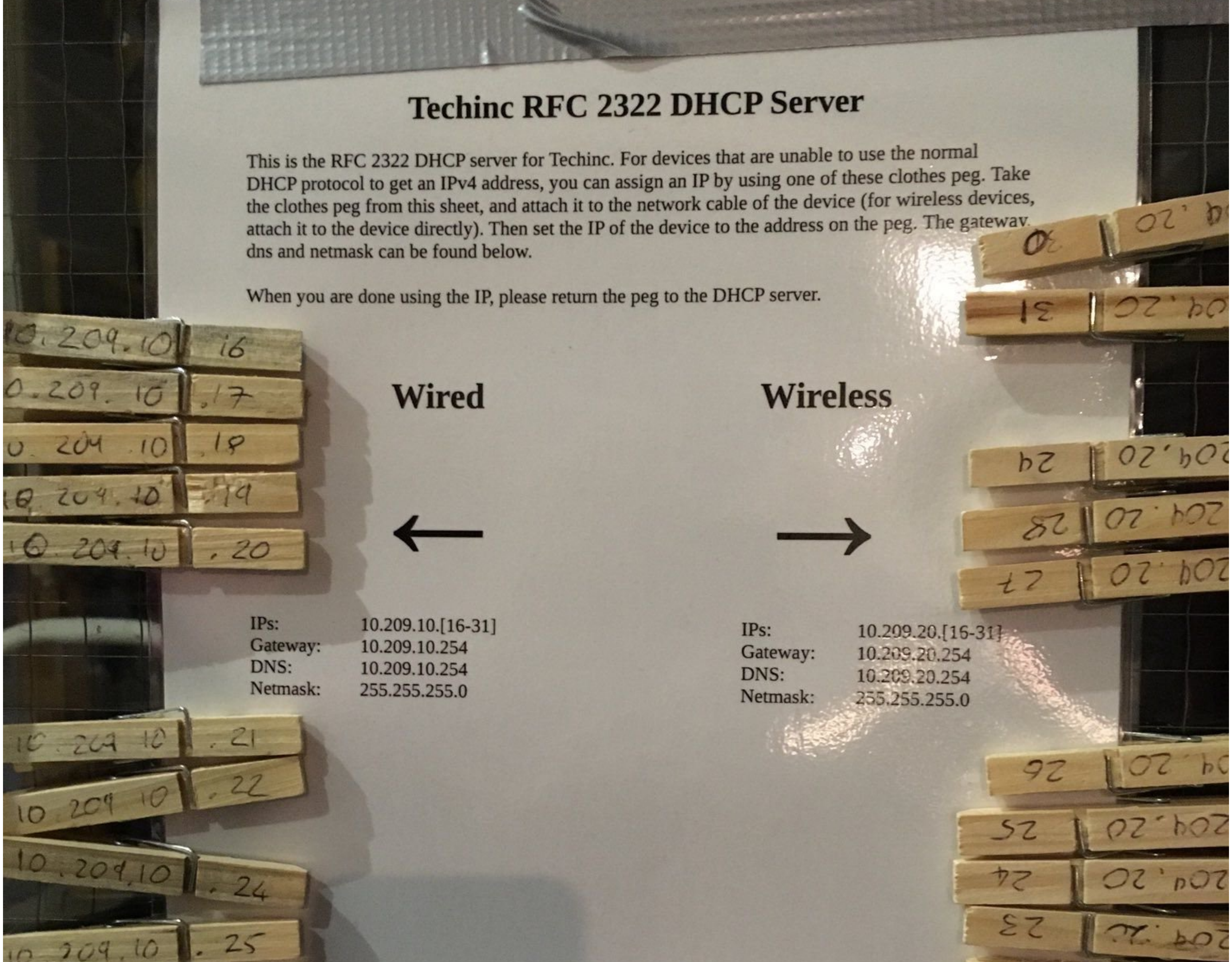
QUORUMS AND THE GOOGLE FILE SYSTEM

George Porter
Feb 8, 2020



Techinc RFC 2322 DHCP Server

This is the RFC 2322 DHCP server for Techinc. For devices that are unable to use the normal DHCP protocol to get an IPv4 address, you can assign an IP by using one of these clothes peg. Take the clothes peg from this sheet, and attach it to the network cable of the device (for wireless devices, attach it to the device directly). Then set the IP of the device to the address on the peg. The gateway, dns and netmask can be found below.

When you are done using the IP, please return the peg to the DHCP server.

Wired ←		Wireless →	
IPs:	10.209.10.[16-31]	IPs:	10.209.20.[16-31]
Gateway:	10.209.10.254	Gateway:	10.209.20.254
DNS:	10.209.10.254	DNS:	10.209.20.254
Netmask:	255.255.255.0	Netmask:	255.255.255.0

