Bayou / Chord

Oct 6, 2022

[Adapted from Andrew Or and Ion Stoica's original slides]

Context on Bayou: Disconnected Nodes [Stoica]

Early days: nodes always on when not crashed

- Network bandwidth always plentiful.
- Never needed to work on a disconnected node

Now: nodes detach then reconnect elsewhere

- Even when attached, bandwidth is variable
- Reconnection elsewhere means often talking to different replica
- Work done on detached nodes

Bayou

- ”[R]eplicated, [eventually] consistent storage system designed for … portable machines with less-than-ideal network connectivity.”
- System developed at PARC in the mid-90’s
- First coherent attempt to fully address the problem of disconnected operation

Bayou

What is it?

Weakly consistent, replicated storage system

Goals:

Maximize availability, *support offline collaboration*

Minimize network communication

Agree on all values (eventually)

Bayou Update Protocol: Review from Class

- Client sends update to a server
- Updates uniquely identified by:

    <Commit Sequence Number (CSN), Local Timestamp, Node-id

- Updates are either committed or tentative

    CSNs increase monotonically

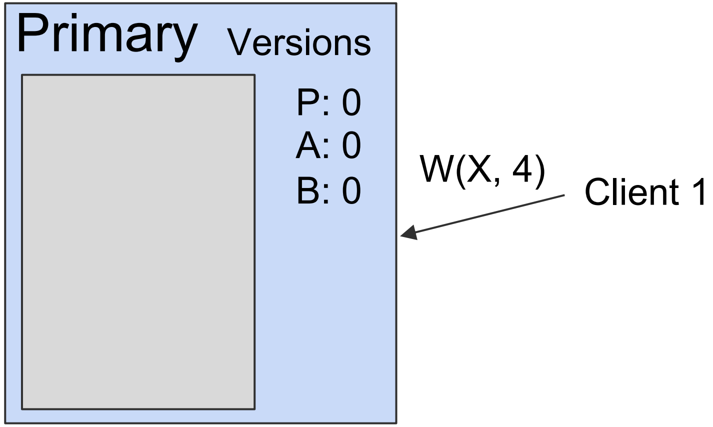
    Tentative updates have commit-stamp = ∞

- Only Primary server can commit updates

    Allocates CSN in monotonically increasing order

    CSN is different from time-stamp

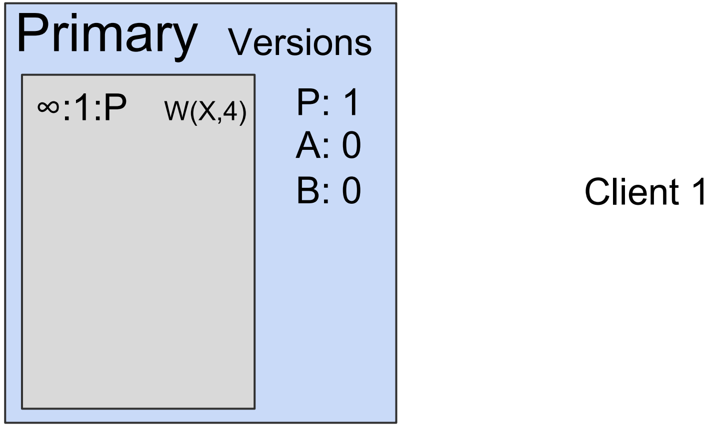
Bayou Writes



Legend

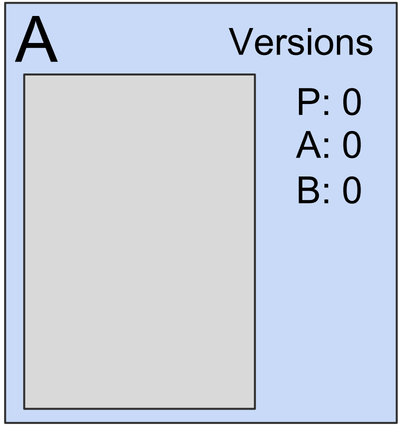
Commit Sequence Number (CSN):Local Timestamp: Node-id

