EECS 482
Introduction to Operating Systems

Winter 2018

Harsha V. Madhyastha

Distributed file systems

- Remote storage of data that appears local
- Examples:
  - AFS
  - Dropbox
  - Google Docs

- Benefits?
  - Share files across users
  - Uniform view of file system across machines

Caching for performance

- Bottleneck if many clients interacting with server?
  - Server
  - Network

- Benefits of client-side caching:
  - Improves server scalability
  - Better latency and throughput
  - Reduces network traffic
  - Can improve availability

Client-side caching

- Two approaches:
  - Migrate: Transfer sole copy from server to client
    - » Simpler to implement
    - » Concurrent reads leads to ping-ponging
  - Replicate: Create additional copy at client
    - » Clients can read from local copy
    - » Must worry about inconsistent replicas

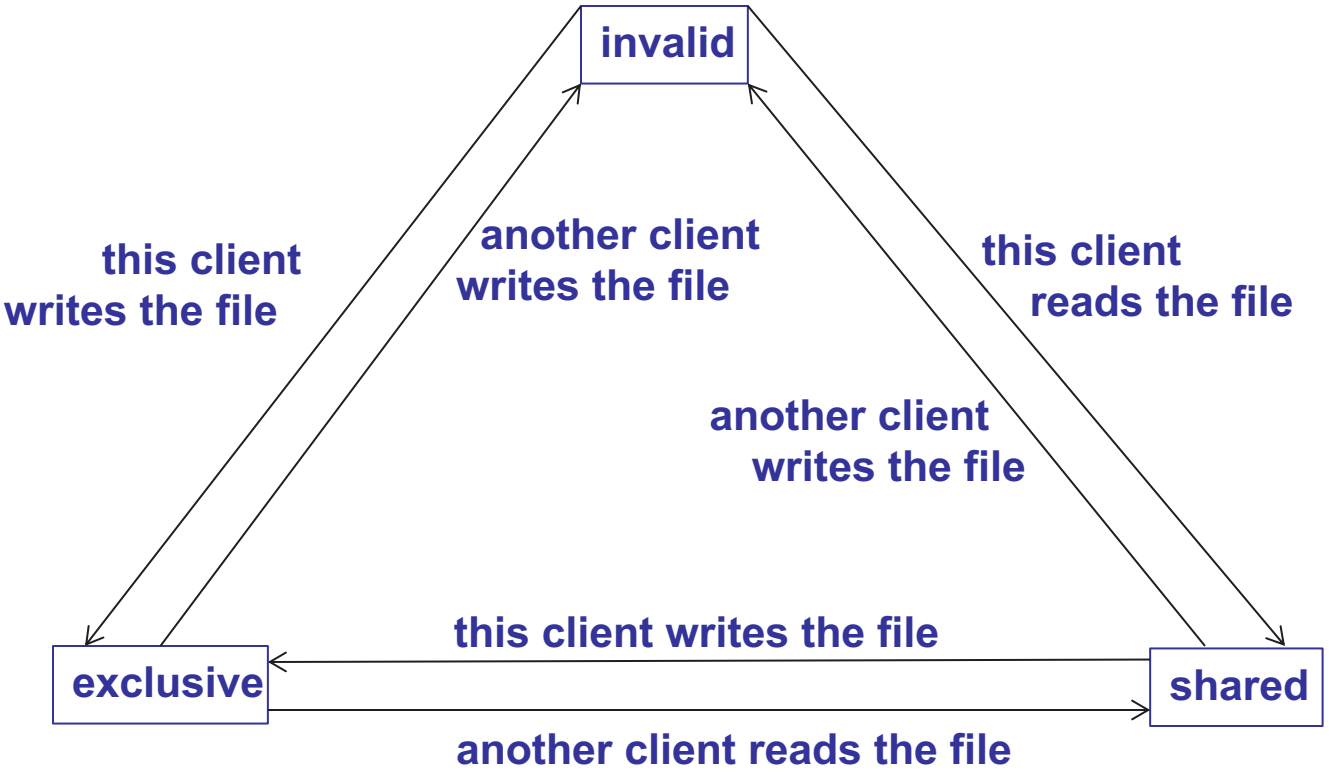
- How do the two approaches compare?

