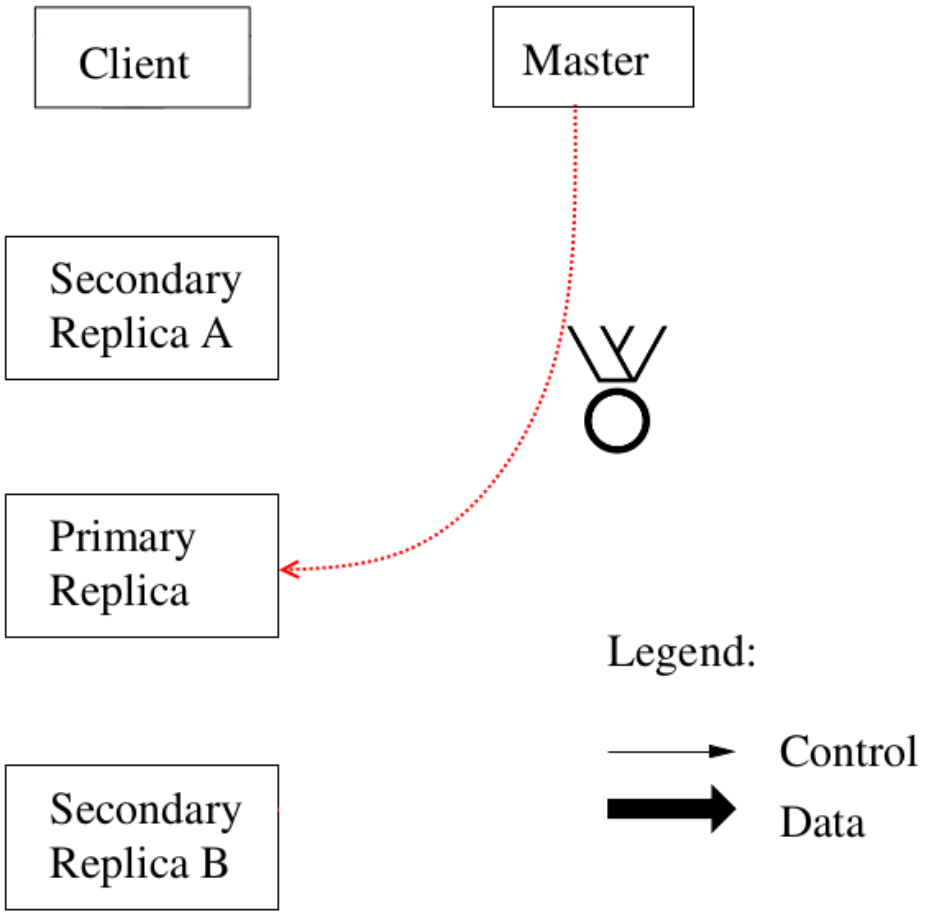
Interactions for a read

Step 4: The data stream starts.



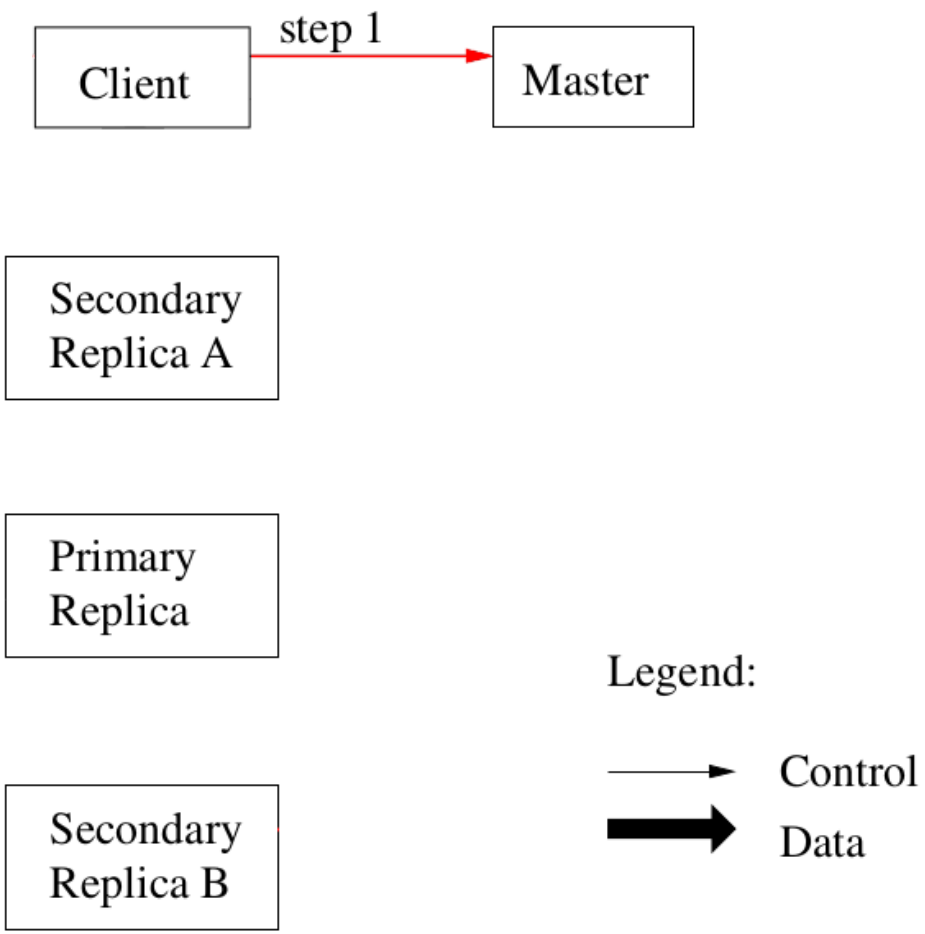
Interactions for a write

Step 0: The master grants a replica.



