Part 11

殷亚凤

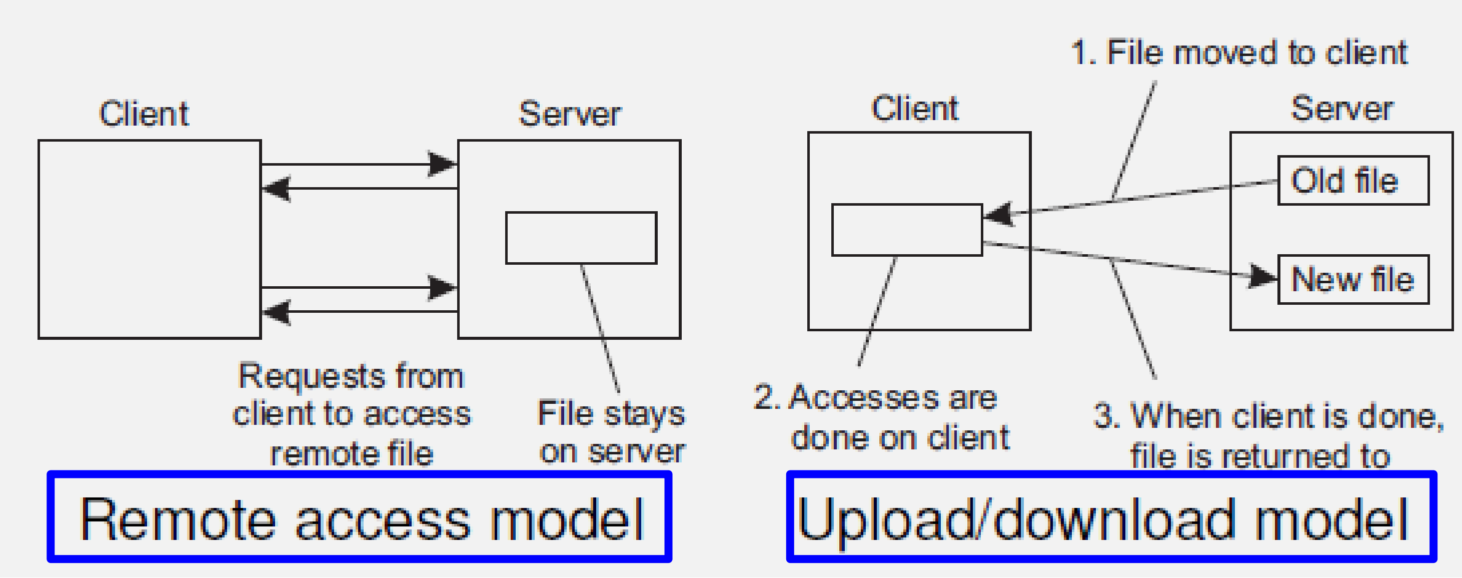
Email: yafeng@nju.edu.cn

Homepage: http://cs.nju.edu.cn/yafeng/

Room 301, Building of Computer Science and Technology

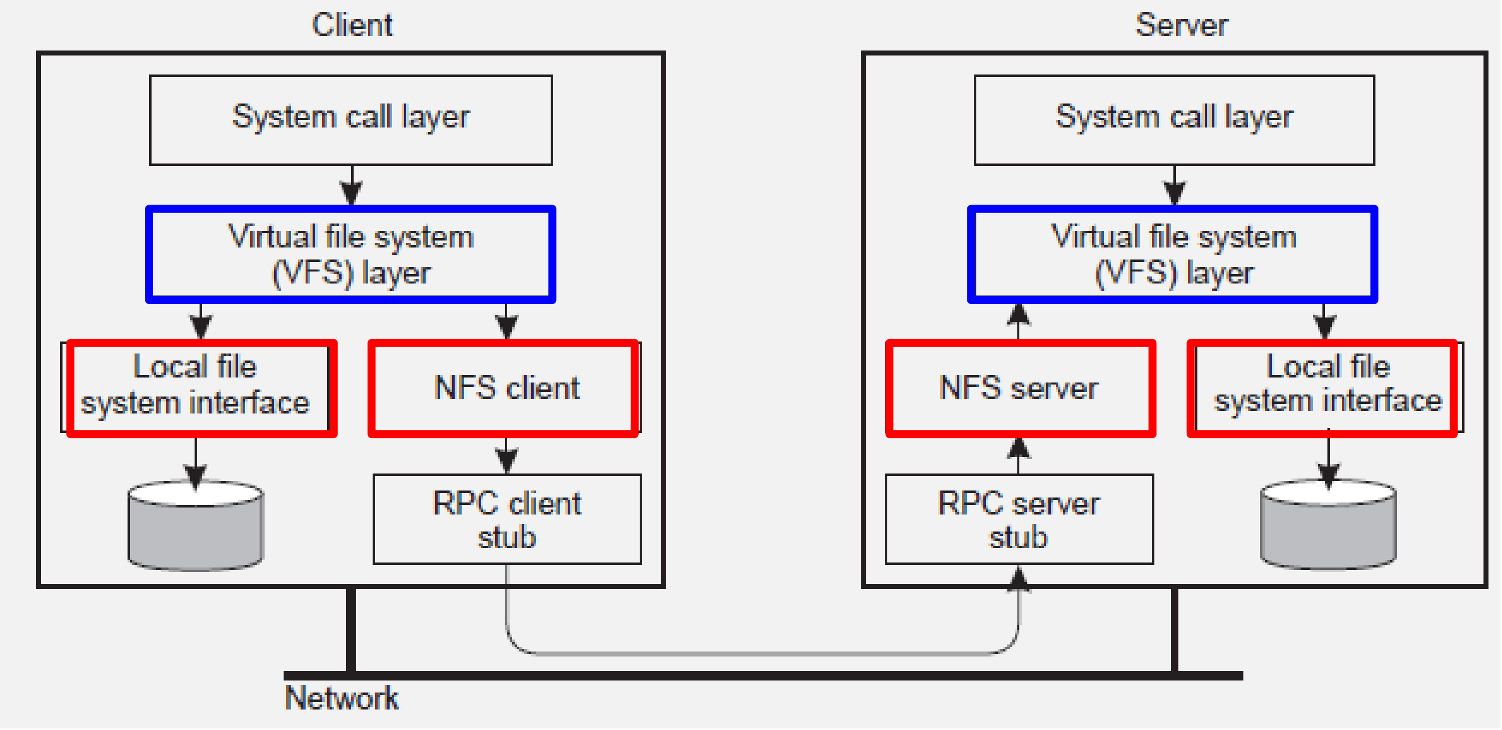
Distributed File Systems

- Try to make a file system transparently available to remote clients.



Example: NFS Architecture

- NFS is implemented using the Virtual File System abstraction, which is now used for lots of different operating systems.

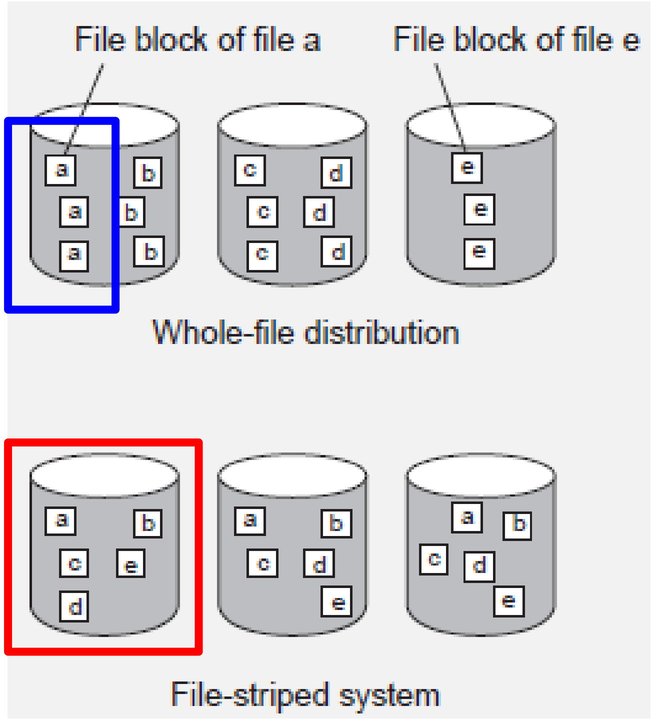


NFS File Operations

Oper.	v3	v4	Description
Create	Yes	No	Create a regular file
Create	No	Yes	Create a nonregular file
Link	Yes	Yes	Create a hard link to a file
Symlink	Yes	No	Create a symbolic link to a file
Mkdir	Yes	No	Create a subdirectory
Mknod	Yes	No	Create a special file
Rename	Yes	Yes	Change the name of a file
Remove	Yes	Yes	Remove a file from a file system
Rmdir	Yes	No	Remove an empty subdirectory
Open	No	Yes	Open a file
Close	No	Yes	Close a file
Lookup	Yes	Yes	Look up a file by means of a name
Readdir	Yes	Yes	Read the entries in a directory
Readlink	Yes	Yes	Read the path name in a symbolic link
Getattr	Yes	Yes	Get the attribute values for a file
Setattr	Yes	Yes	Set one or more file-attribute values
Read	Yes	Yes	Read the data contained in a file
Write	Yes	Yes	Write data to a file