Handling writes with caching

- If clients use write-back caching,
  - Other clients may read before data written back

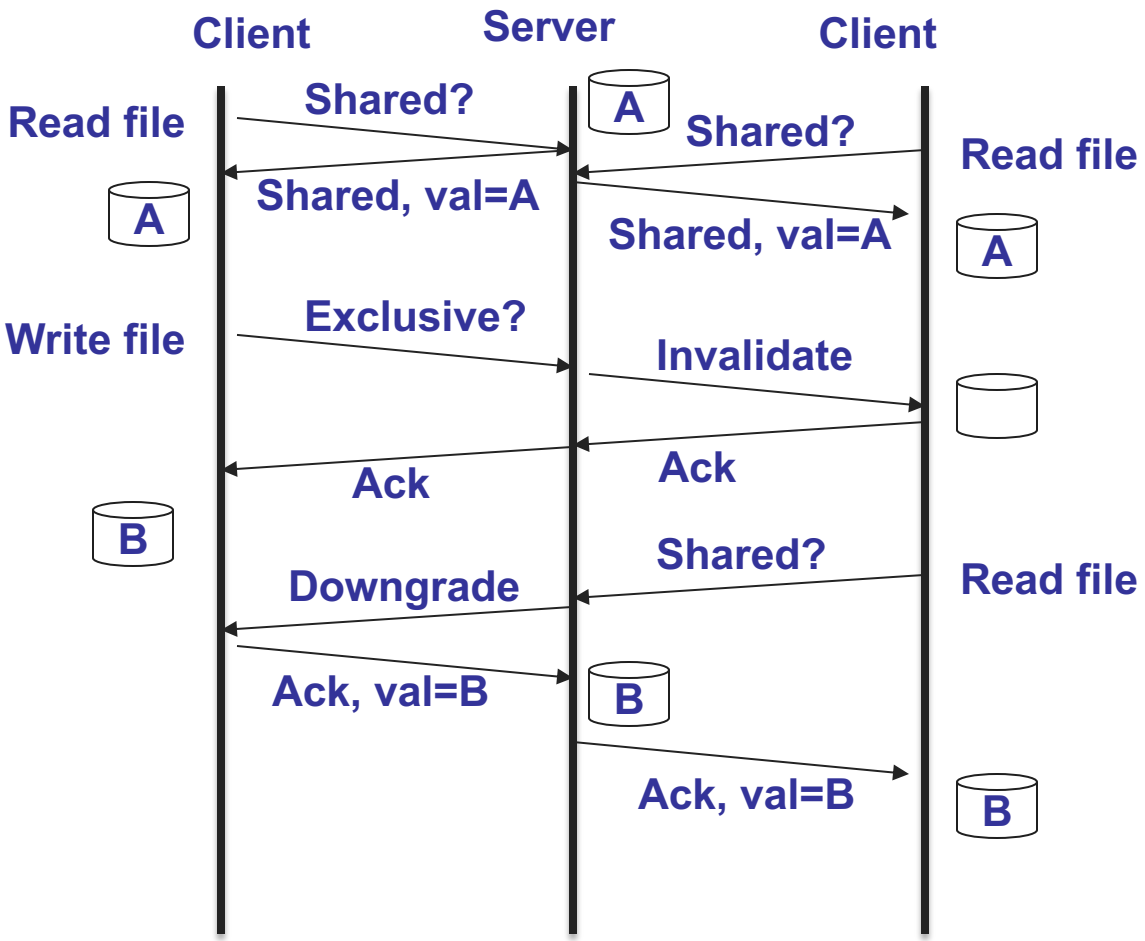
- How to preserve consistency?
  - Write through cache?
  - Update all copies or invalidate other copies
  - Pros and cons?

