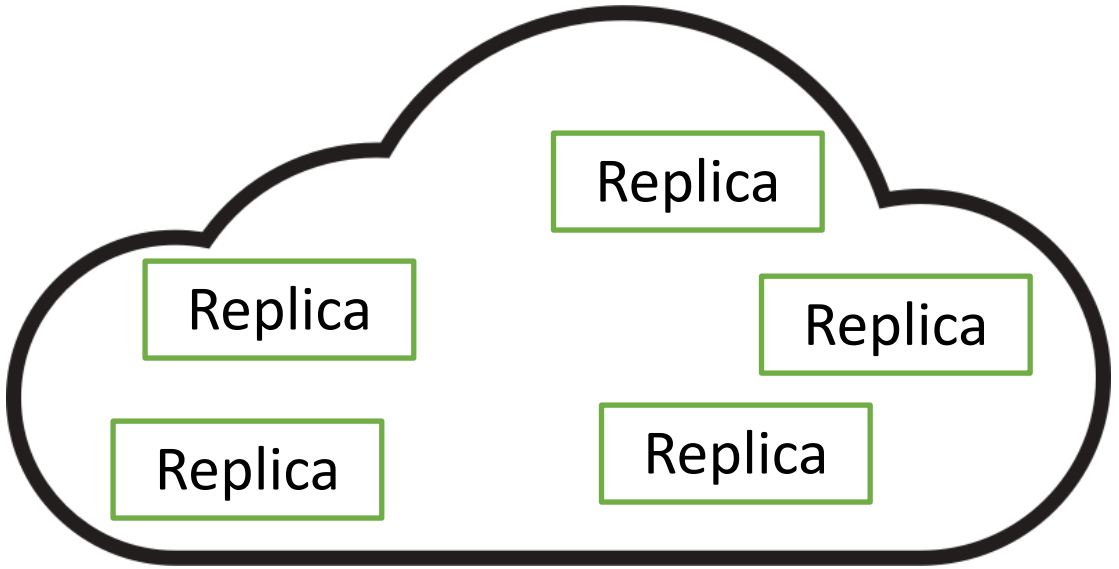
Discussion 06

Spring 2019 – CS 188

Section 2B

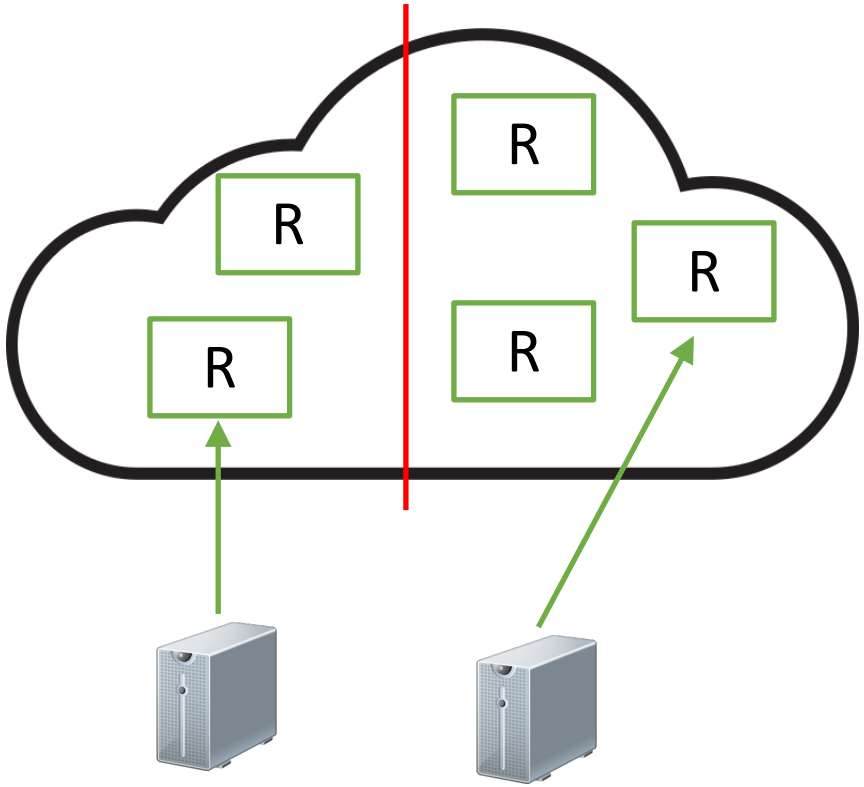
CAP Theorem

- Pick any two: consistency, availability, and partition-tolerance
  - In practice, choose between CP and AP



PACELC

- If partition,
  - Choose availability vs. consistency
- Else,
  - Choose latency vs. consistency
- Unifies two separate tradeoffs that we've talked about



Example1

- **Eventually consistent** key-value store.
  - When two replicas get near each other, they sync with each other.
  - Similar to Shared calendar application in the lecture.
  - If no new updates, all replicas eventually converge to same state.

- Format of updates?
  - <(k1, v1), X>

Updates

- Ordering of updates using Lamport clock
  - <(k1, v1), X, 20> (20 is the timestamp)

- **Resolve conflicts** upon receiving updates from the other node
  - Node A: <(k1, v1), X, 20>
  - Receives update from B: <(k1, v2), Y, 10>

- To be able to "roll back": maintain a log of updates in each replica.
  - Node A: <(k1, v0), X, 5> <(k2, v2), X, 15> <(k1, v1), X, 20>
  - Re-sort the log again
  - Replay

Remember: Quick Syncs

- How to sync without state exchange proportional to size of log?

A
⟨-,10, X⟩
⟨-,20, Y⟩
⟨-,30, X⟩
⟨-,40, X⟩

B
⟨-,10, X⟩
⟨-,20, Y⟩
⟨-,30, X⟩
⟨-,40, Y⟩

- B tells A: highest timestamp for every node
  - E.g., "X 30, Y 40" ← Version Vector
  - In response, A sends all of X's updates after (, 30, X) and all of Y's updates after (, 40, Y)

Sync

- Sync updates, not the state!

- Arguments?
  - Version vector: [X:20, Y:10]
- Return Value?
  - List of updates the caller does not have
  - All updates from X after 20 and from Y after 10.