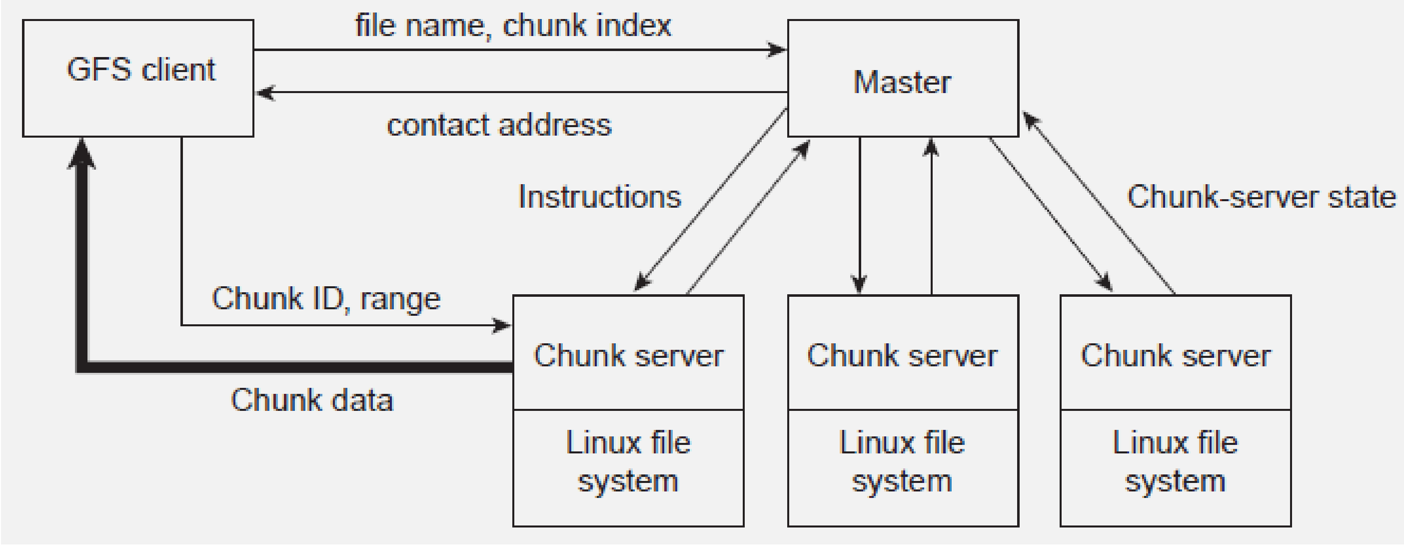
Cluster-Based File Systems

- With very large data collections, following a simple client-server approach is not going to work ⇒ for speeding up file accesses, apply striping techniques by which files can be fetched in parallel.