ATTRIBUTION

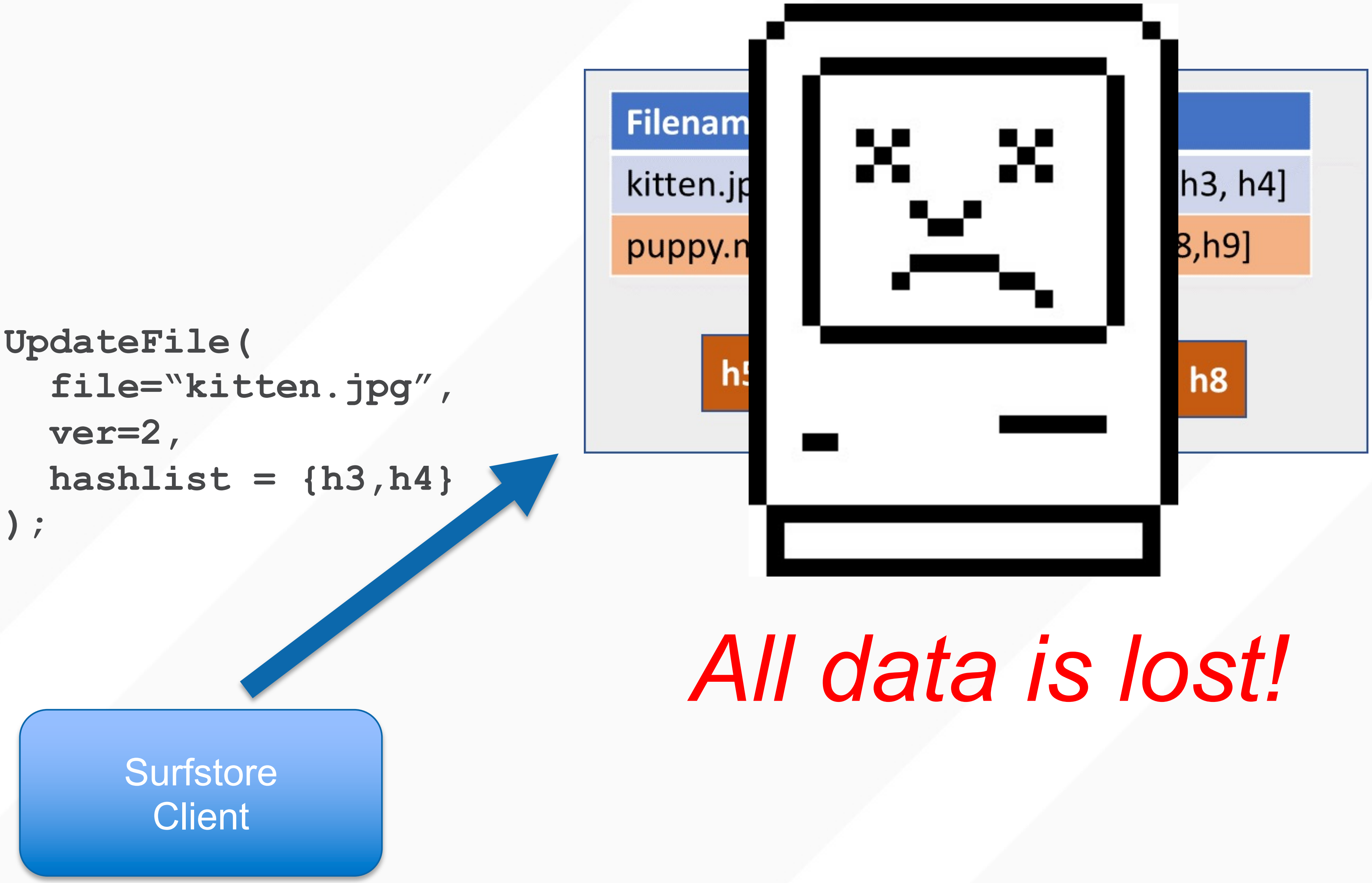
UC San Diego

OUTLINE

1. Quorums
2. Google File System

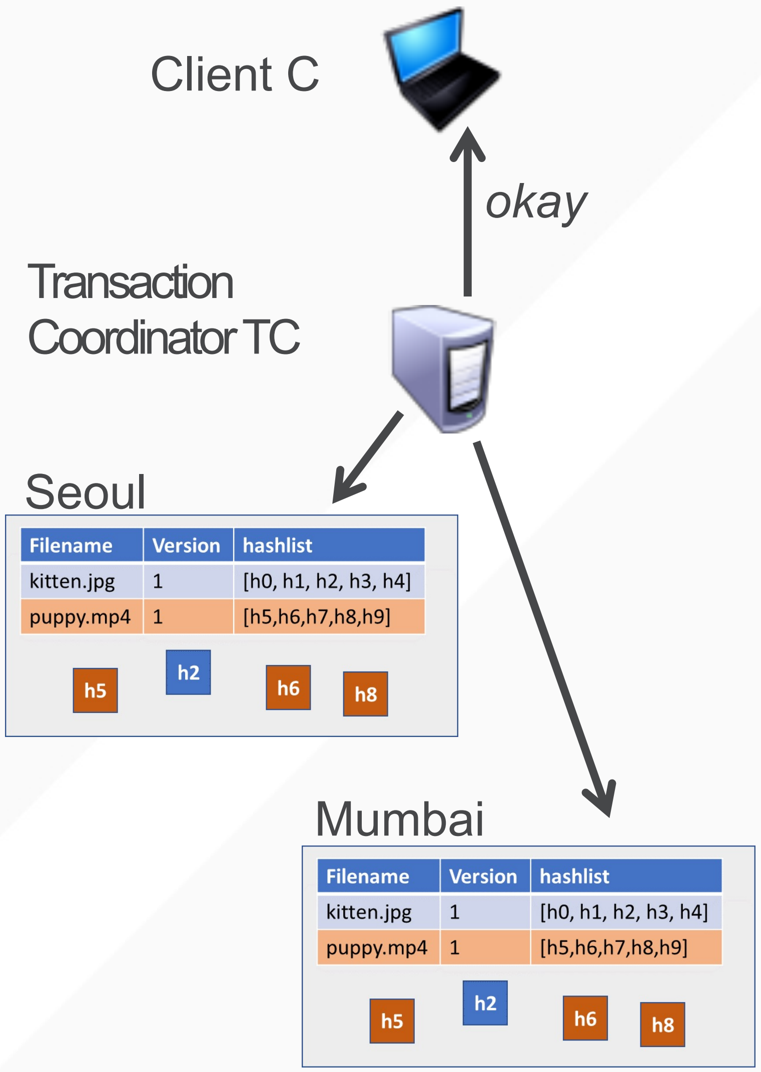
SURFSTORE METADATA SERVER PROBLEM

```
UpdateFile(
  file="kitten.jpg",
  ver=2,
  hashlist = {h3,h4}
);
```