State machine for cached copy



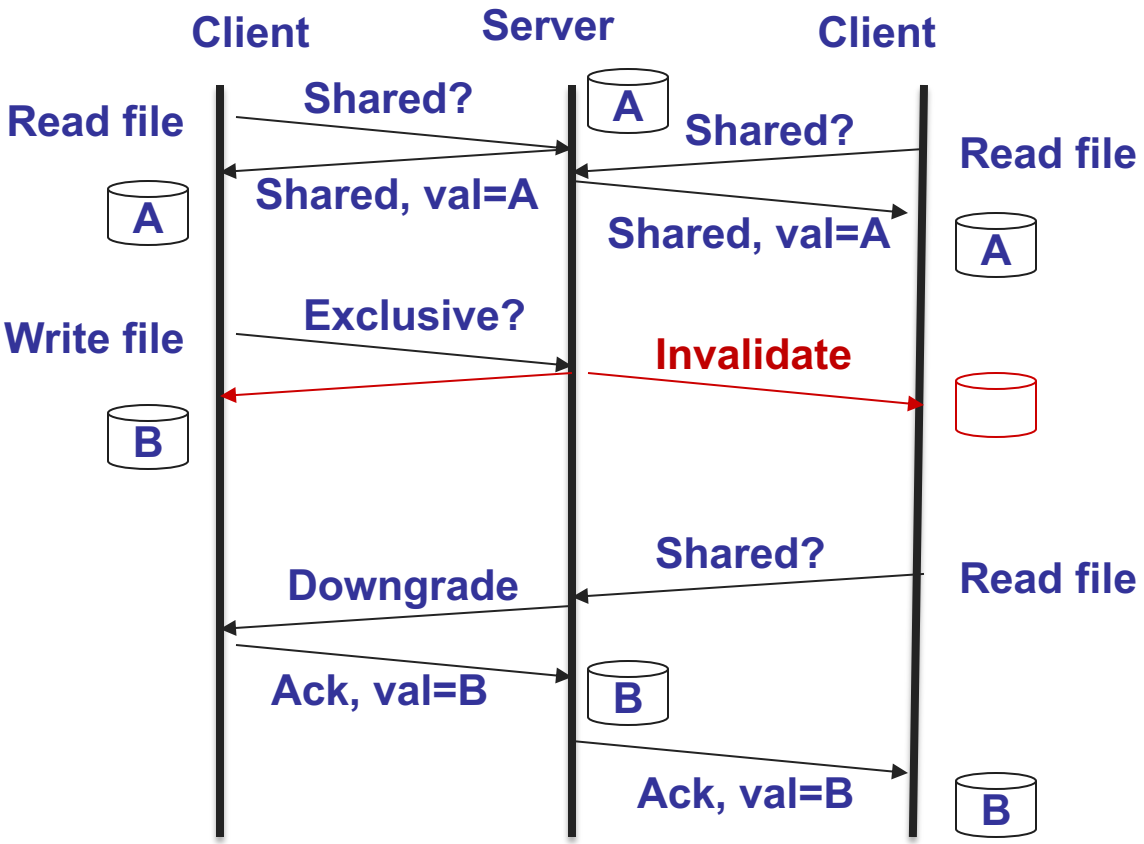
Similar to anything else we've discussed previously?