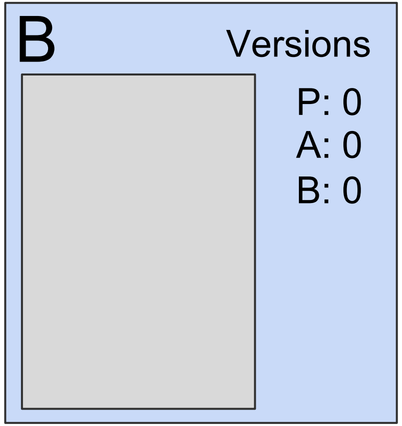
Bayou Writes



Legend

Commit Sequence Number (CSN):Local Timestamp: Node-id

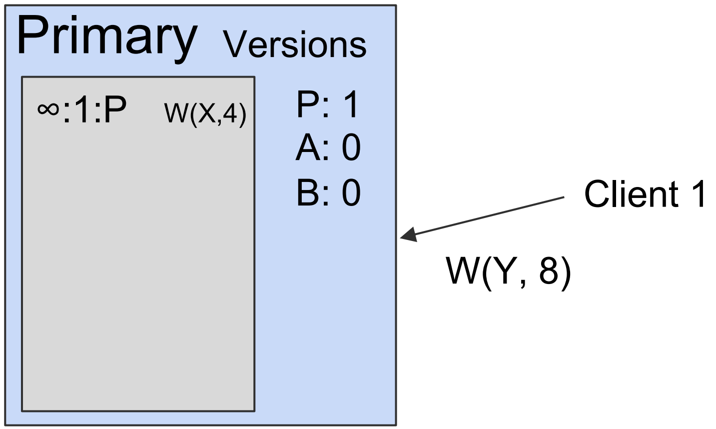


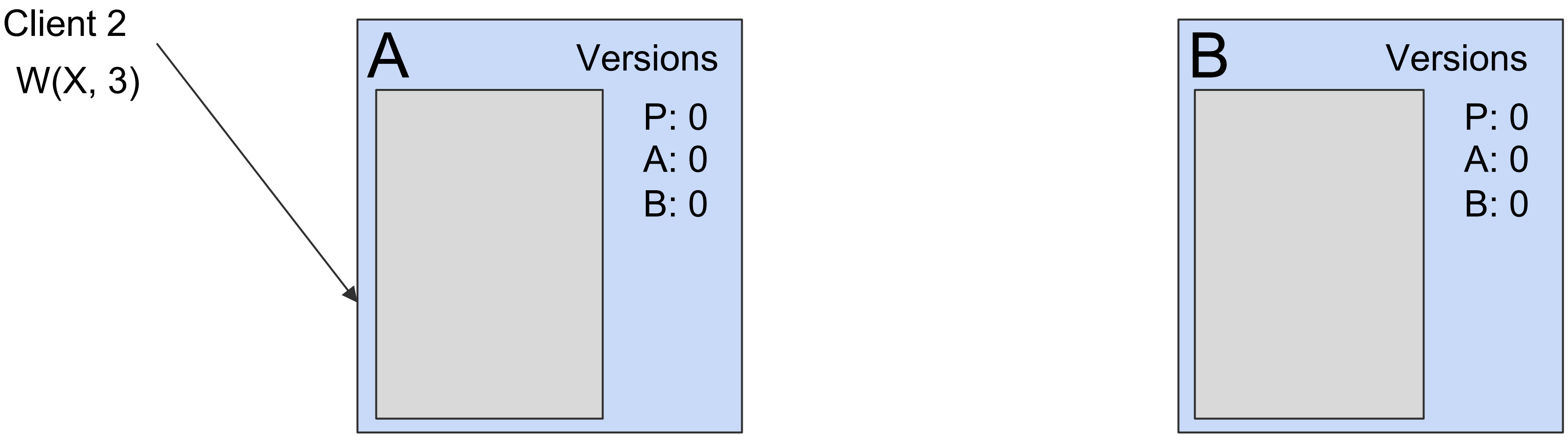


Bayou Writes

Legend

Commit Sequence Number (CSN):Local Timestamp: Node-id

