Last week:
- HA NFS
- Friday: ML \rightarrow Systems

Replication \Rightarrow \text{Fault Tolerance}
\Rightarrow \text{"hard"} \Rightarrow \text{agreement}

\begin{itemize}
  \item \text{consensus}
  \item \text{electing a leader}
  \item \text{atomic broadcast}
\end{itemize}

\begin{itemize}
  \item \text{values} \rightarrow \text{S}_1 \leftarrow \text{S}_2 \Rightarrow \text{"same"}
\end{itemize}

\text{why hard?}
\Rightarrow \text{concurrency}
\Rightarrow \text{failures} \ (\text{contrast to shared-memory multiprocessors})

\text{fail} \cdots \text{fail}
efficiency

Conceptually:

Dist Sys: Practice vs Theory

possibility of consensus

e.g.: 2-generals problem

other team

some team
Example: Replicated Service

client

Write to S_1

Write to S_2

Read from S_1

Read from S_2

"strict "strong" consistency":

Servers behave as if they are one server.

Approach #1: (broken)

Client: Request just sends req to each server.
Problems:

=> ORDER => replication

=> ordered network
("atomic broadcast")

=> still needs agreement

=> Determinism => challenging

=> divergence

=> Faults/Failure

=> S_i

C_i

=> miss updates

=> divergence

depends on failure model:

=> full stop: down, stay down, and easily
more realistic: detected

→ fail recover:
crash…, reboot
miss updates
how to recover?

Break:
→ (Mid-term)

→ Mon eve (7:15pm)
→ 2 hours long
→ "open mind"
  + pen/pencil

if desired [1 page cheat sheet]
  (2 sides)
  [8.5" x 11"]
  no posters

old exams on web page

"Daily" writeups

"Correct": answers (share reasonable from class w/ all)
Primary/Backup

Often: backup is "passive"

"State" (result) shipped to backup

Solves:
- ordering
- determinism

Hard: how to deal w/ failures

→ failure detection
  "heartbeat" + timeout

Problem:
backup takes over,
but primary still up

("Split brain")
Conservative: 

\[ \Rightarrow \text{ lack of availability} \]

P: \[ B_1, B_2, B_3 \]

new problems? (consistency agreement) \( \Rightarrow \) which backup should take over?

---

Two-Phase Commit:

\[ DB_1 \]

\[ TX_1 \]

\[ DB_2 \]

\[ TX_2 \]

\[ \text{technology} \]

\[ \text{extract pieces} \]
example: dist. storage server

block
server

local locks

independent lock

S₁

or

S₂

API:
read \( \text{addr} = x \) \Rightarrow \text{data}

write \( \text{addr} = y, \text{data} = 2 \)

"single copy" semantics:
"strong" consistency

on each machine:
need locks (mutex)

Read

Write

C \rightarrow S₁ \rightarrow S₂

lock

lock
Problems:

> 1 client:

Solve deadlock: how?

- Ordering: $S_1$ then $S_2$ → high latency
- Say no:

retry (w/ backoff)

C $\rightarrow$ $S_1$ lock

C $\rightarrow$ $S_2$ lock

C $\rightarrow$ lock

nope

reject ("abort")
nodes fail: log some state at particular points