Invalidation protocol



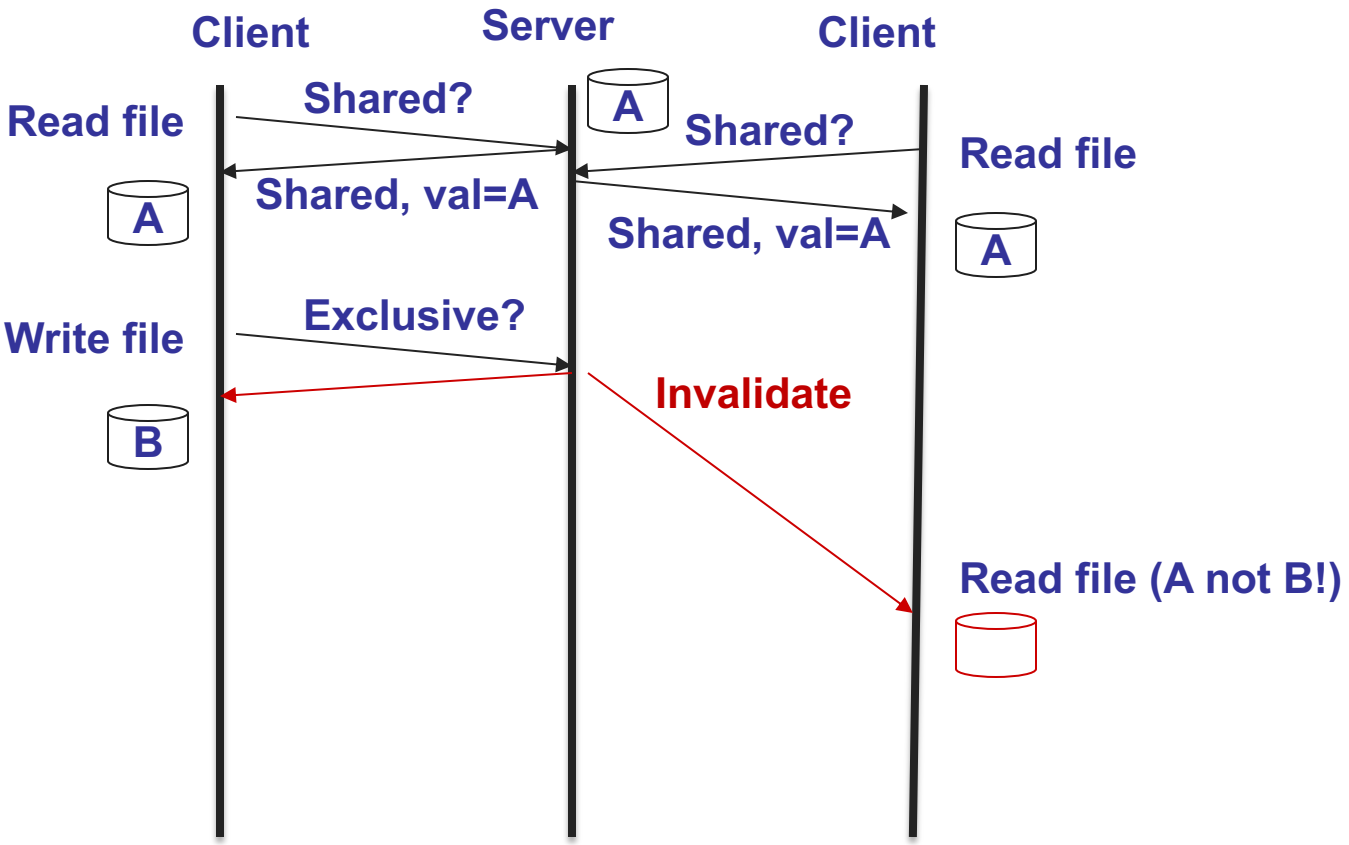
Order of operations

- Necessary to wait for invalidations?



Order of operations

- Necessary to wait for invalidations?