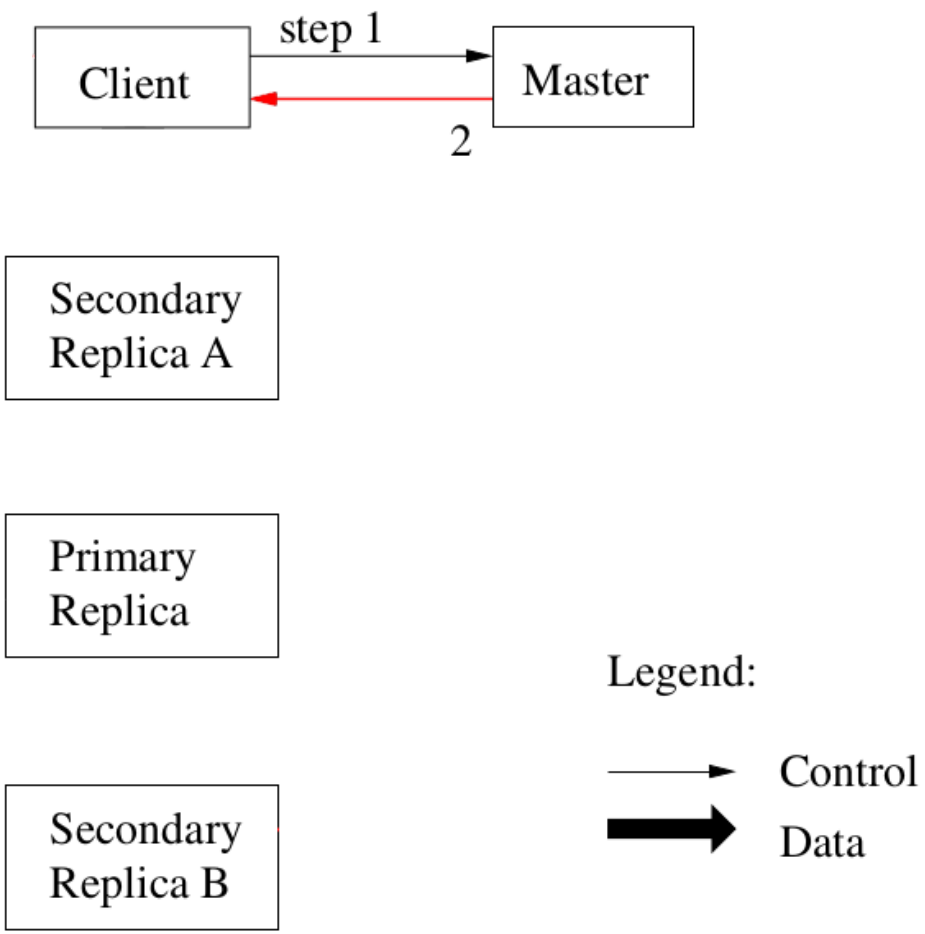
Interactions for a write

Step 1: The client asks which is the primary.



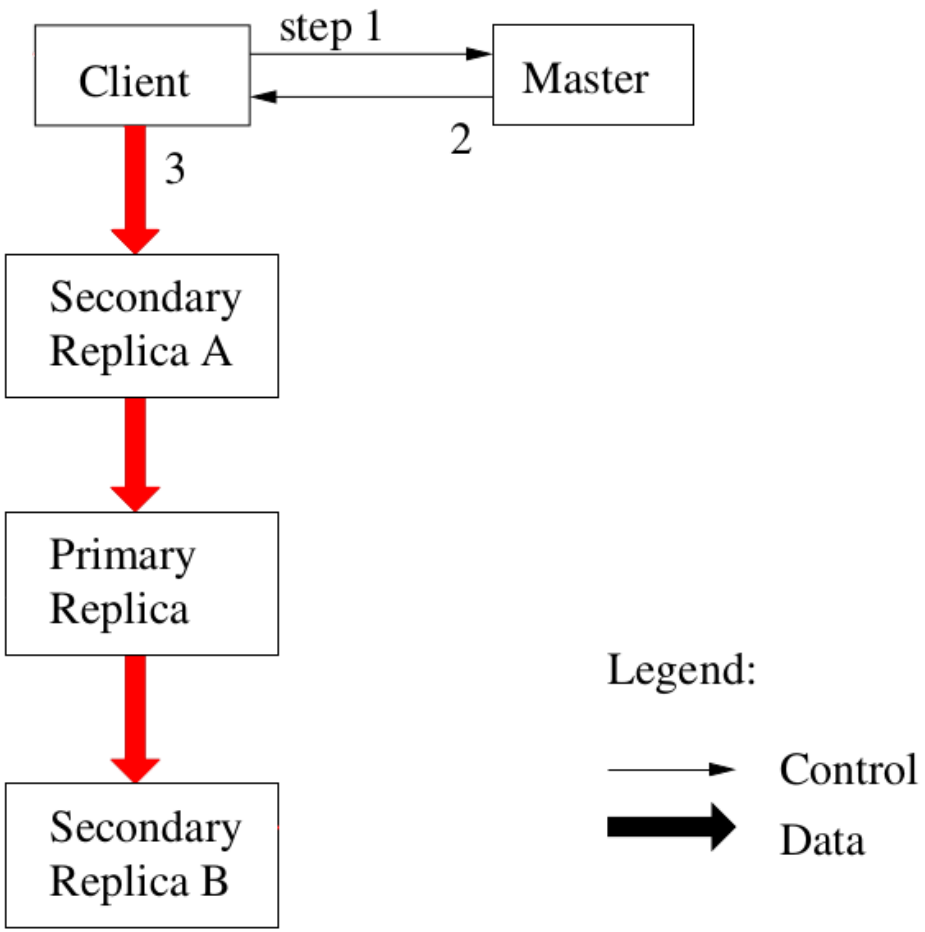
Interactions for a write

Step 2: The master responds.



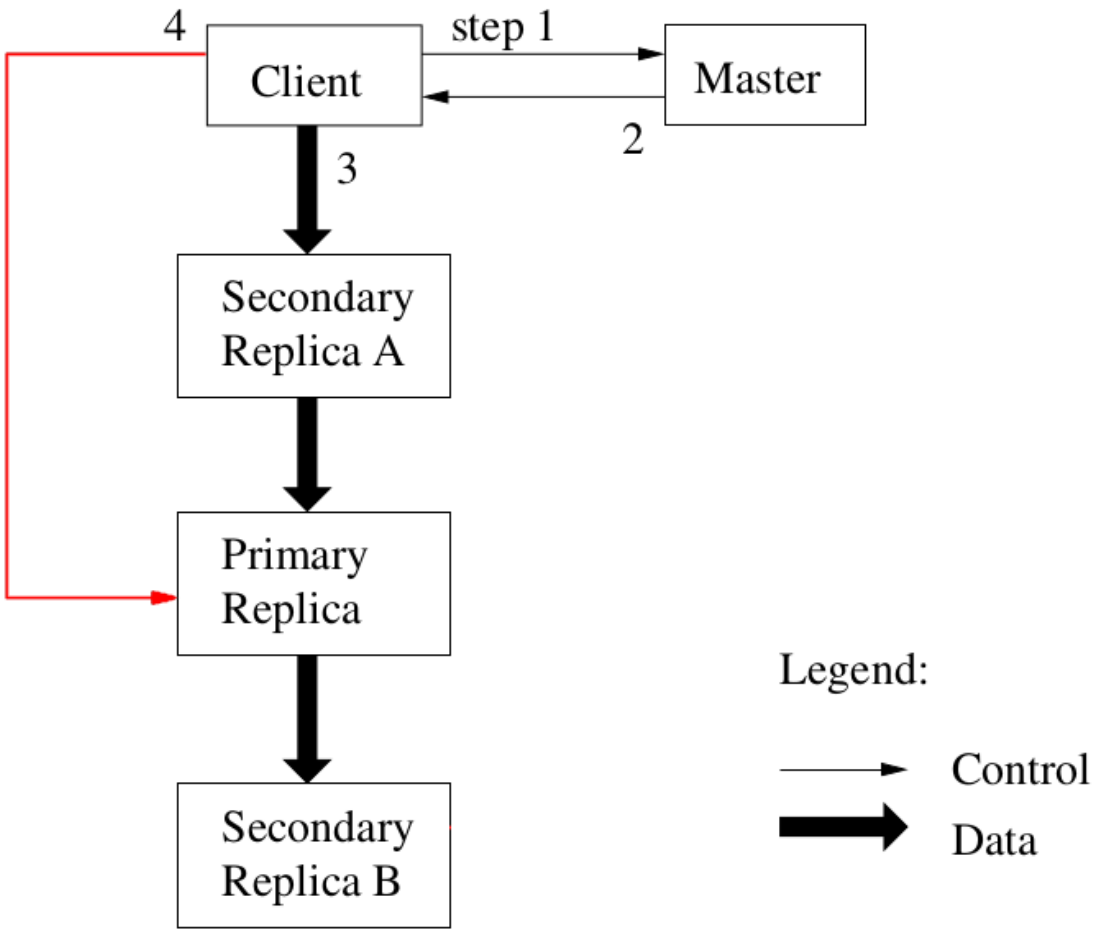
Interactions for a write

Step 3: The data is transfered to chunkservers.