All data is lost!

IDEA 1: ADAPT TWO-PHASE COMMIT TO SAVE DATA



Seoul:

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

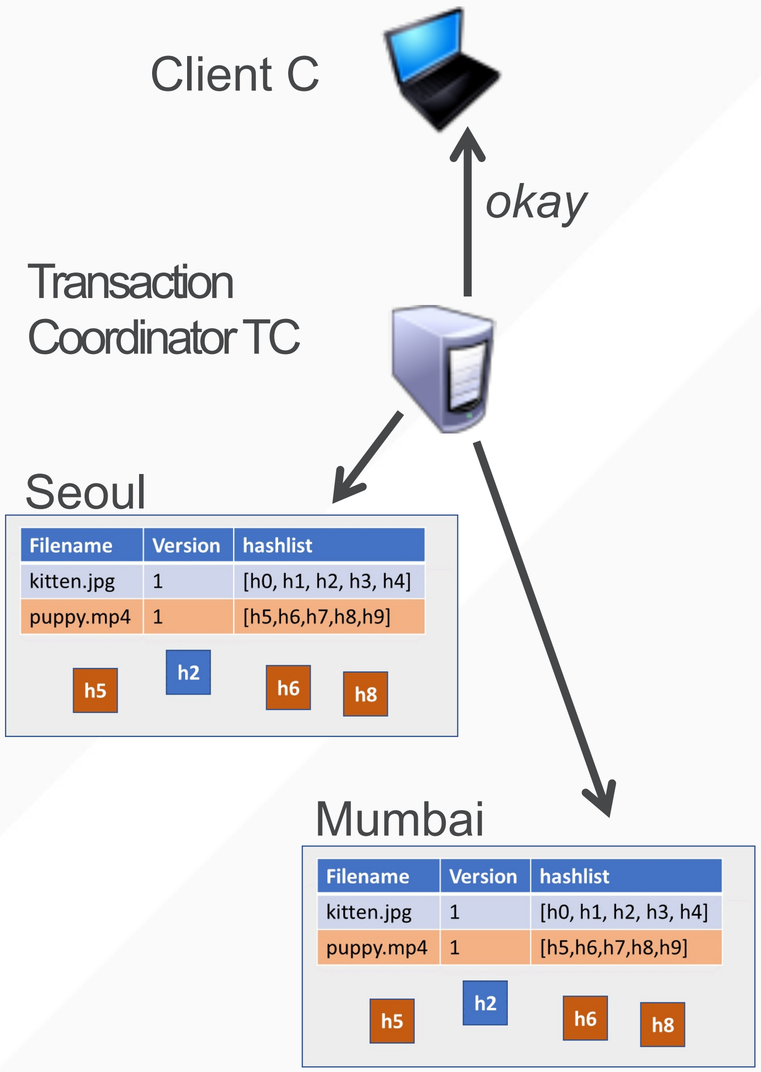
Mumbai:

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

1. **C → TC:** *"UpdateFile()"*
2. **TC → Seoul (S), Mumbai (M):** *"prepare!"*
3. **S, M → P:** *"yes"* or *"wrong_version"*
4. **TC → S, M:** *"commit!"* or *"abort!"*
   - **TC** sends ***commit*** if **both** say *yes*
   - **TC** sends ***abort*** if **either** say *no*
5. **TC → C:** *"okay" or "failed"*

- **S, M** commit on receipt of commit message

IDEA 2: ASSUME TC DOESN'T FAIL (FOR NOW)



Seoul

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

Mumbai

Filename	Version	hashlist
kitten.jpg	1	[h0, h1, h2, h3, h4]
puppy.mp4	1	[h5,h6,h7,h8,h9]

1. **C → TC:** *"UpdateFile()"*
2. **TC → Seoul (S), Mumbai (M):** *"prepare!"*
3. **S, M → P:** *"yes" [why always yes?]*
4. **TC → S, M:** *"commit!"*
   - **TC** sends ***commit***
5. **TC → C:** *"okay"*

- **S, M** commit on receipt of commit message
- ***Why do we still need the commit?***

NETWORK PARTITIONS

- Some failure (either network or host) keeps replicas from communicating with one another
- Two-phase commit (even if we assume all replicas agree) only works if all nodes can be contacted
- How to proceed with read/write transactions in case where not all replicas can be contacted?

