Flash-based SSDs : not today

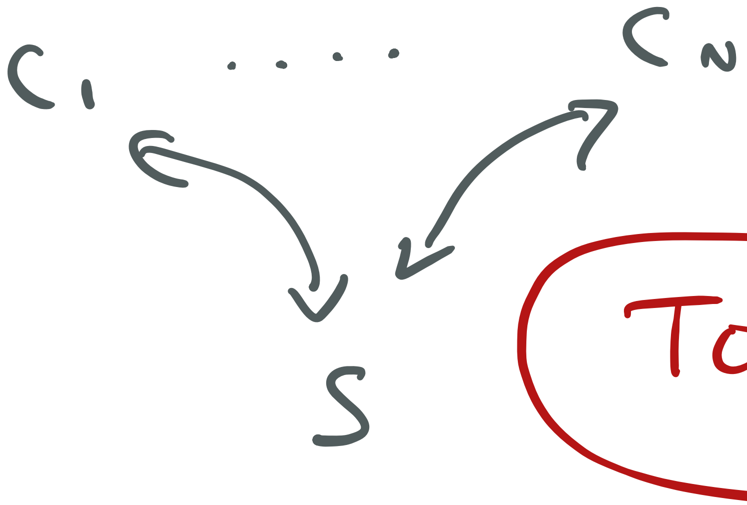
Distributed File Systems: today!

Sun Network File System (NFS)
=) focus, and related to p 4

Crash recovery is key
=) design of protocol is central
to simple crash recovery
(server)

Distributed Systems:

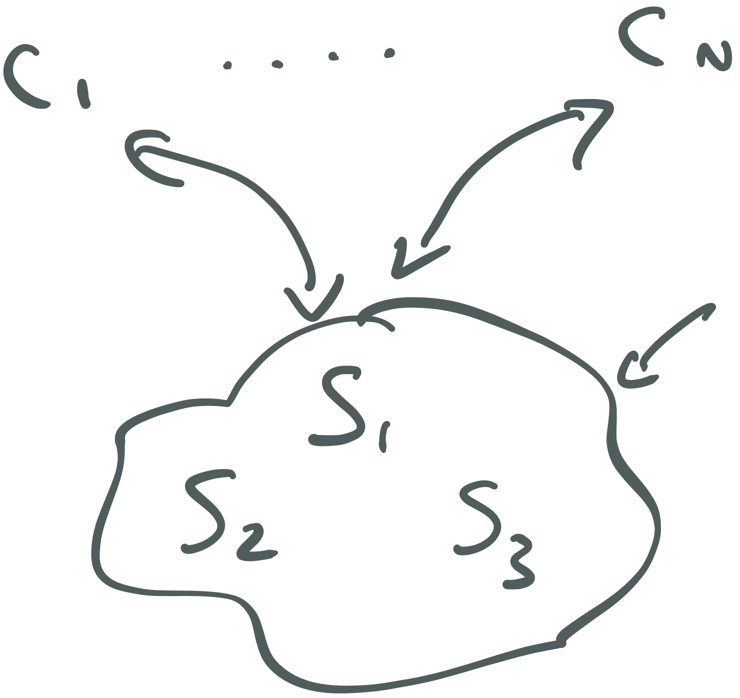
Type 1: Client/server



Today

Type 2: Replicated

Dist System Class (not today)



make server reliable via replication

why are distributed systems different than single machine?

(Partial)