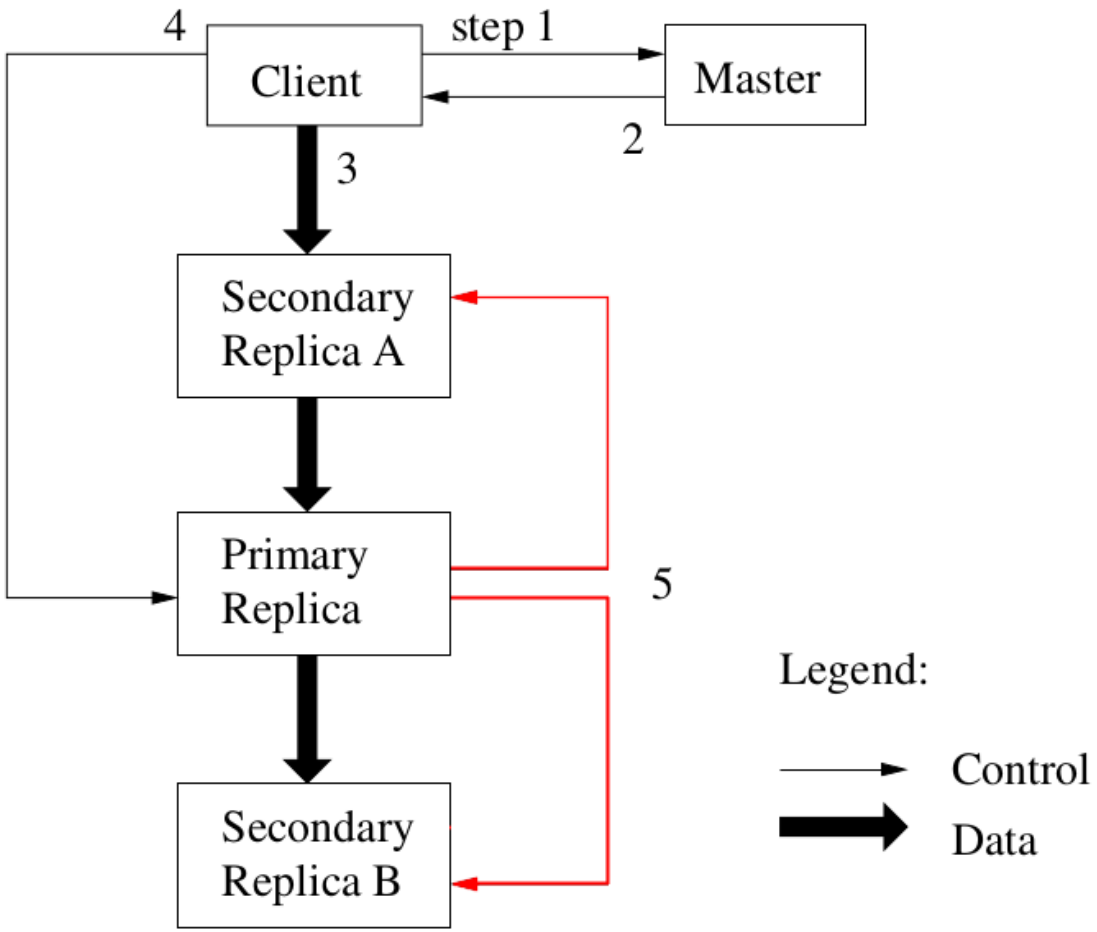
Interactions for a write

Step 4: The client sends a write request to the primary.



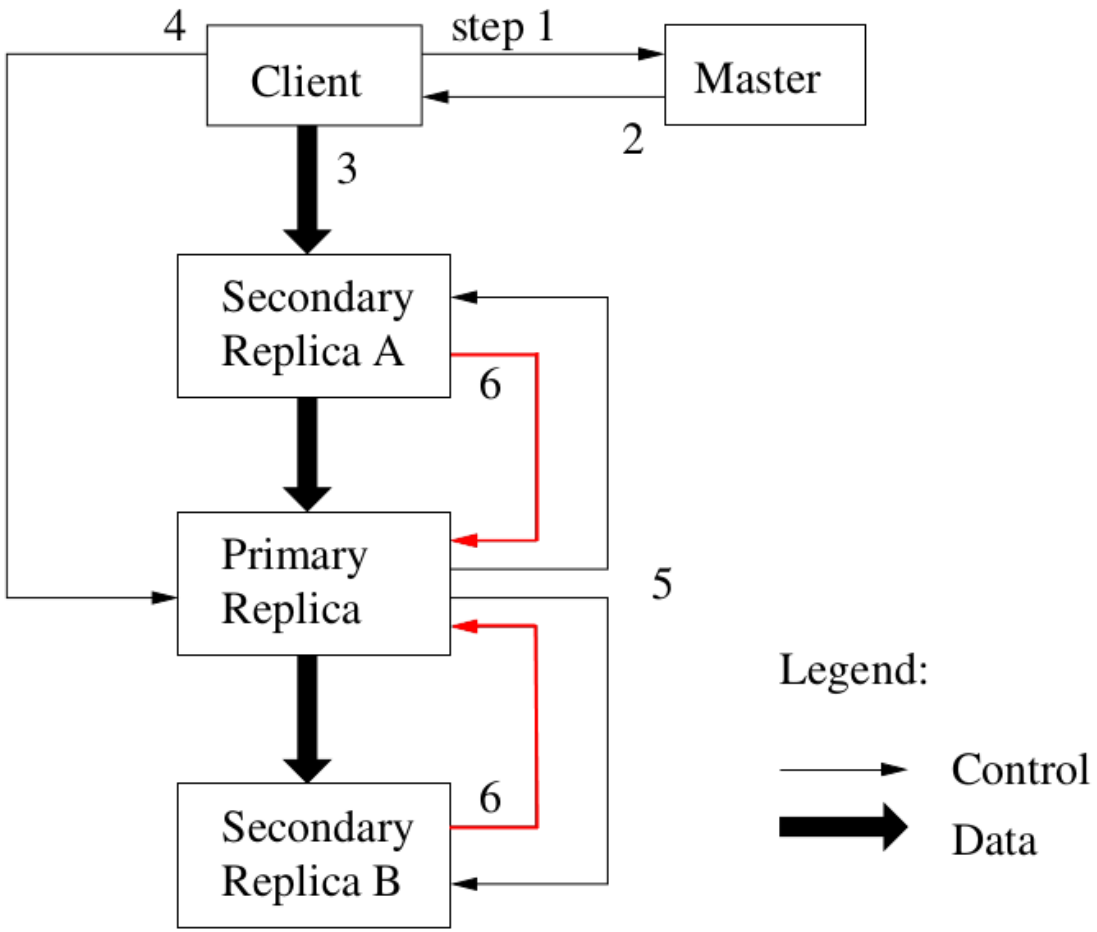
Interactions for a write

Step 5: The primary sends a write request to secondaries.