QUORUM-BASED PROTOCOLS

- Idea: Tell client that a file's version is updated after a majority of SurfStoreServers get the update
- Form a "read quorum" of size N_R
  - Contact N_R servers and read all their versions
  - Select highest version as the "correct" version
- Form a "write quorum" of size N_W
  - Contact N_W servers
  - Increment the highest version from that set
  - Write out that new version to the servers in the write quorum

CONSTANTS AND CONSTRAINTS

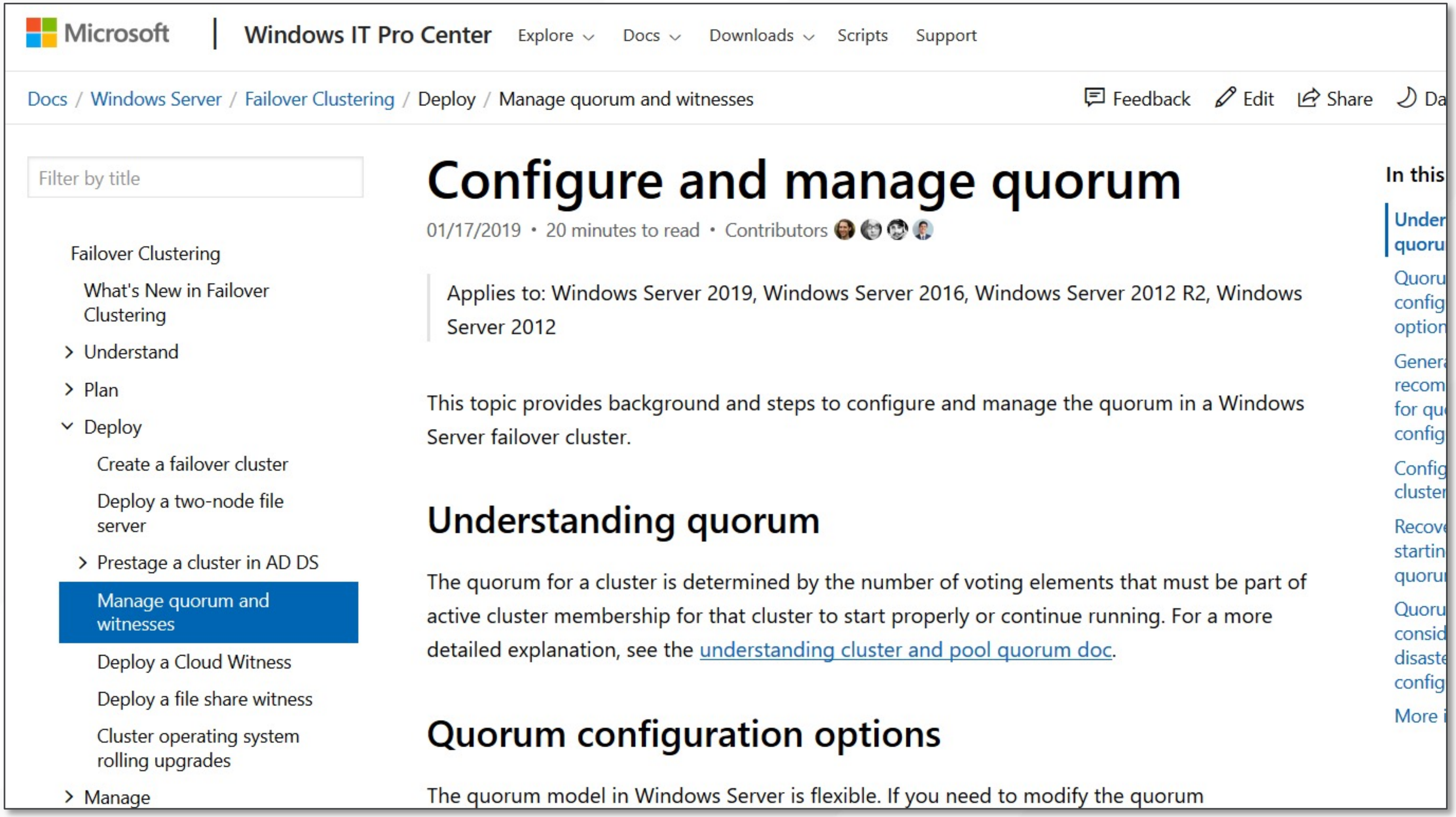
- N: Total #Replicas
- N_R: #Replicas in Read Quorum
- N_W: #Replicas in Write Quorum
- Constraints:
  1. $N_R + N_W > N$
  2. $N_W > N/2$

QUORUM CONSENSUS

- Write operations can be propagated in background to replicas not in quorum
  - Assumes eventual repair of any network partition

- Operations are slowed by the necessity of first gathering a quorum
  - Though previously, all writes had to go to all replicas
    - With quorum system, must only contact subset of replicas

QUORUMS IN MICROSOFT ACTIVE DIRECTORY


Microsoft | Windows IT Pro Center Explore Docs Downloads Scripts Support
Docs / Windows Server / Failover Clustering / Deploy / Manage quorum and witnesses
Feedback Edit Share Da
Filter by title
Failover Clustering
What's New in Failover Clustering
Understand
Plan
Deploy
Create a failover cluster
Deploy a two-node file server
Prestage a cluster in AD DS
Manage quorum and witnesses
Deploy a Cloud Witness
Deploy a file share witness
Cluster operating system rolling upgrades
Manage
In this
Under quoru
Quoru config option
Genera recom for qu config
Config cluster
Recove startin quoru
Quoru consid disaste config
More i

Configure and manage quorum

01/17/2019 • 20 minutes to read • Contributors

Applies to: Windows Server 2019, Windows Server 2016, Windows Server 2012 R2, Windows Server 2012

This topic provides background and steps to configure and manage the quorum in a Windows Server failover cluster.