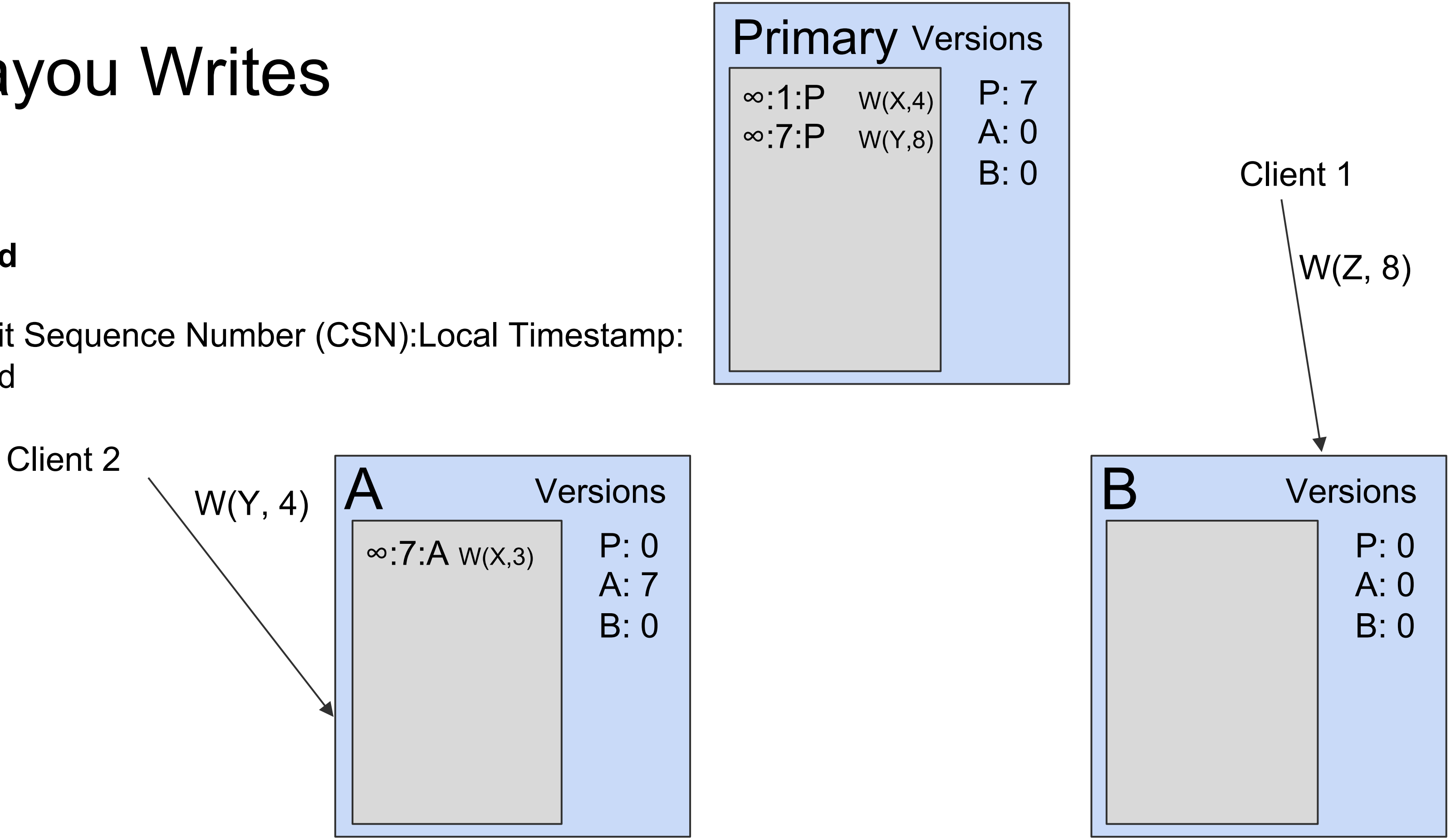




Bayou Writes

Legend

Commit Sequence Number (CSN):Local Timestamp: Node-id



Bayou Writes

Legend

Commit Sequence Number (CSN):Local Timestamp: Node-id

Primary

∞:1:P W(X,4)
∞:7:P W(Y,8)

Versions
P: 7
A: 0
B: 0

A

∞:7:A W(X,3)
∞:12:A W(Y,4)