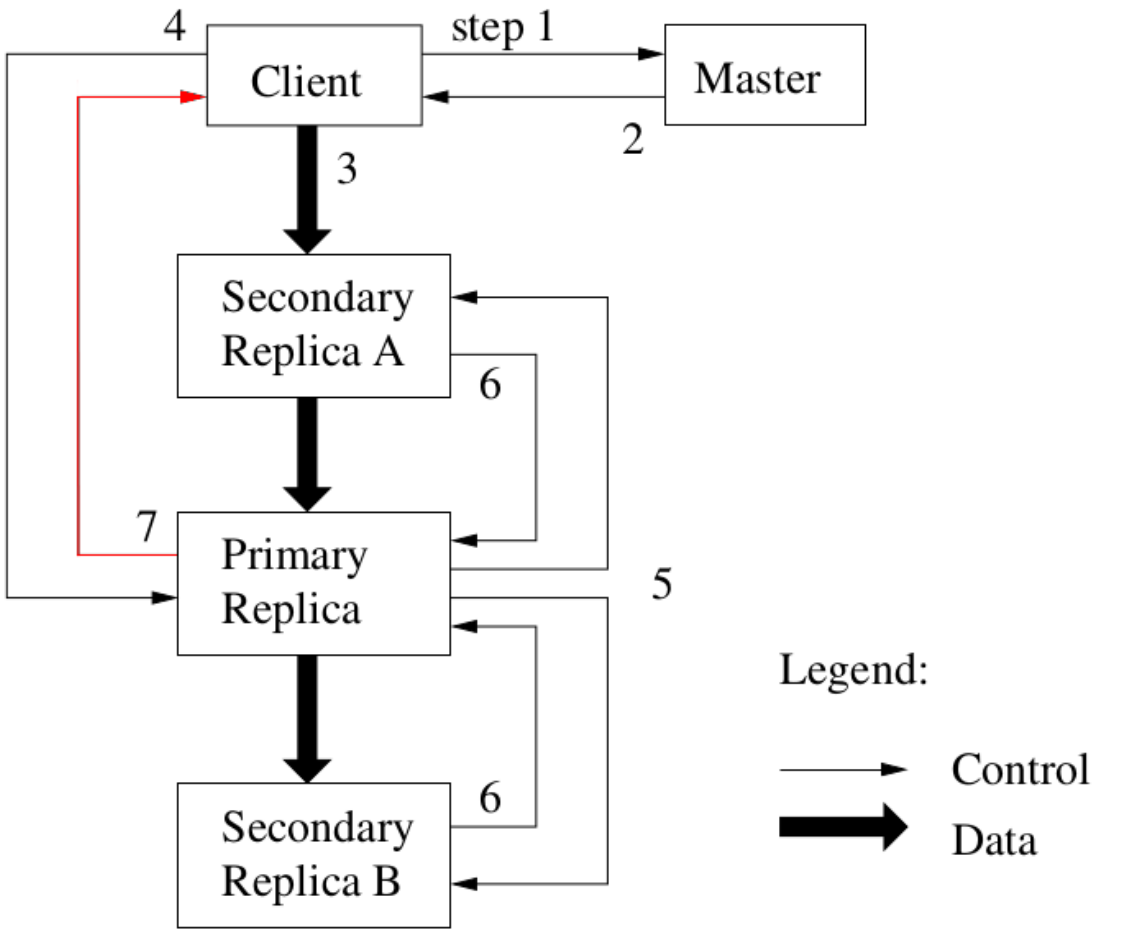
Interactions for a write

Step 6: The secondaries indicates failure or success to the primary.



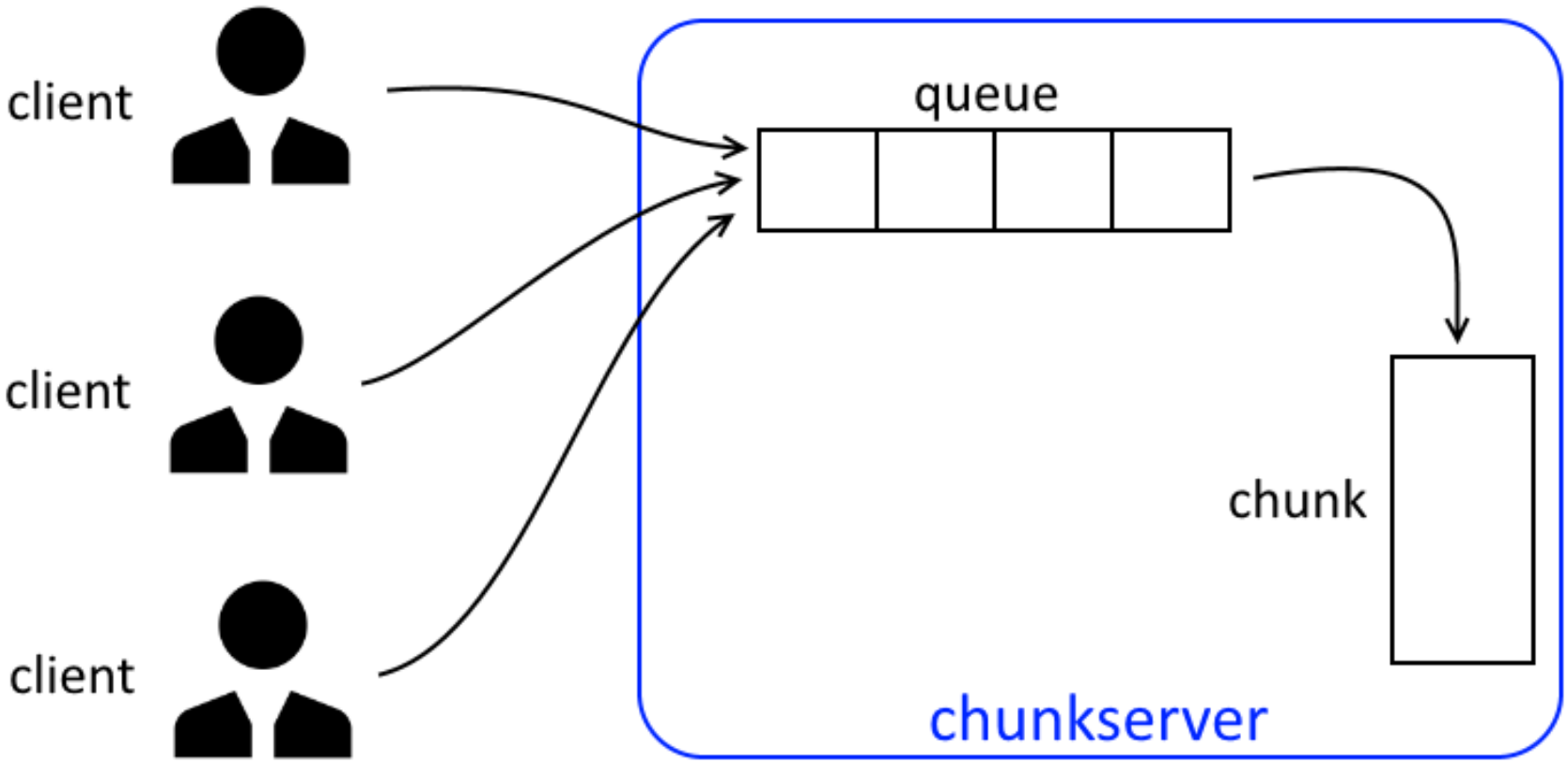
Interactions for a write

Step 7: The primary indicates failure or success to the client.