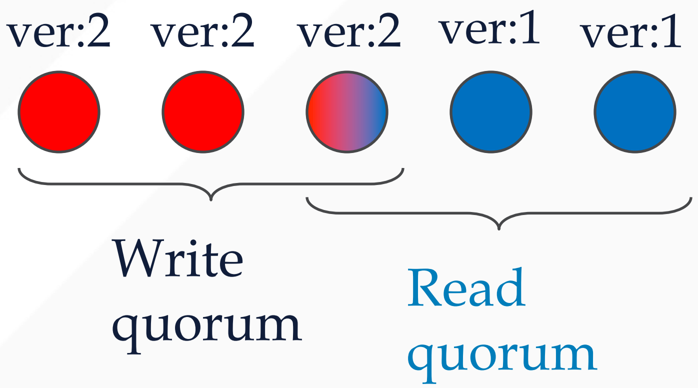
Understanding quorum

The quorum for a cluster is determined by the number of voting elements that must be part of active cluster membership for that cluster to start properly or continue running. For a more detailed explanation, see the understanding cluster and pool quorum doc.

Quorum configuration options

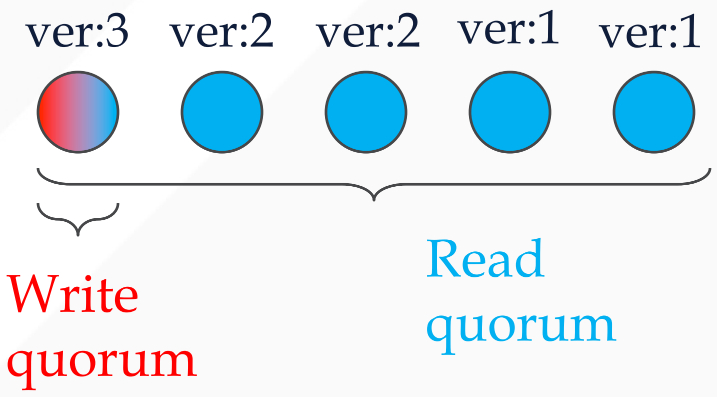
The quorum model in Windows Server is flexible. If you need to modify the quorum

QUORUM EXAMPLE



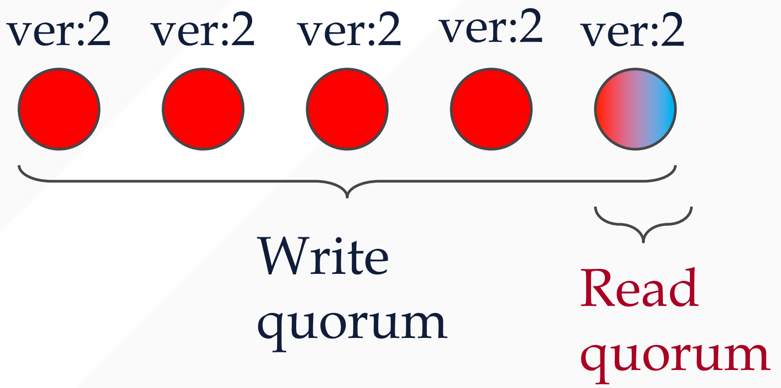
- 5 replicas, read quorum: 3, write quorum: 3
  - R+W>5 votes ensures overlap between any read/write quorum
- How does this perform for reads?
- How does this perform for writes?

QUORUM EXAMPLE



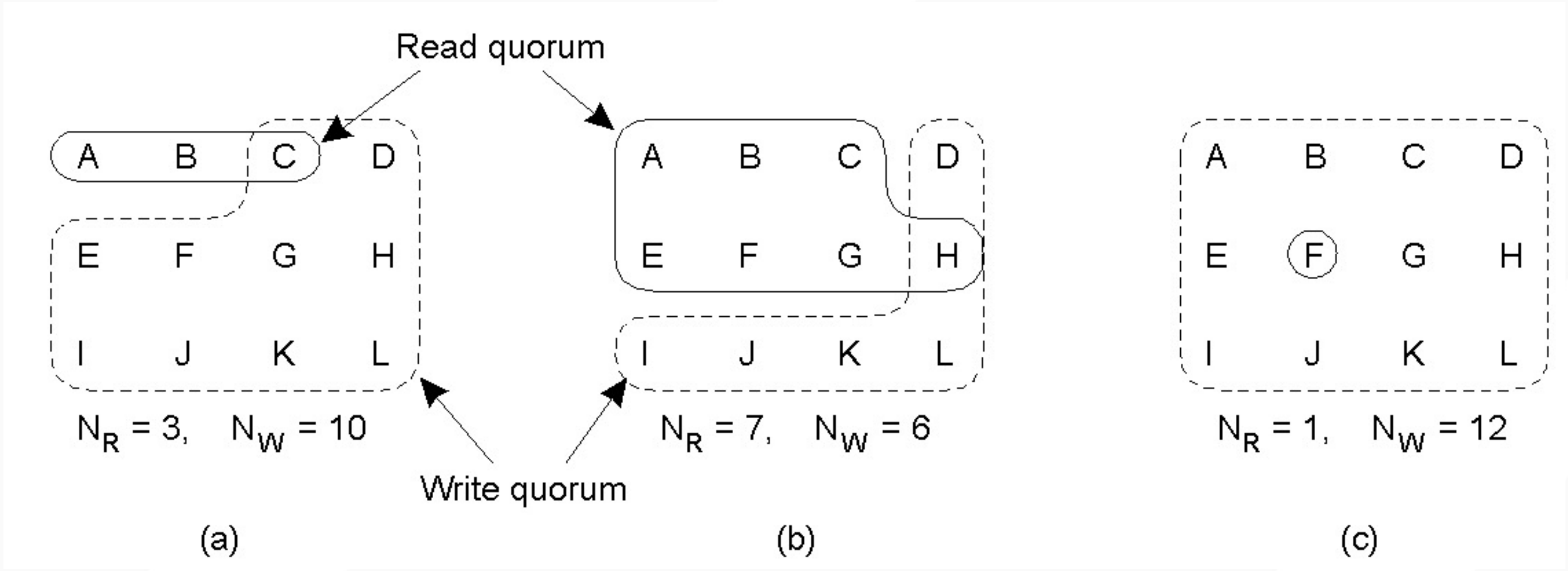
- 5 replicas, read quorum: 5, write quorum: 1
  - R+W>5 votes ensures overlap between any read/write quorum
- How does this perform for reads?
- How does this perform for writes?