→ Failure

network packet loss

machine crashes

most important

→ Performance

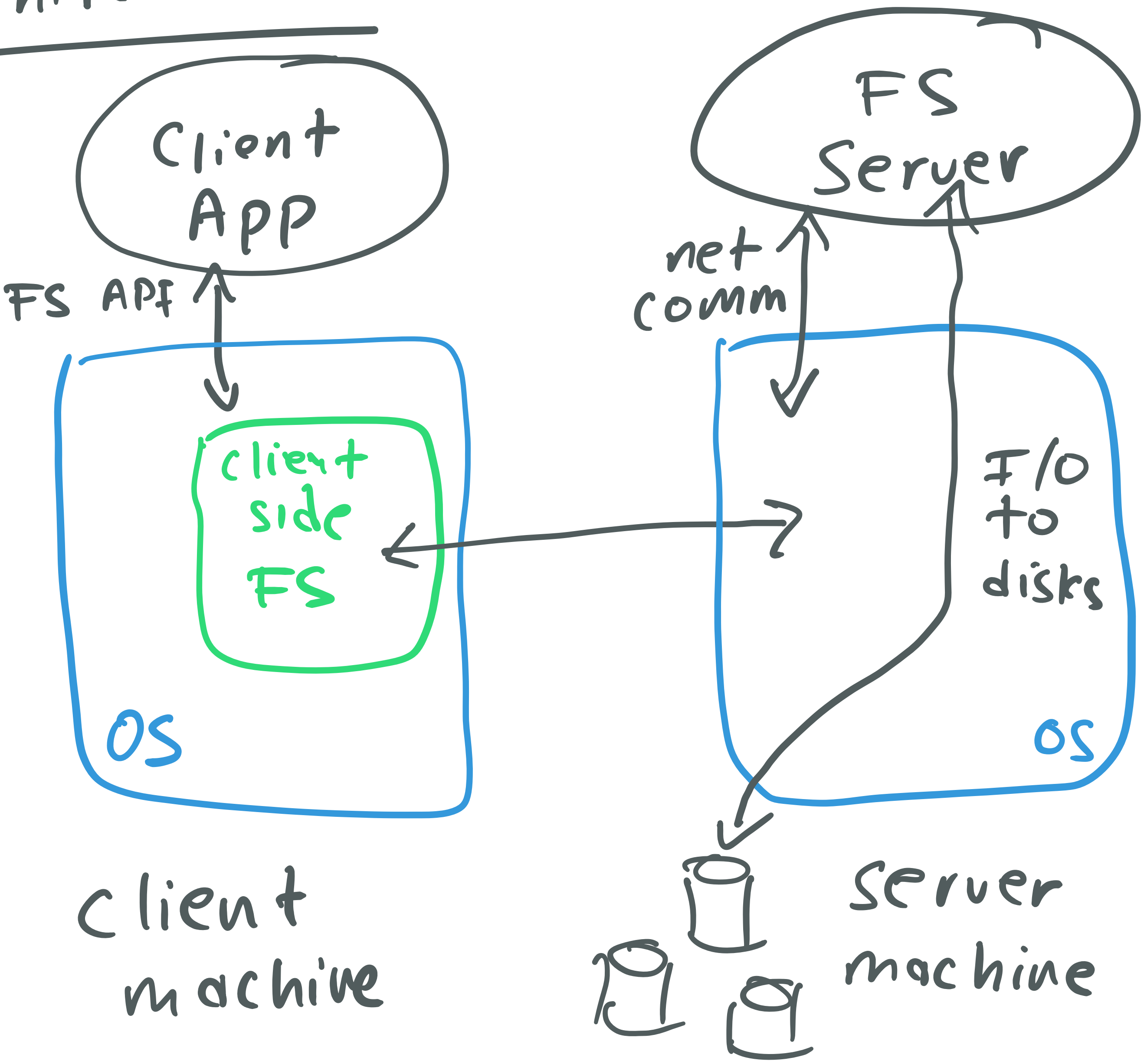
latency, bandwidth

resource mgmt policies

NFS: Outline

- Basics
- Protocol
- From protocol → FS API (open, read/write, close)
- Idempotency + Failure handling
- Client caching

Architecture



Why client/server FS?

→ Sharing

→ Admin (ease of backup, etc.)