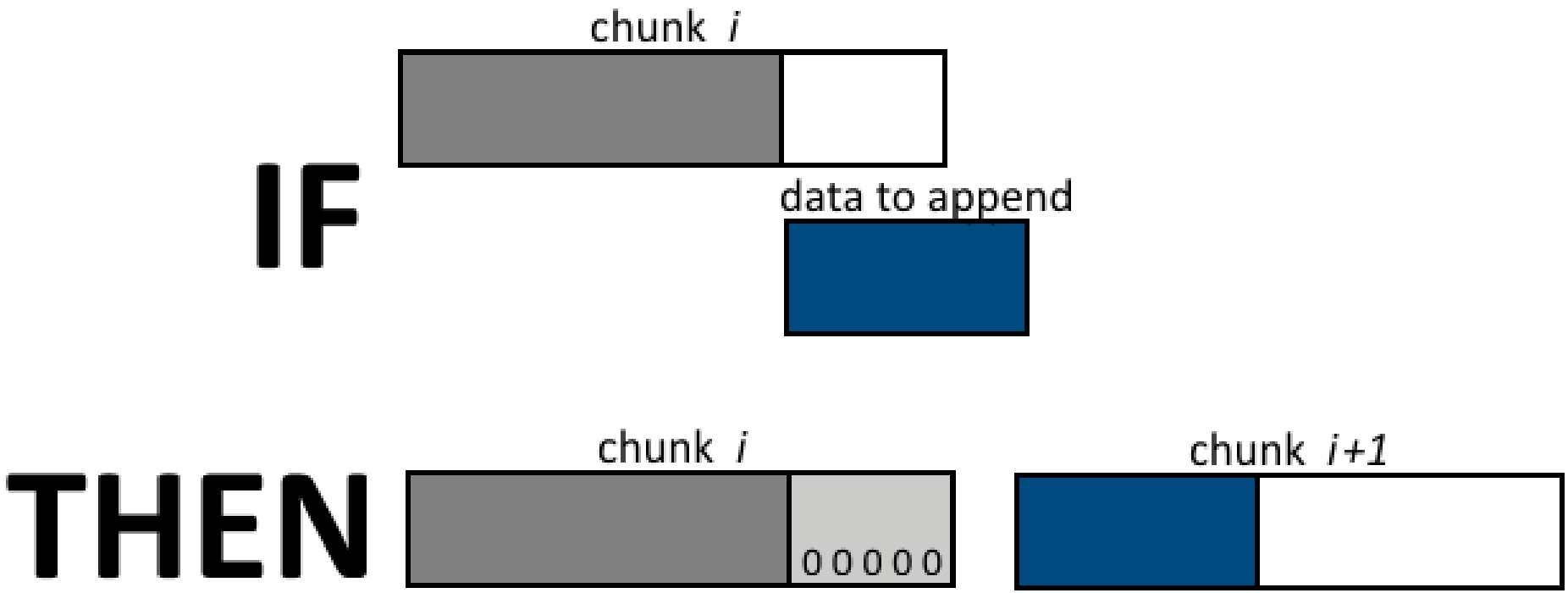
The append operation

Concurrent appends in one operation



The append operation

Data that exceeds chunk

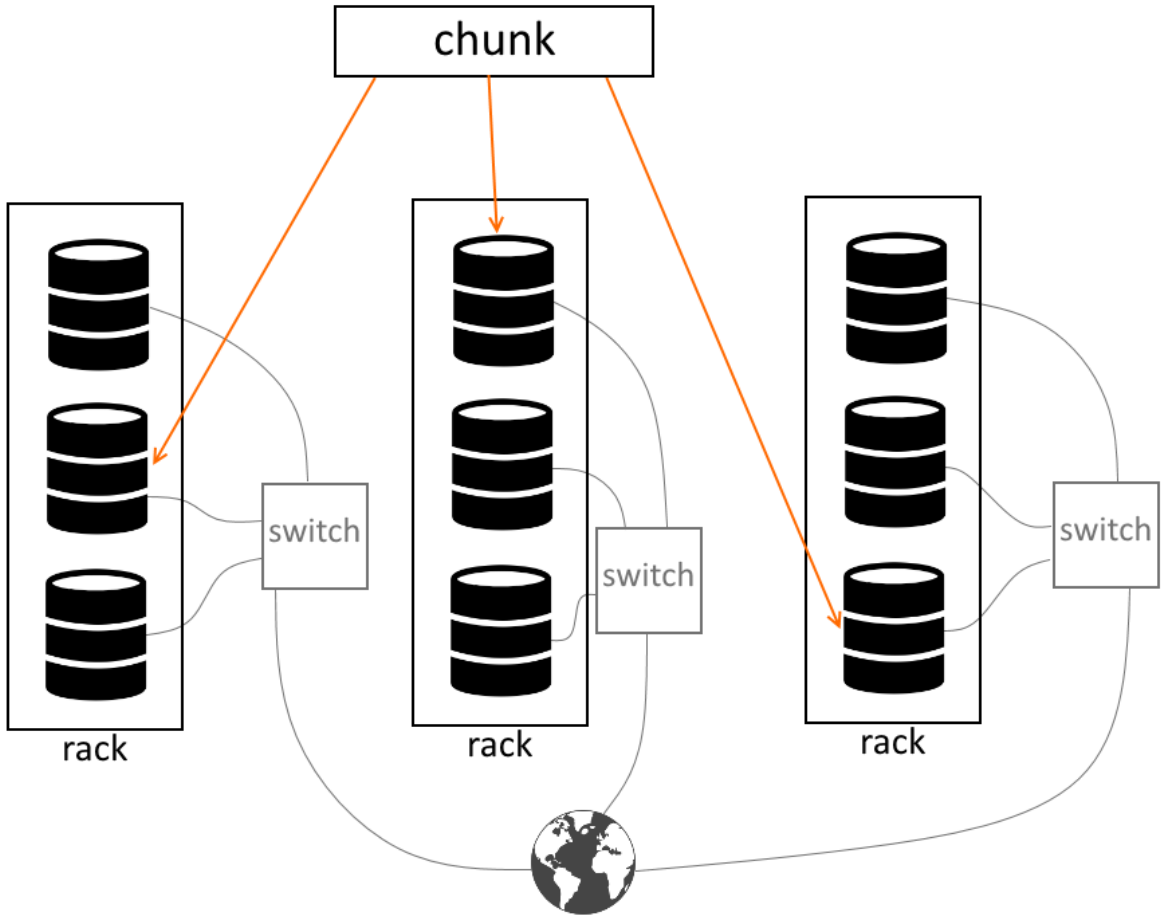


Master operations

- Replica placement
- Re-replica (creation and copy of replica)
- Rebalancing (displacement of replica)
- Garbage collection

Replica placement

In different racks



Re-replica and rebalancing

There are 3 reasons that a replica is created:

1. The chunk is just created.
2. The number of replicas is below an user-specified goal.
3. The master moves it for better load balancing.