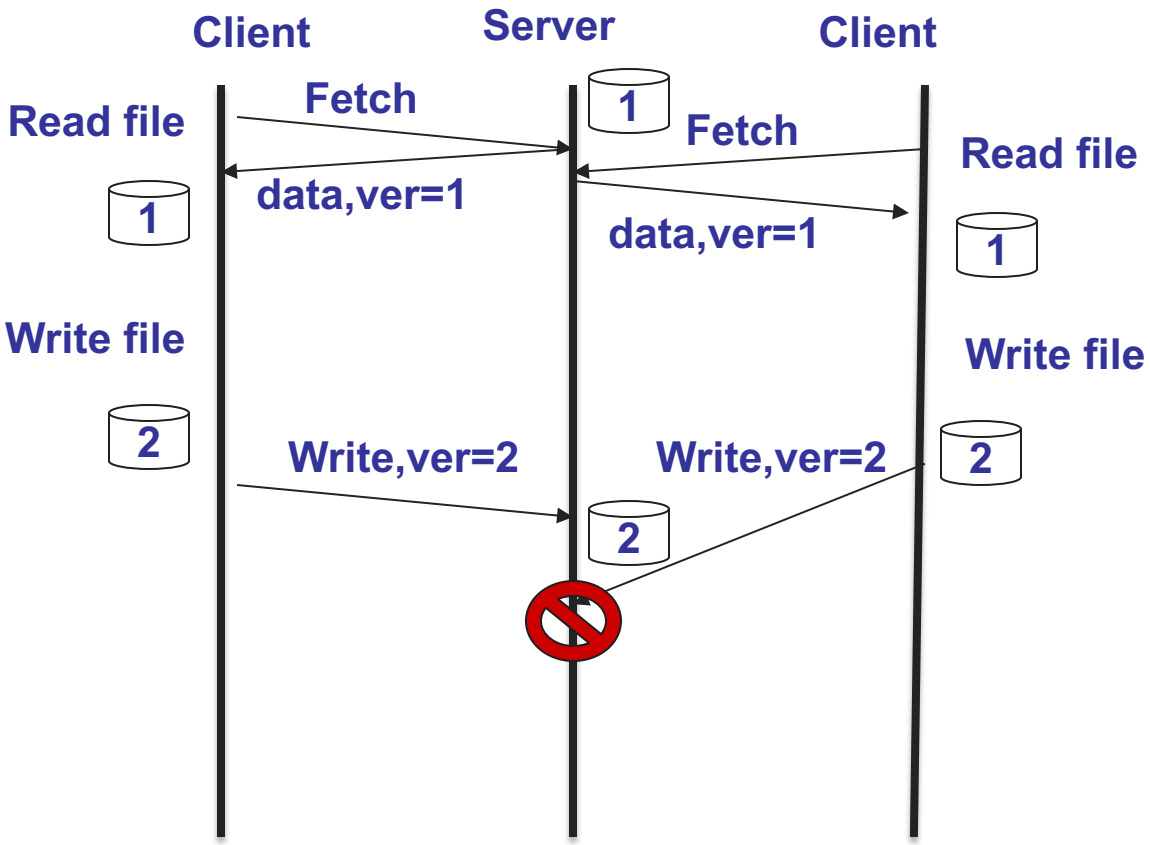


Optimistic concurrency control

- Alternate approach to prevent inconsistency
  - Allow clients to modify replicas freely
  - Server detects and resolves conflicting updates

- How to detect conflicts?
  - Assign a version to each object (file, etc.)
  - Increment the version on update
  - Conflict if server version >= client version

Conflicting operations



Project 4 Testing

- State space coverage
  - Test every request type with every possible state
- For example: FS_CREATE
  - File vs. directory
  - In root directory vs. elsewhere
  - Adding direntry in first data block vs. later
  - Free direntry at the beginning vs. later
- Test close to resource limits
  - Disk size, max path name, max file name, …

Project 4 Testing

- Verifying concurrency
  - Test with a pair of requests
  - Consider every combination of request types
  - Vary commonality in pathnames
  - Block around “slow” operations
- Macro test cases
  - Crash or deadlock with lots of concurrent requests
  - Check for memory leaks

Announcements

- Final exam: 7-9pm on April 23rd
  - Will post room assignments on Piazza
- Review of sample final exams on April 21st
  - 12:30-3:30pm in CHRYS 220
  - Before then, take exams yourself **in exam setting**
- IAs will go over sample questions in discussion section this Friday