NFS approach: client retry

(on any problem)

Protocol:

File Handle

< volume #, inode #, generation # >

which volume

which file in that volume?

deals w/ file getting deleted + inode reused

Now, protocol: key idea is statelesness

all info needed to complete request is in message

Protocol examples:

```
read (file handle, offset, size)
write (file handle, data, offset, size)

        parent
lookup( file handle, name )
         parent
create ( file handle, name)

getattr (file handle)
```

From protocol to FS API

Example : open + read file

```
fd = open ("/a/b.txt", O_RDONLY);
read (fd, buffer, size);
     ⋮
```

what protocol requests?

assume we have root file handle
directory

open

⇒ lookup (root fh, "a")
returns a's file handle (afh)

⇒ lookup (a fh, "b.txt")
returns b.txt file handle (bfh)

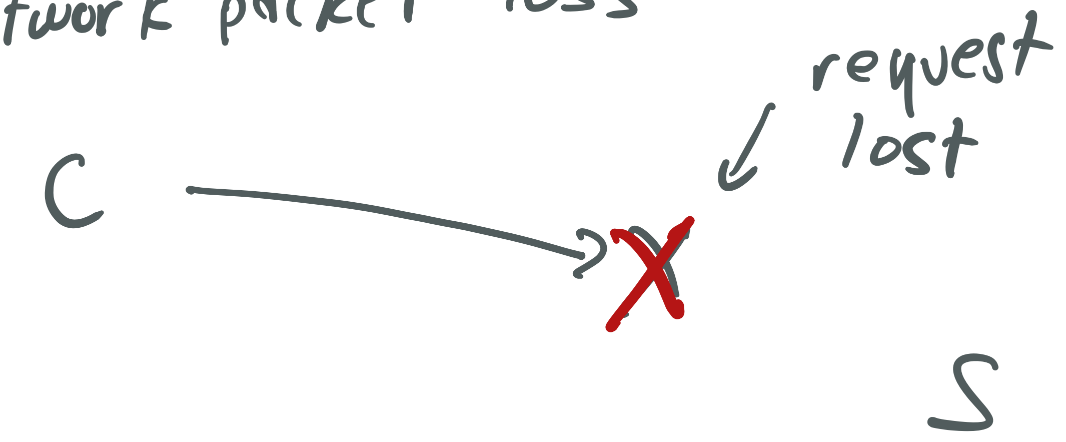
read

⇒ read (bfh, offset, size)
returns data

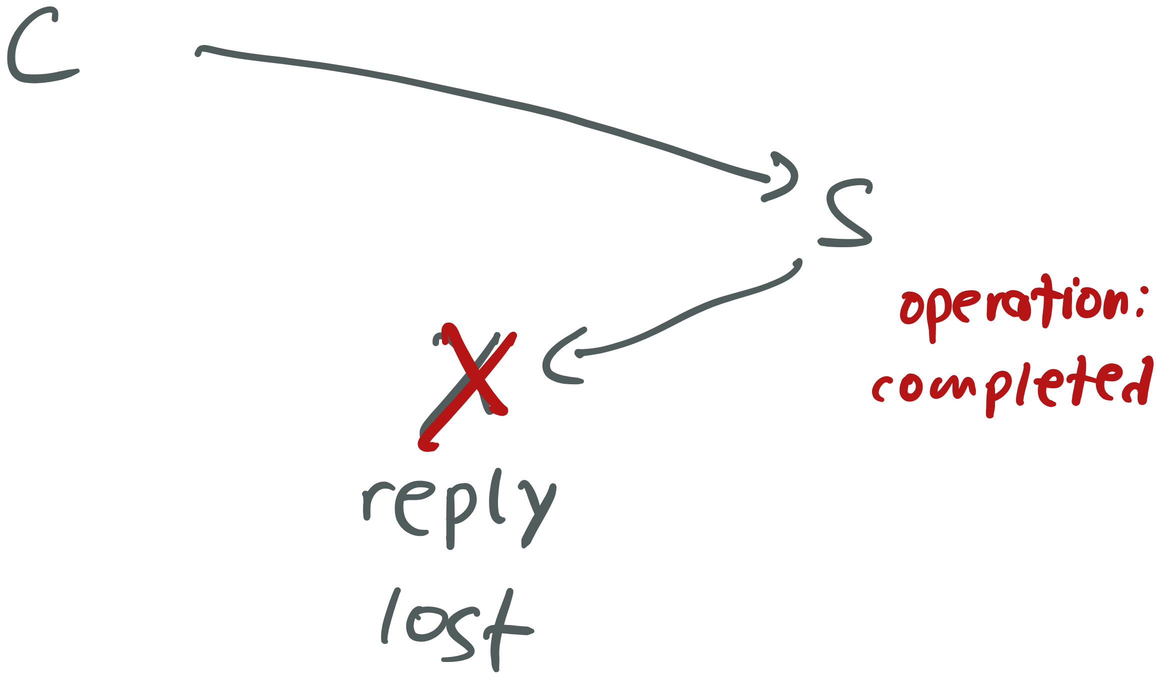
BREAK BREAK

Failure handling:
network packet loss

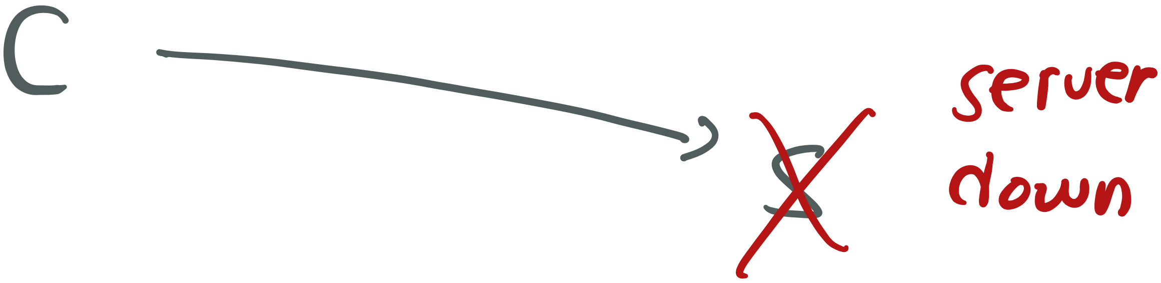
1)



2)



3)



Uniform approach:

timeout: wait to see if you get reply

if no reply, retry

Relies on property : idempotency

operation is idempotent iff doing it N times is equivalent to doing it once

protocol examples:

read, write are idempotent (but not all, e.g., create, delete)

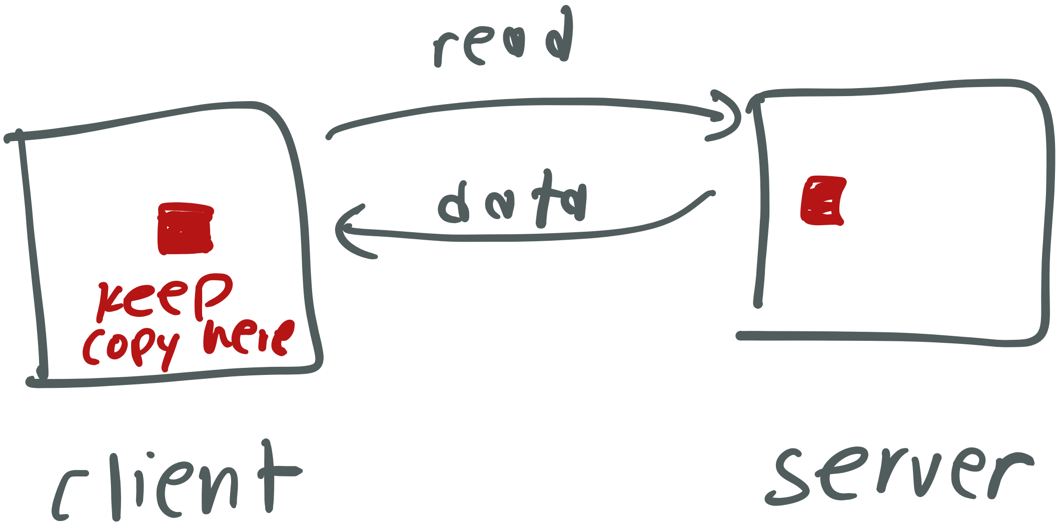
how long to wait?