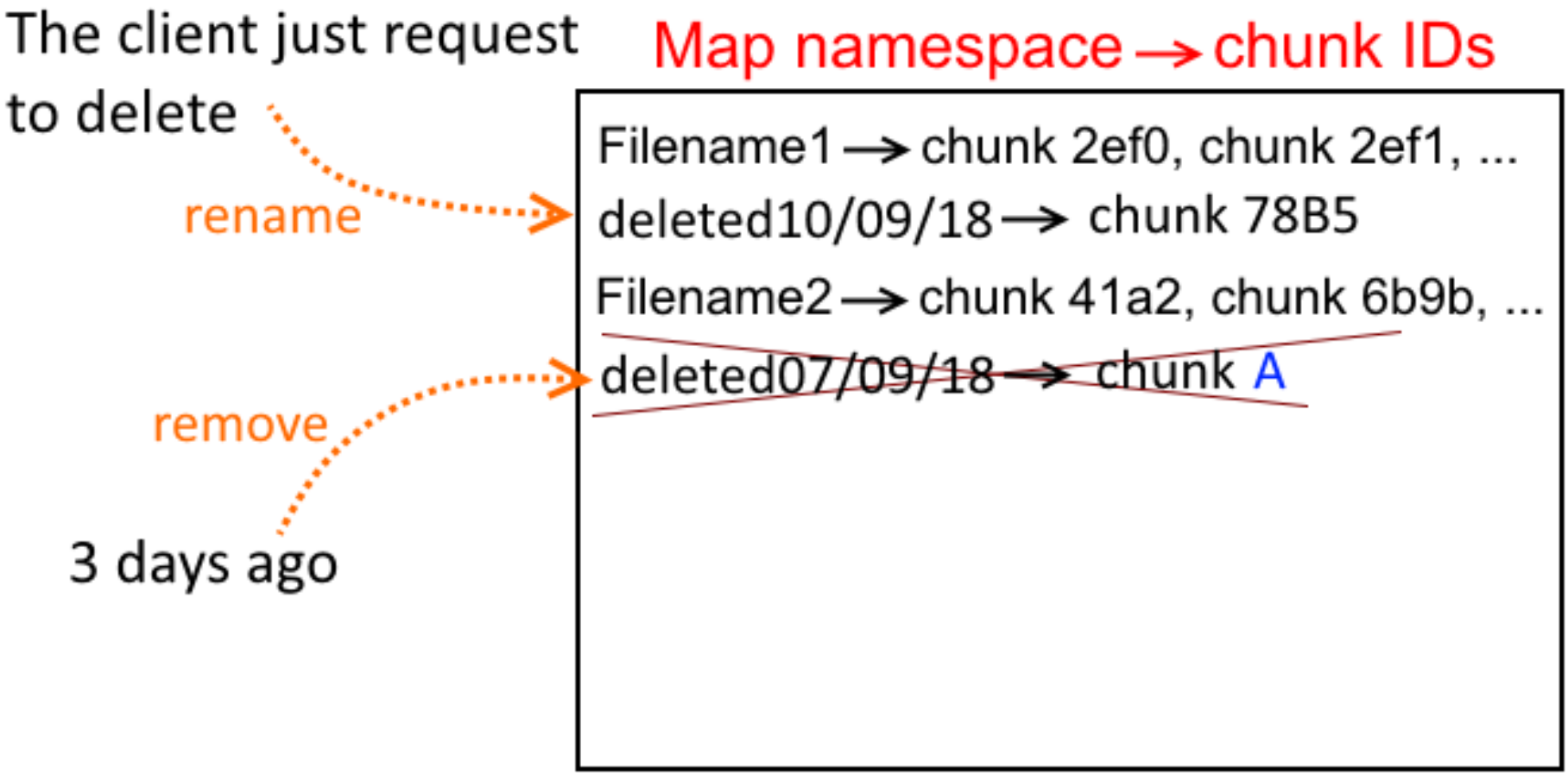
Garbage collection

It cleans up:

- Chunk replicas deleted by a client
- Corrupted replicas using checksum
- Stale replicas using version number
- Orphan chunks