QUORUM EXAMPLE



- 5 replicas, read quorum: 1, write quorum: 5
    - R+W>5 votes ensures overlap between any read/write quorum
    - Also called ROWA (read one, write all)
- How does this perform for reads?
- How does this perform for writes?

EXAMPLES



- (a) Correct choice
- (b) Possible write-write conflict (why?)
- (c) ROWA

OUTLINE

1. ~~Quorums~~
2. Google File System

LEASES

- Client obtains ***lease*** on file for read or write
  - "A lease is a ticket permitting an activity; the lease is valid until some expiration time."
- Read lease allows client to cache clean data
  - *Guarantee:* no other client is modifying file
- Write lease allows safe delayed writes
  - Client can locally modify than batch writes to server
  - *Guarantee:* no other client has file cached

USING LEASES

- Client requests a lease
  - May be implicit, distinct from file locking
  - Issued lease has file version number for cache coherence
- Server determines if lease can be granted
  - *Read leases* may be granted concurrently
  - *Write leases* are granted exclusively
- If conflict exists, server may send *eviction* notices
  - Evicted write lease must write back
  - Evicted read leases must flush/disable caching
  - Client acknowledges when completed

BOUNDED LEASE TERM SIMPLIFIES RECOVERY

- Before lease expires, client must *renew* lease
- Client fails while holding a lease?
  - Server waits until the lease expires, then unilaterally reclaims
  - If client fails during eviction, server waits then reclaims
- Server fails while leases outstanding? On recovery,
  - Wait *lease period + clock skew* before issuing new leases
  - Absorb renewal requests and/or writes for evicted leases