Announcements

- Submit course evaluations

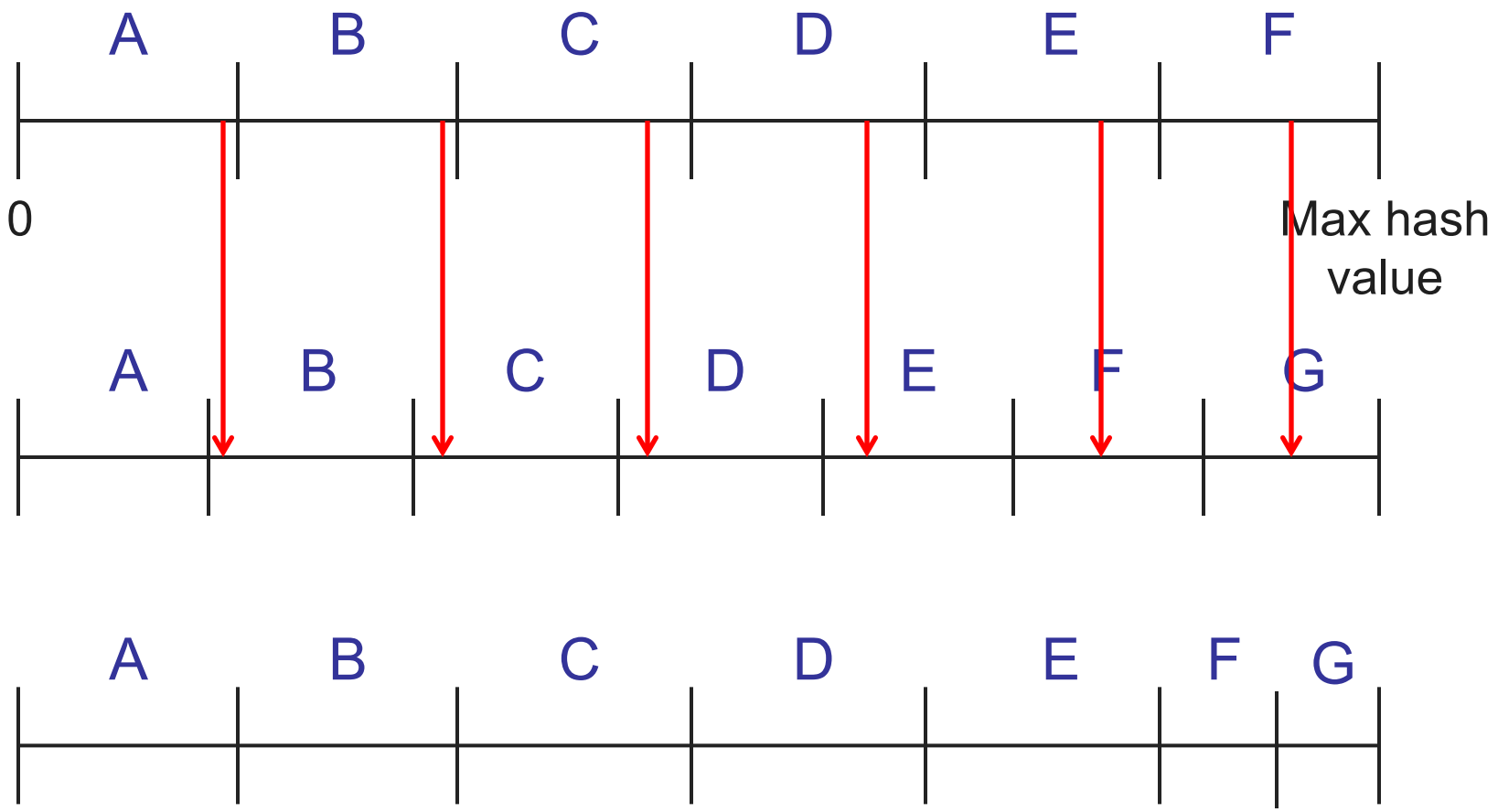
- No lecture on Wednesday
- Email me topics to review next Monday

Load balancing across servers

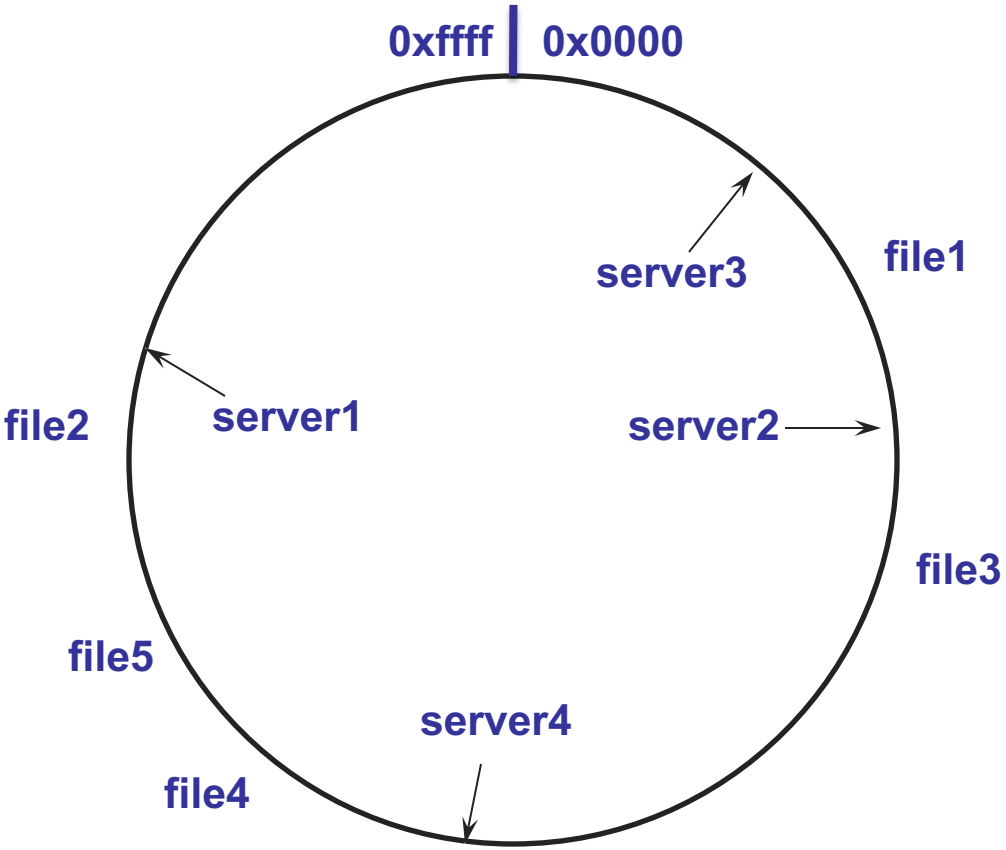
- Two options:
  - Partition clients across servers
  - Partition files across servers