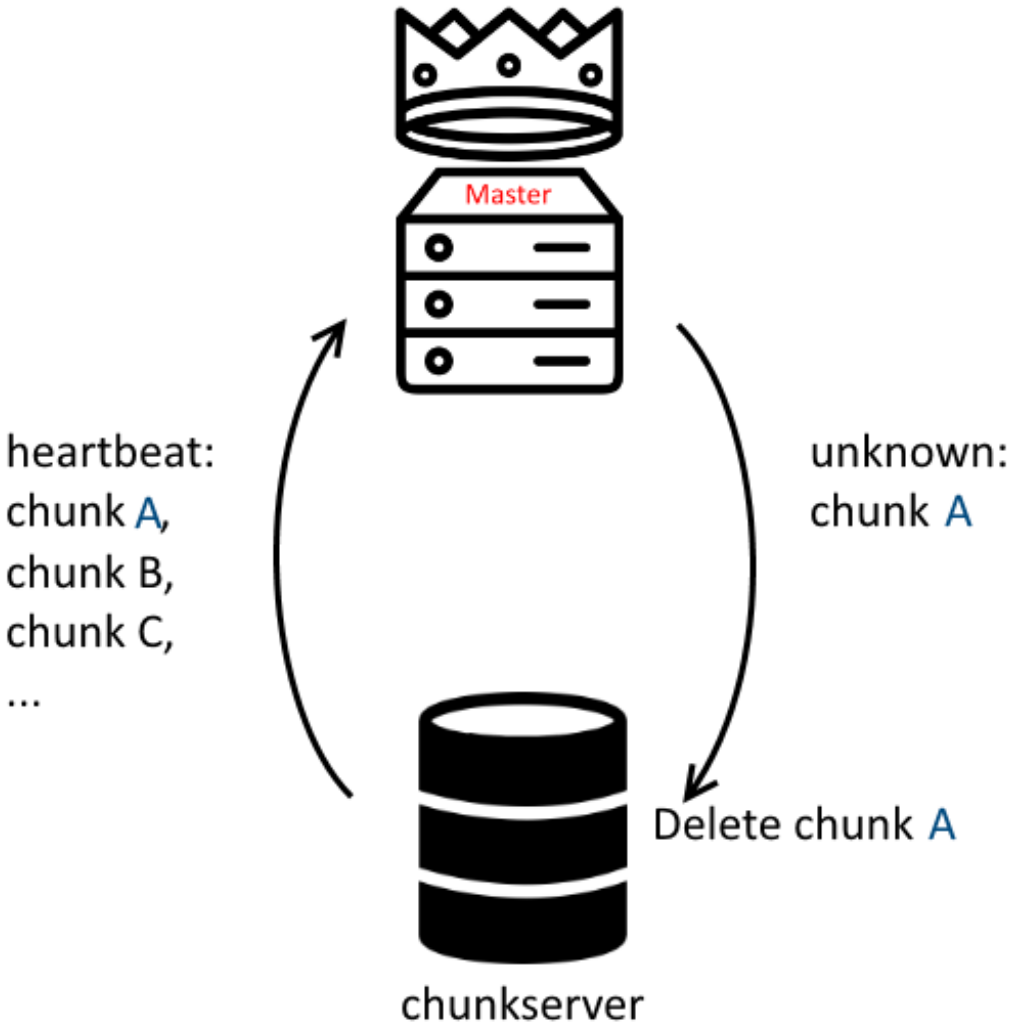
Garbage collection

Removing file



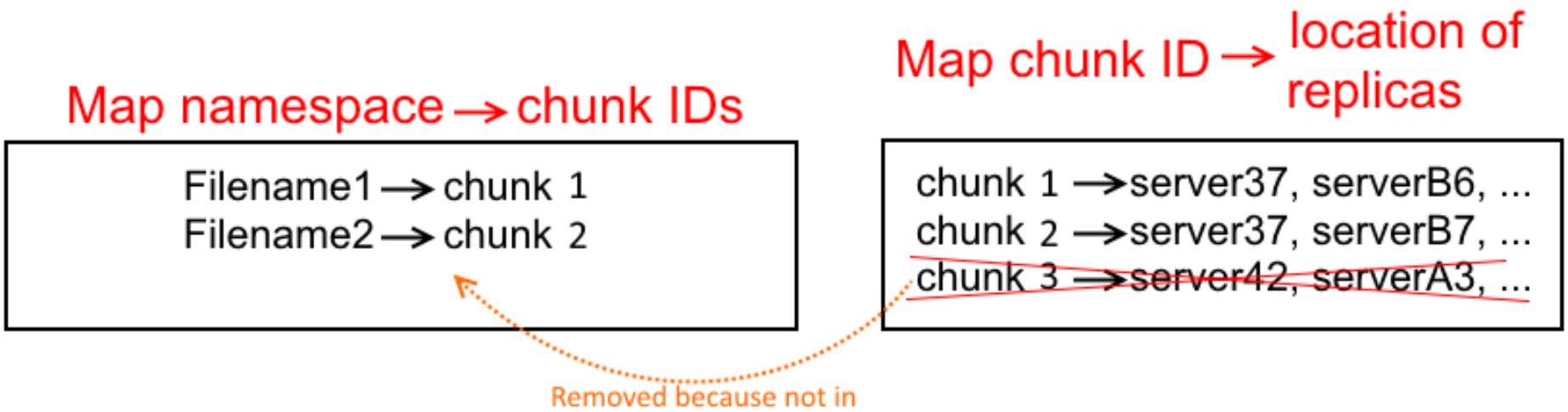
Garbage collection

Recover memory



Garbage collection

Orphan chunks



Stale replica detection

If mutation missed

- $\Rightarrow$ Stale replica
- $\Rightarrow$ Version number not updated
- $\Rightarrow$ Detected during garbage collection or chunserver reboot

Corrupted replica detection

If mutation incorrect

$\Rightarrow$ Corrupted replica

$\Rightarrow$ Checksum not corresponding to data

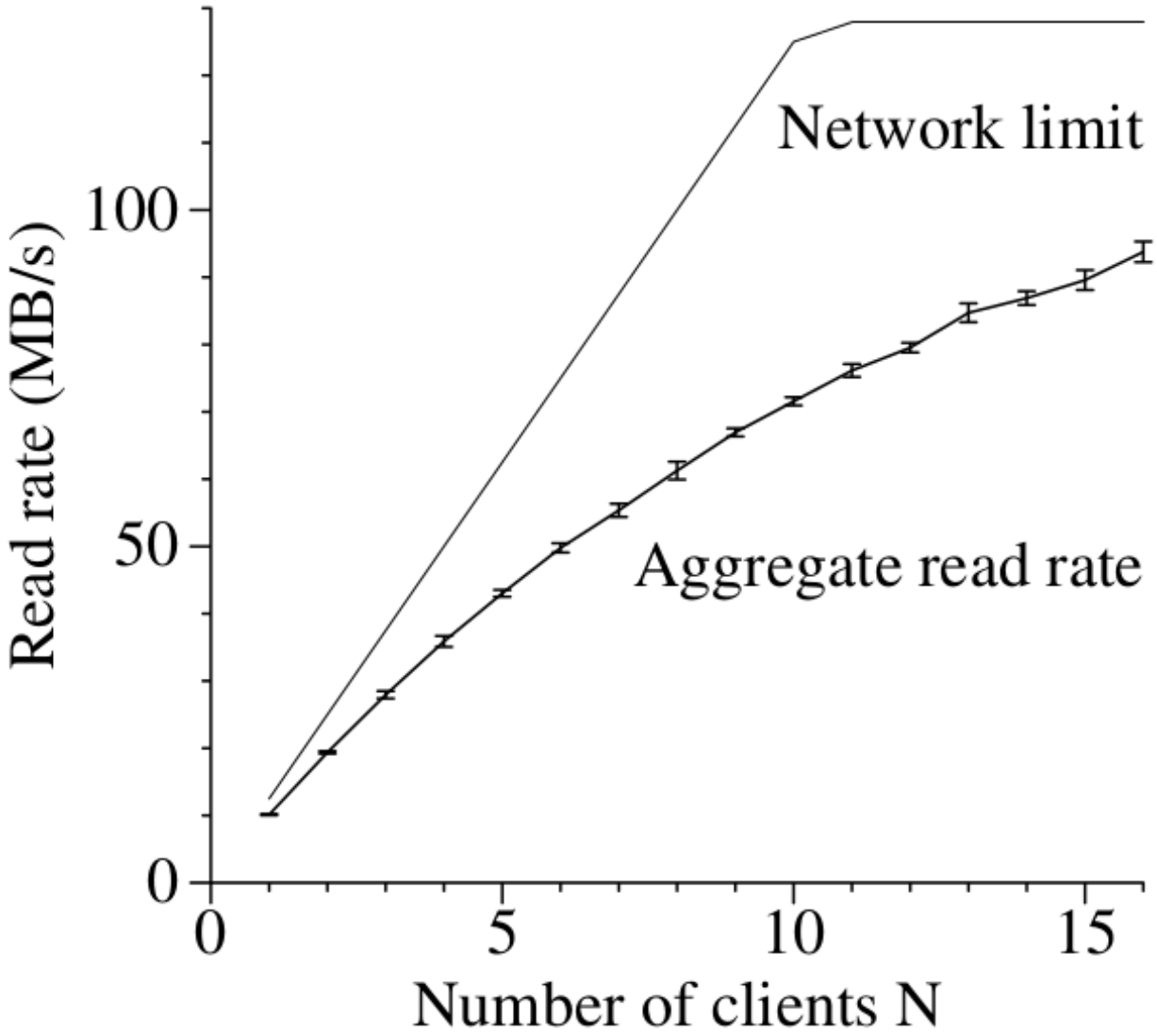
$\Rightarrow$ Detected after a read request