some factor longer than expected time of reply

also, remember to "back off" (avoid flooding server)

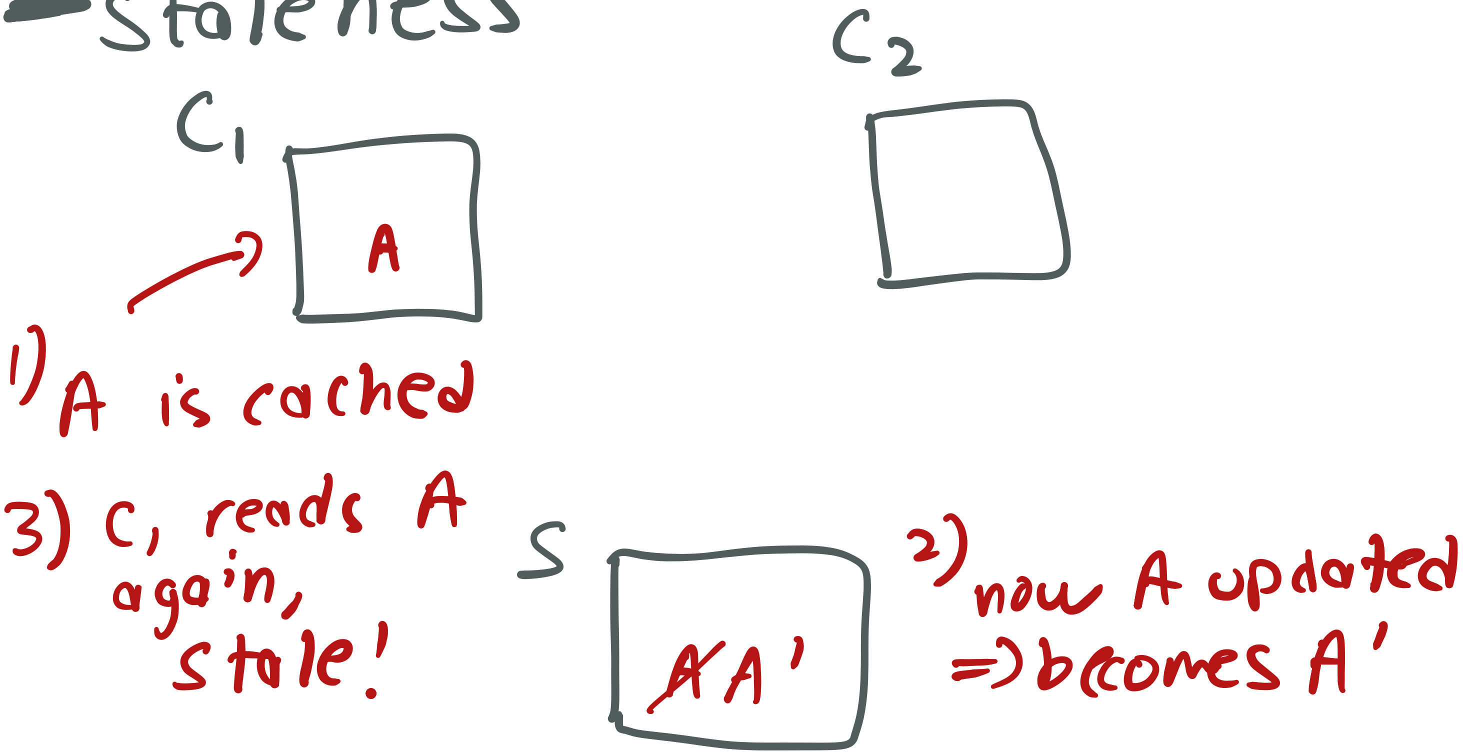
improve availability by making server reboot quickly

Caching (client)