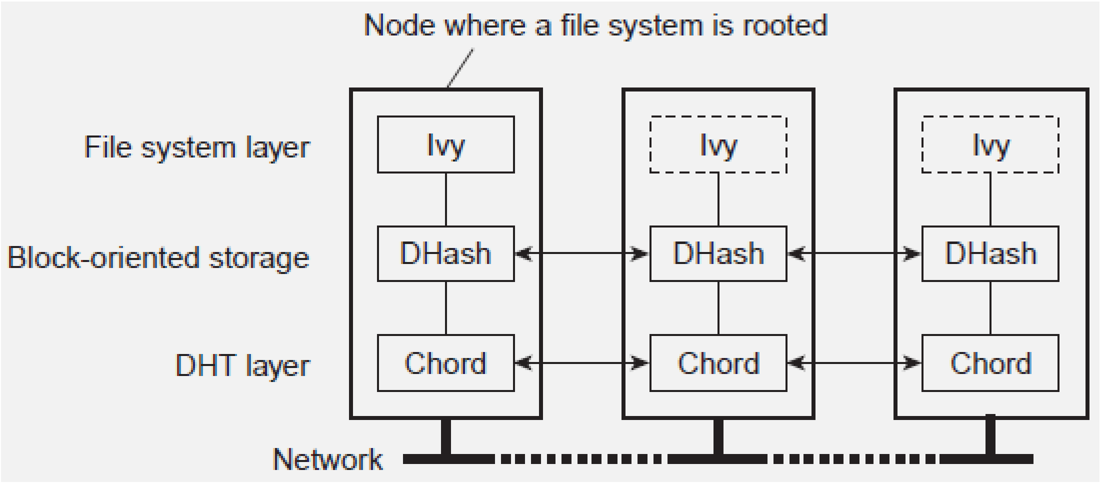
Example: Google File System

- Divide files in large 64 MB chunks, and distribute/replicate chunks across many servers:
  - The master maintains only a (file name, chunk server) table in main memory ⇒ minimal I/O
  - Files are replicated using a primary-backup scheme; the master is kept out of the loop



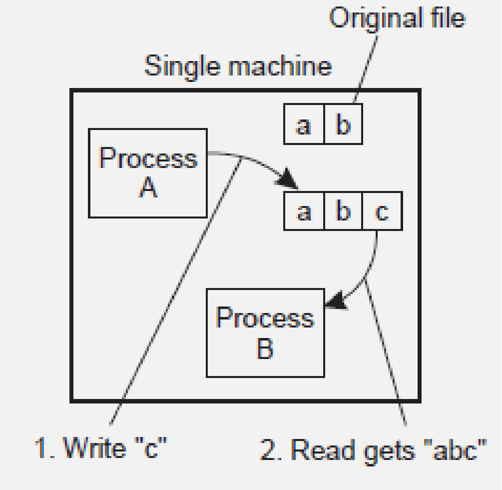
P2P-based File Systems

- Store data blocks in the underlying P2P system:
  - Every data block with content D is stored on a node with hash $h(D)$. Allows for integrity check.
  - Public-key blocks are signed with associated private key and looked up with public key.
  - A local log of file operations to keep track of $\langle blockID, h(D) \rangle$ pairs.

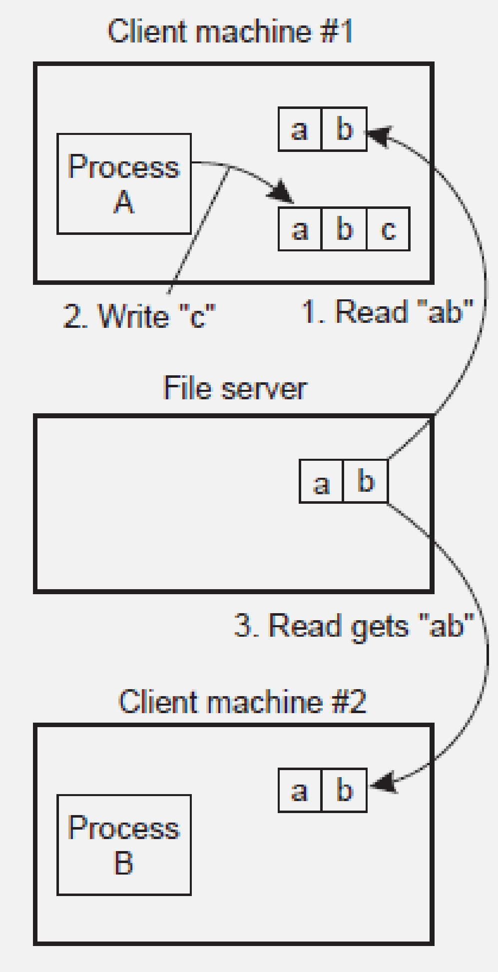


File sharing semantics

- When dealing with distributed file systems, we need to take into account the ordering of concurrent read/write operations and expected semantics(i.e., consistency).