- How to find a file?
  - Ask a directory server
  - Hash-based mapping:
    - » If N servers, store file *foo* on server *hash(foo) % N*
    - » What if we need to add or remove a server?
    - » File is now mapped to server *hash(foo) % (N+1)*

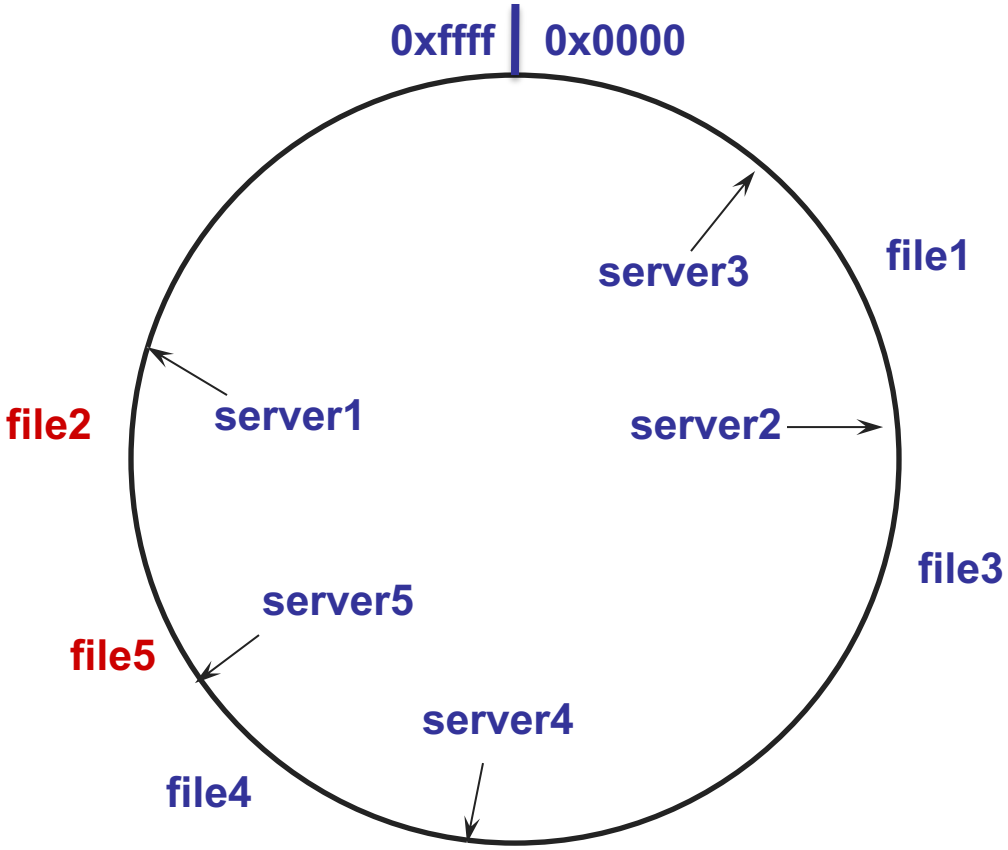
Load balancing across servers



Consistent hashing



Adding a server

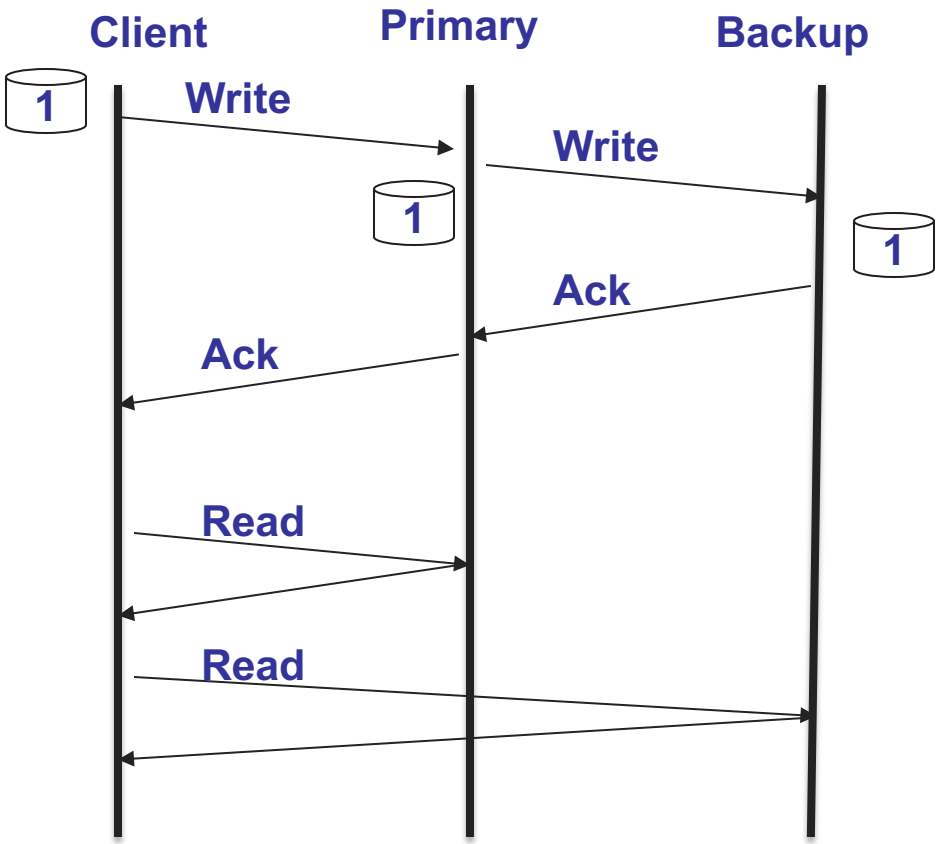


Replication across servers

- More servers → Likelihood of failure increases

- Replicate files to tolerate failures
  - Example: Primary + backup
    - » Write data by writing to both primary AND backup
    - » Read data by reading from primary OR backup

Primary-Backup



Handling failures

- When primary fails
  - Backup becomes new primary
  - Backup handles all reads/writes
  - Primary recovers, syncs state, can become backup

- When backup fails
  - Primary handles all reads/writes
  - Backup recovers, syncs state

Fault tolerance

- What if backup fails before primary recovers?
  - Data is unavailable, lost if failures permanent
- How can we tolerate 2 failures?
  - Use 2 backups
  - Need f+1 servers to tolerate f failures
- What are we assuming about failures?

Fault models

- Fail stop (primary-backup)
  - Machine stops executing immediately
  - Can detect the failure
  - Examples: power failure, OS crash

- Byzantine
  - Anything goes!
  - Server may send erroneous messages
  - Server could be malicious/hacked

Byzantine generals

- Say there's one commander C and two lieutenants L1 and L2
  - Goal: Decide whether to attack enemy or retreat
  - Attack succeeds if at least 2 attack
  - 1 of the 3 is a traitor
- Solution 1: L1 and L2 follow C's command to either attack or retreat
- Solution 2: C sends command to L1 and L2, who then exchange notes and follow majority

Byzantine generals

- C sends command to L1 and L2, who then exchange notes and follow majority
- Case 1: L2 is traitor
  - C sends attack to both L1 and L2
  - L1 receives {attack, retreat}
- Case 2: C is traitor
  - C sends attack to L1 and retreat to L2
  - L1 receives {attack, retreat}

Byzantine generals

- Need at least 4 generals to cope with 1 traitor
  - 3f+1 generals to cope with f traitors

- Solution: C sends command to L1, L2, and L3, who then exchange notes and follow majority
- Three cases:
  - C sends 3 attacks
  - C sends 2 attacks, 1 retreat
  - C sends 1 attack, 2 retreats