LECTURE
AFS

(NFS, block based)

AFS

what is the def?

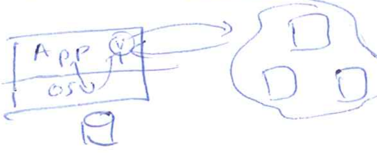
~~main~~ goal: scale

what is the limit of this?

(+ cache consistency, + use local disk!)

FUN IDEA: Reverse engineer AFS!

Prototype 1: AFS v1



notice: all clients have disks!

client (venus):

open() => fetch entire file to local disk

read()/write() => entirely local

close() => if modified, flush entire file to server

writes visible
1) locally
2) open-close (elsewhere)
3) concurrent writing allowed

server

gets full pathname from client (does full lookup)

caching: before using file in cache, check if "ok" w/ server

measure!
=> too slow, does not scale
(60% of calls to server -> validate cache entry)
=> load imbalance
=> cpu load (path traversal)

how to fix?

change the protocol
=> too many interactions w/ server
=> interactions are inefficient

AFS v2

same basic structure (whole-file caching on disks)

consistency: callbacks

client: when file is accessed first time, obtain callback from server (a promise)
=> assume up-to-date unless told otherwise

server: ~~before~~ when mod occurs, notify all clients with callback

name resolution:

old: full path => server (too inefficient, high load on server CPU)

new: piece @ time
"name" => FID (vol #, vnode #, unique ID)

Thus:

open("/x/y/z.doc")
[assume root is local]

For each remote dir "x", "y" =>

1) if in local cache w/ callback, use w/o comm

2) if in cache but w/o callback [e.g. after client reboot, server reboot, net failure]
ask server for new callback

3) not in cache, get it, get callback

Next open => all local

Crash Recovery: harder

=> [why?]

server crash: what happens to all the state?

[what if just a network failure?]

=> (availability vs. consistency vs. partitions) ⇐

Other Fun: mgmt

volumes => allows admin to move these across servers

vol map: cached on clients
(vol => machine) xlation

(Compare: workloads)