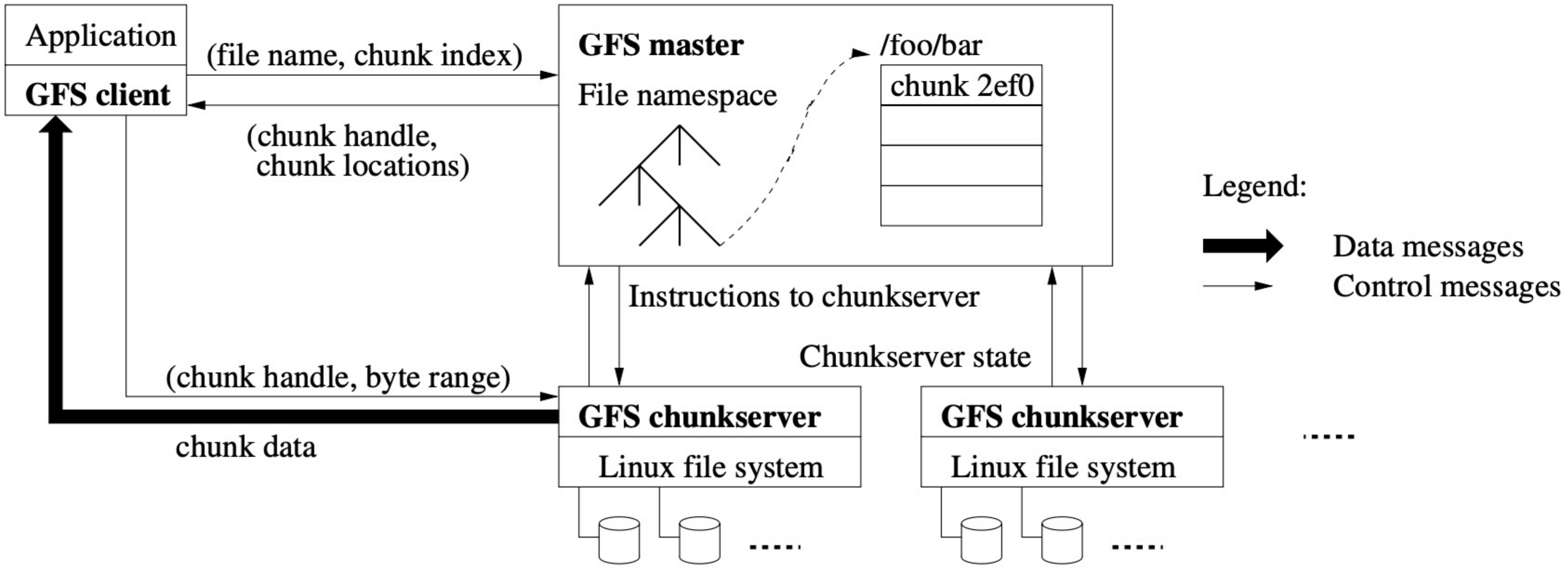
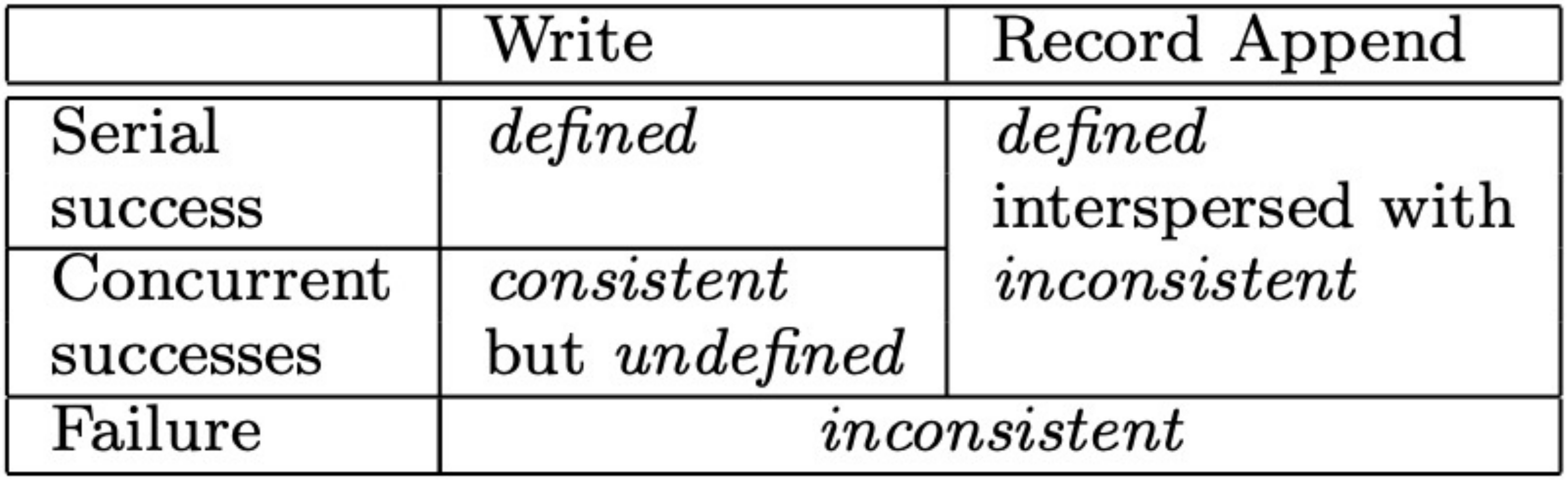


Figure 1: GFS Architecture



	Write	Record Append
Serial success	*defined*	*defined* interspersed with *inconsistent*
Concurrent successes	*consistent* but *undefined*	
Failure	*inconsistent*	