(a)



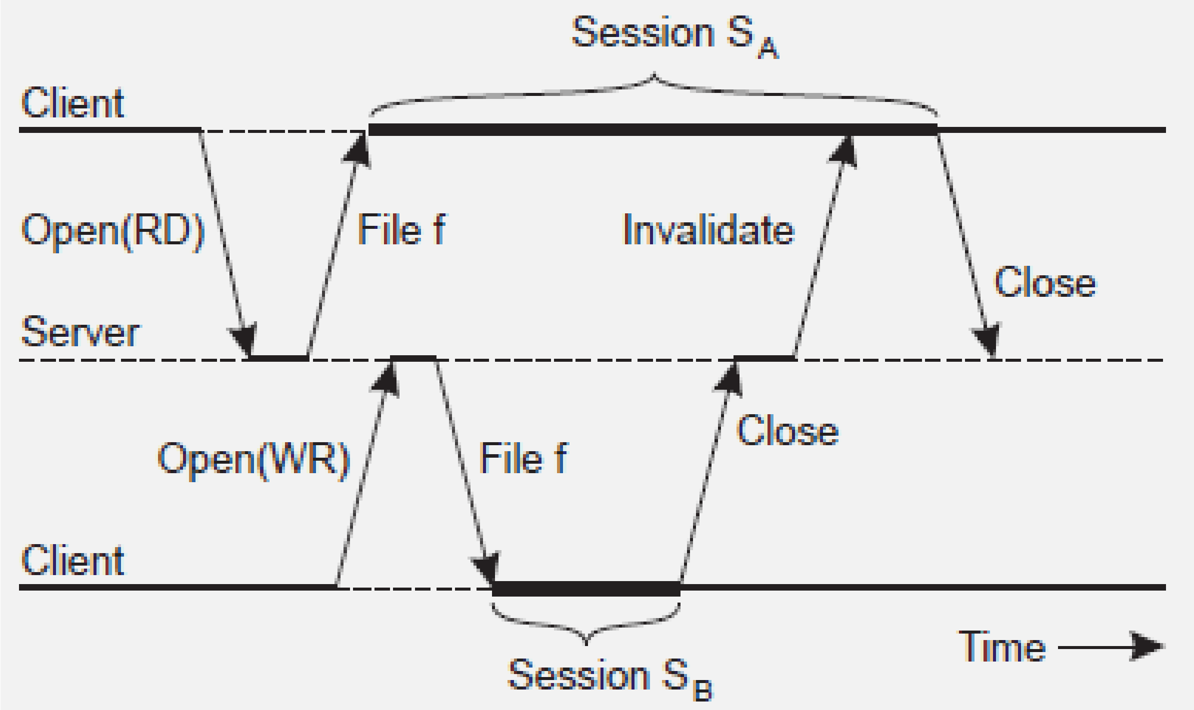
(b)

File sharing semantics

- UNIX semantics: a read operation returns the effect of the last write operation ⇒ can only be implemented for remote access models in which there is only a single copy of the file
- Transaction semantics: the file system supports transactions on a single file ⇒ issue is how to allow concurrent access to a physically distributed file
- Session semantics: the effects of read and write operations are seen only by the client that has opened (a local copy) of the file ⇒ what happens when a file is closed (only one client may actually win)

