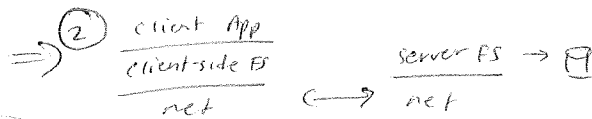


LECTURE

NFS

DBF FS : $\begin{matrix} c \\ c \\ c \end{matrix} \begin{matrix} s \\ s \\ s \end{matrix} \begin{matrix} \square \\ \square \\ \square \end{matrix}$



NFS : goal simple / fast
server crash recovery

key: design of protocol
(way that machines interact)
property (statelessness)

statelessness:

$\left\{ \begin{array}{l} \text{incorporate, into every request,} \\ \text{all info needed to} \\ \text{complete request} \end{array} \right\}$

(server shouldn't track things
like which clients
have open files, etc.)

Protocol:

key is file handle

(volume ID, inode #, gen #)
(why each?)

Examples:

bytes read \Leftarrow read (fh, offset, size)
bytes written \Leftarrow write ()

0, 1 \Leftarrow create (dfh, name)
(mkdir)

fh of name \Leftarrow lookup (dfh, name)

file \Leftarrow getattr (fh)
attr

(stat call, e.g., modified time)

Building FS from this

client calls:

open (" /a/b.txt")

\nearrow
period this is root of NFS volume

lookup(rootfh, "a")

(a's fh)

lookup (a's fh, "b.txt");

(b.txt's fh)

access (b.txt fh, read.)

client fs:

map fh \Rightarrow

this $\begin{matrix} \text{b.txt's} \\ \text{fh} \end{matrix}$

read (size);

\downarrow
read (b.fh, 0, size);

How to handle failure?

key: idempotency

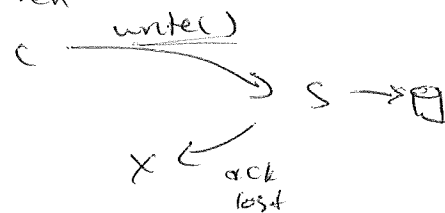
operation is idempotent if ...

use timeout/retry:

thus, client has unified response
to lack of ack \Rightarrow

RETRY

even when



Problem: Performance

server overloaded, client latency \uparrow

solution: client cache

\Rightarrow new problem: cache consistency
for reads:

state (read block X from file
keep in memory of client
for writes:

visibility (write new block Y to file
buffer in memory of client

to address visibility \Rightarrow

flush on close ()

(client cache
dirty writes)

to address state cache \Rightarrow

getattr: check if block
has been changed before
reuse

but, still too many!

\Rightarrow getattr cache w/
3 second timeout

server side caching:

useful? buffering?
correct?