Table 1: File Region State After Mutation

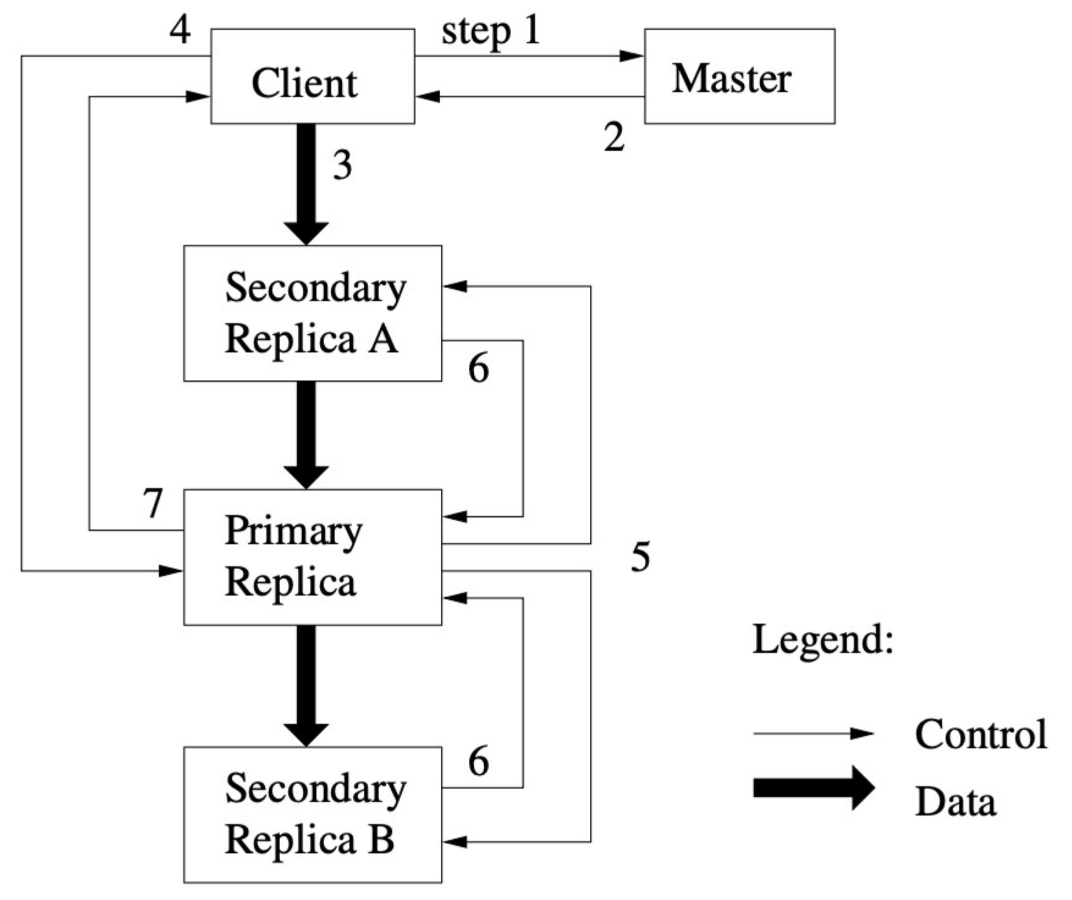
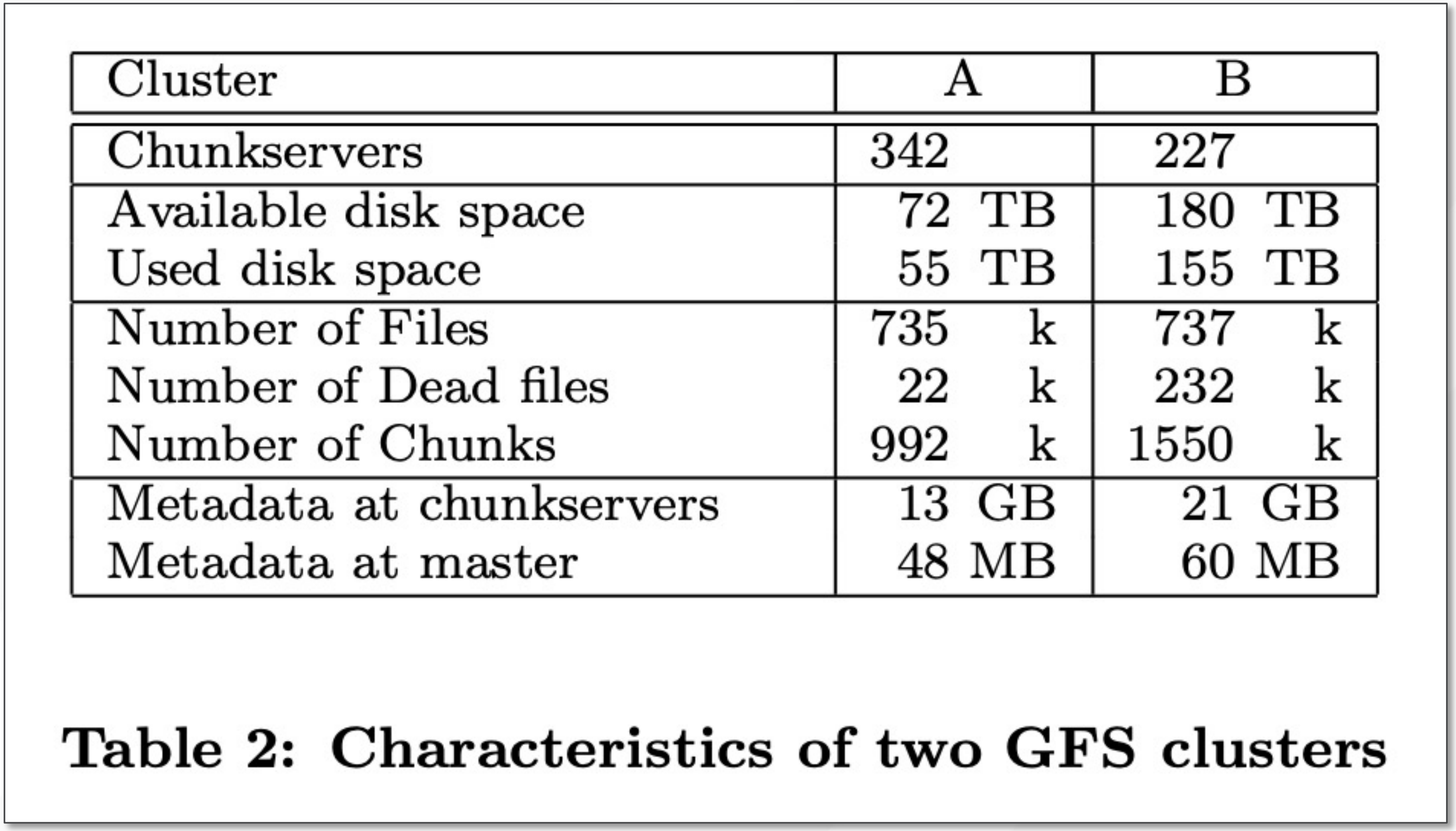


Figure 2: Write Control and Data Flow



Cluster	A	B
Chunkservers	342	227
Available disk space	72 TB	180 TB
Used disk space	55 TB	155 TB
Number of Files	735 k	737 k
Number of Dead files	22 k	232 k
Number of Chunks	992 k	1550 k
Metadata at chunkservers	13 GB	21 GB
Metadata at master	48 MB	60 MB

Table 2: Characteristics of two GFS clusters

UC San Diego