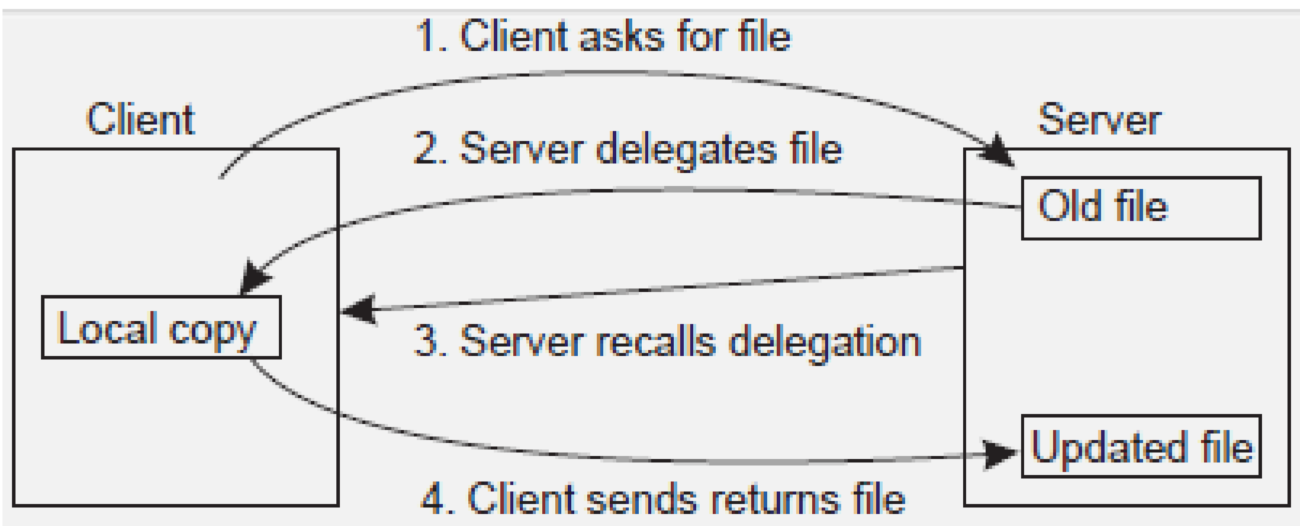
Example: File sharing in Coda

- Coda assumes transactional semantics, but without the full-fledged capabilities of real transactions. Note: Transactional issues reappear in the form of "this ordering could have taken place."



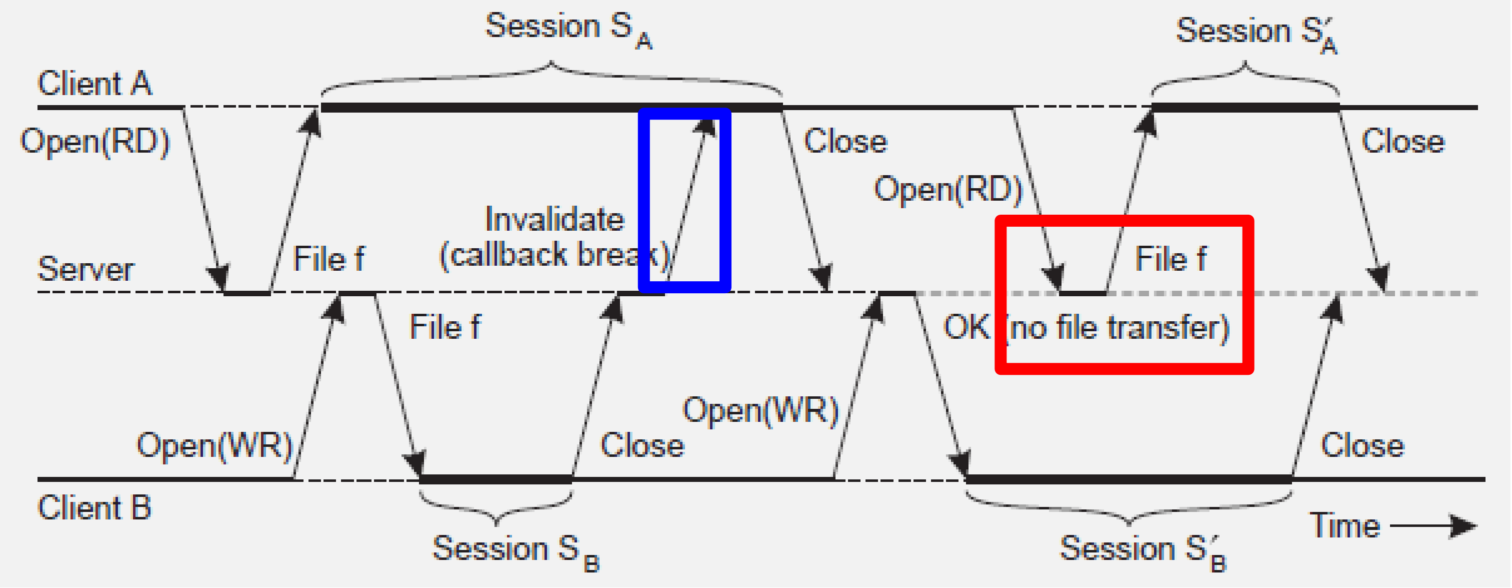
Consistency and replication

- In modern distributed file systems, client-side caching is the preferred technique for attaining performance; server-side replication is done for fault tolerance.
- Clients are allowed to keep (large parts of) a file, and will be notified when control is withdrawn ⇒ servers are now generally stateful



Example: Client-side caching in Coda

- By making use of transactional semantics, it becomes possible to further improve performance.



Example: Server-side replication in Coda

- Ensure that concurrent updates are detected:
  - Each client has an Accessible Volume Storage Group (AVSG): is a subset of the actual VSG.
  - Version vector $CVV_i(f)[j] = k \Rightarrow S_i$ knows that S_j has seen version k of f .
  - Example: A updates $f \Rightarrow S_1 = S_2 = [+1, +1, +0]$; B updates $f \Rightarrow S_3 = [+0, +0, +1]$.