$\Rightarrow$ Error message send

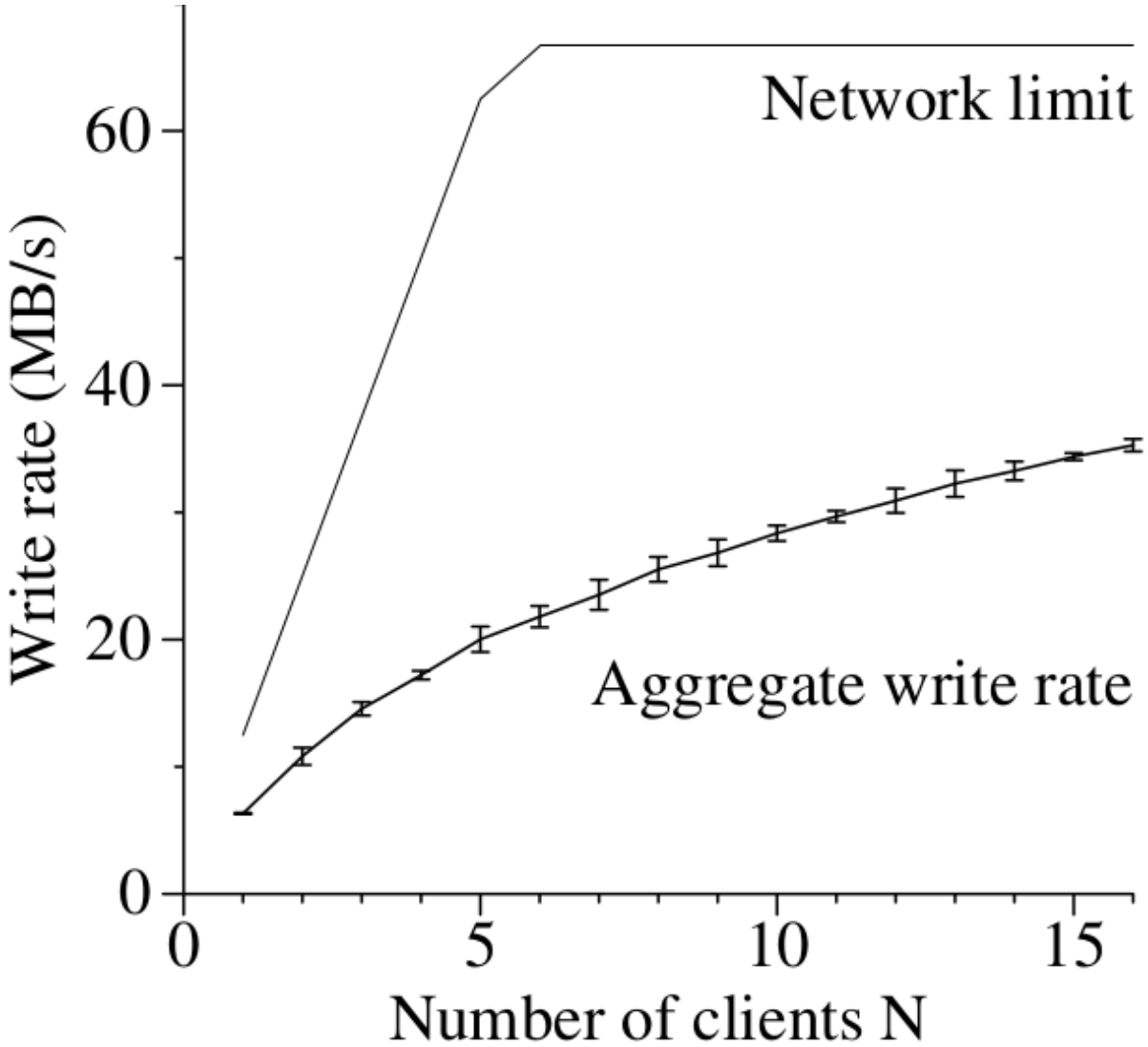
$\Rightarrow$ Master warned

$\Rightarrow$ replica deleted

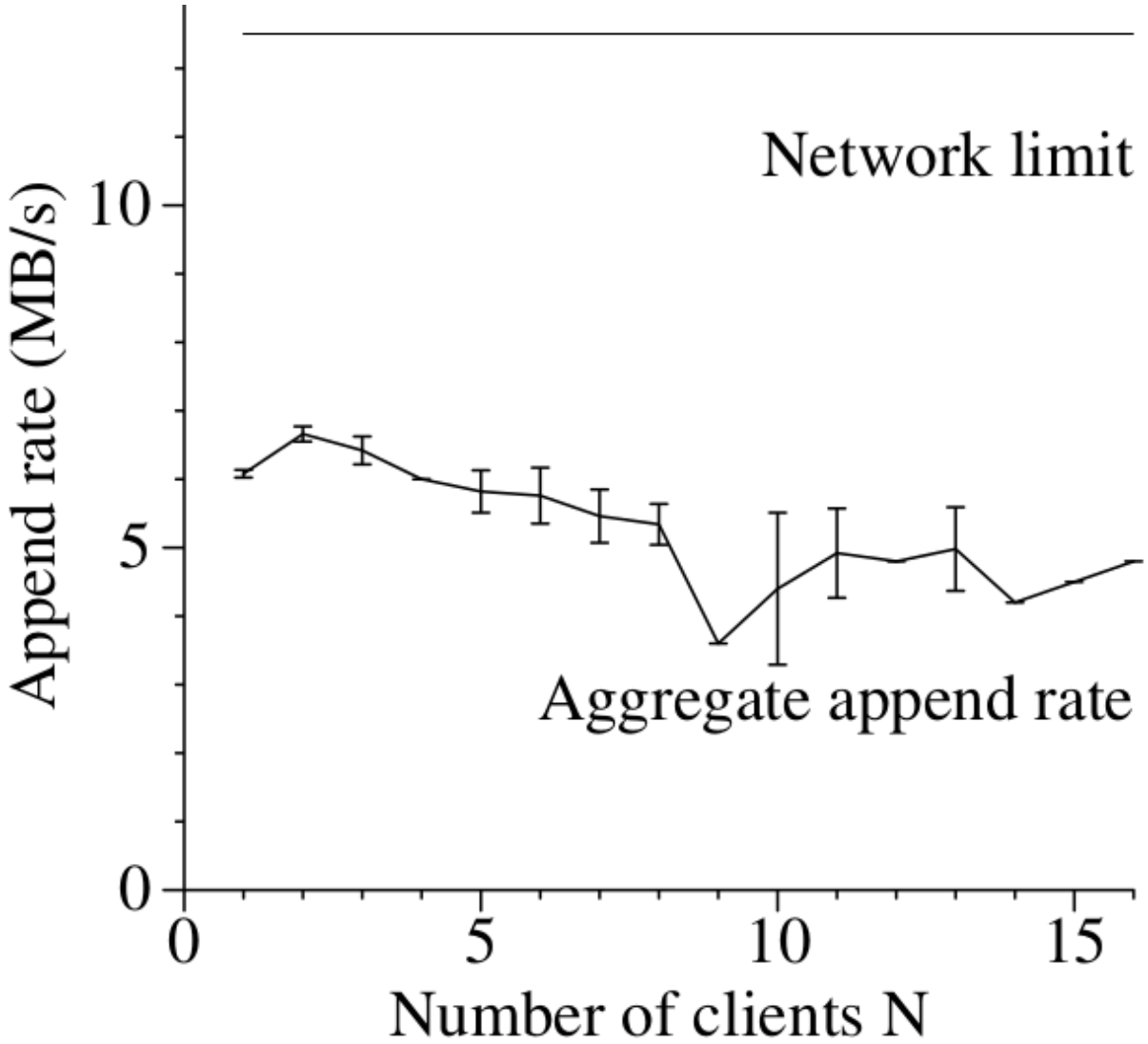
Read rate


Network limit
Aggregate read rate
Read rate (MB/s)
100
50
0
0
5
10
15
Number of clients N

Write rate


Network limit
Aggregate write rate
Write rate (MB/s)
Number of clients N
0
20
40
60
0
5
10
15

Append rate


Network limit
Aggregate append rate
Append rate (MB/s)
Number of clients N
0
5
10
15
0
5
10

Conclusion

- ☑ Fault-tolerant
- ☑ Support huge files
- ☑ Efficient for large streaming reads
- ☑ Efficient for concurrent appends