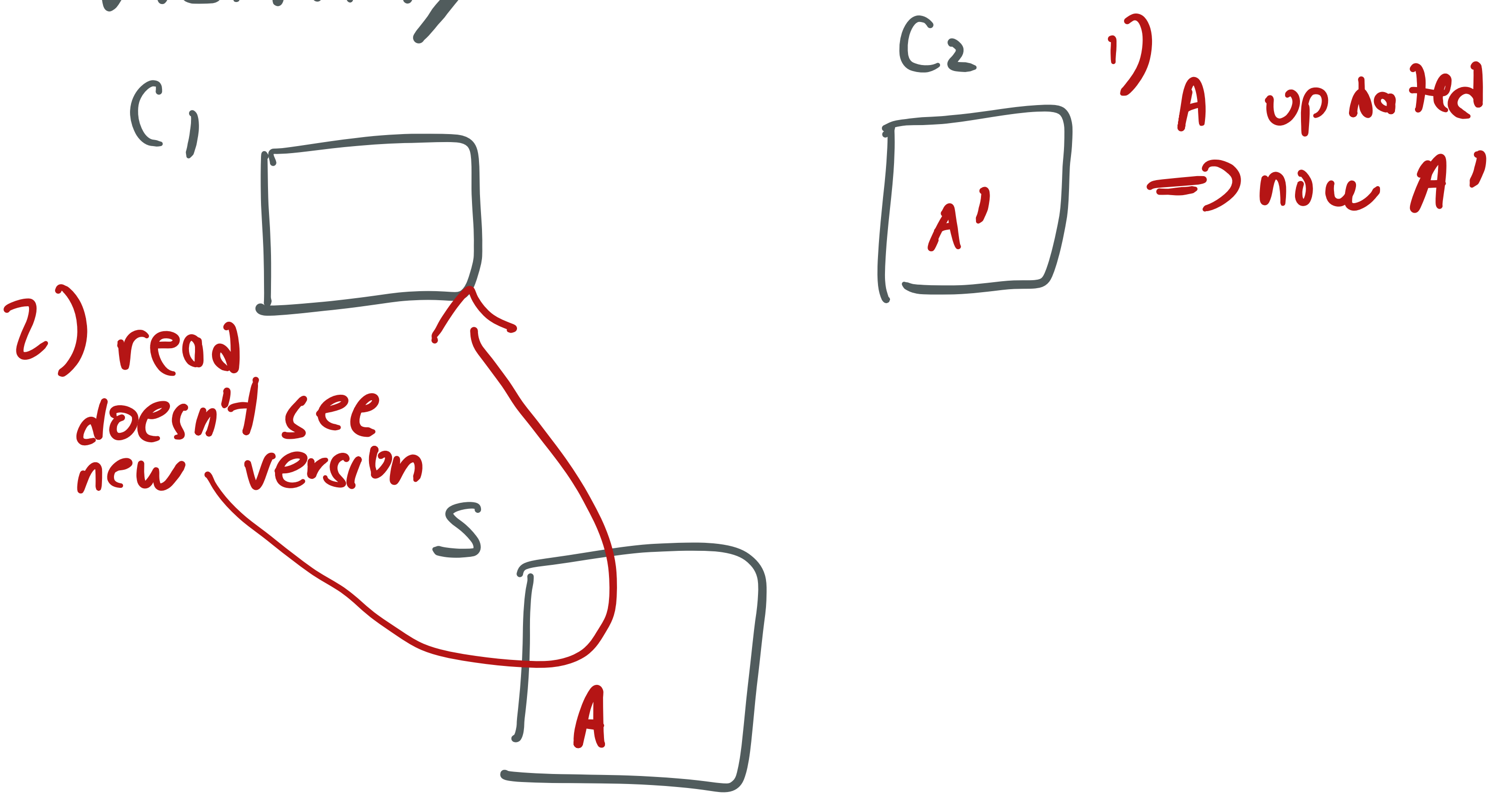


Two problems
→ staleness
→ visibility

Staleness



- Visibility



How to solve?

- Visibility: <u>flush on close</u>
  (ensure we "see" updates after file is closed)

- Staleness:
  check (every so often) if cached copy has changed before using it