Versions
P: 0
A: 12
B: 0

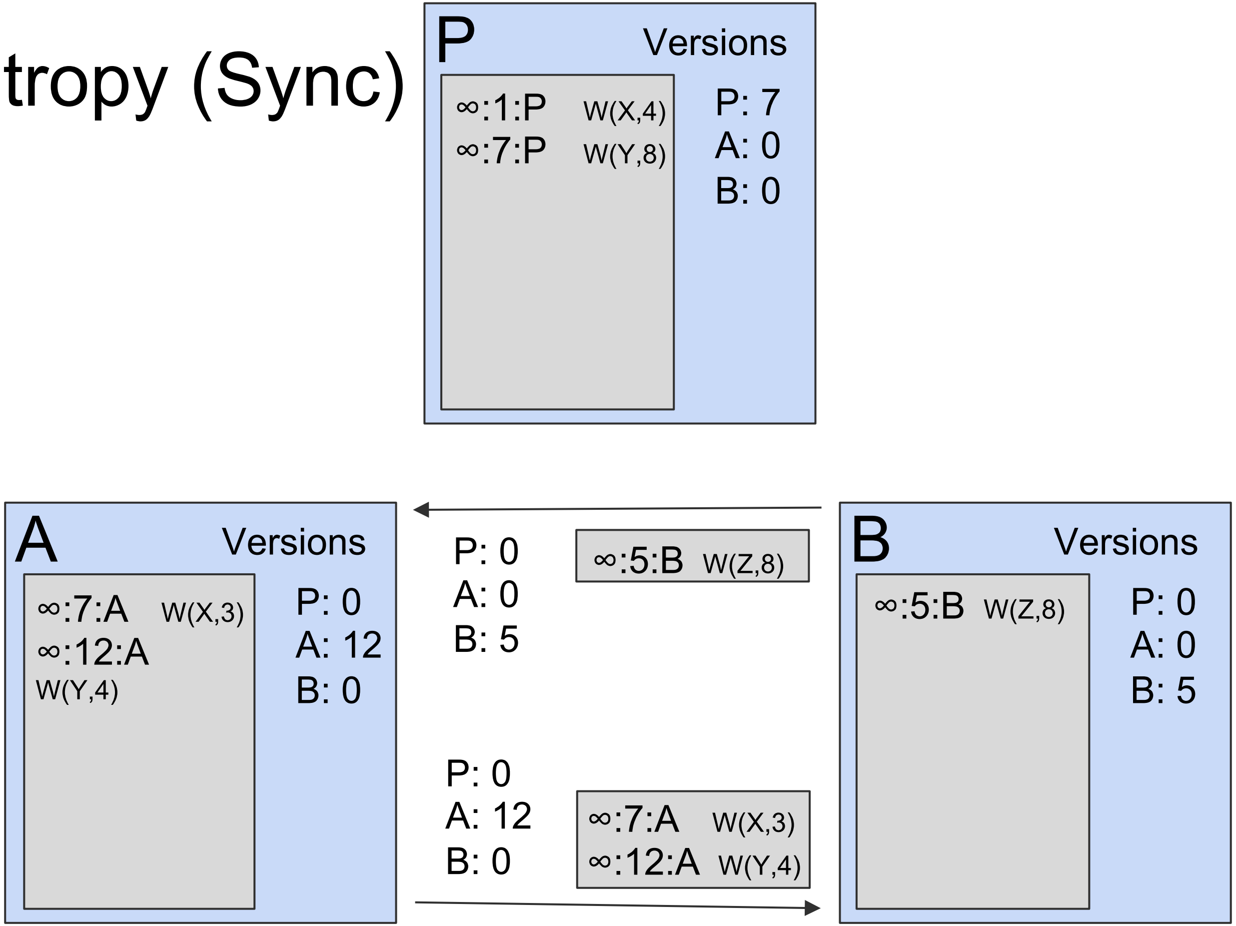
B

∞:5:B W(Z,8)

Versions
P: 0
A: 0
B: 5

Bayou Anti-Entropy (Sync)

Anti-entropy Session
A & B



Bayou Anti-Entropy (Sync)

P

	Versions
∞:1:P W(X,4)	P: 7
∞:7:P W(Y,8)	A: 0
	B: 0

A

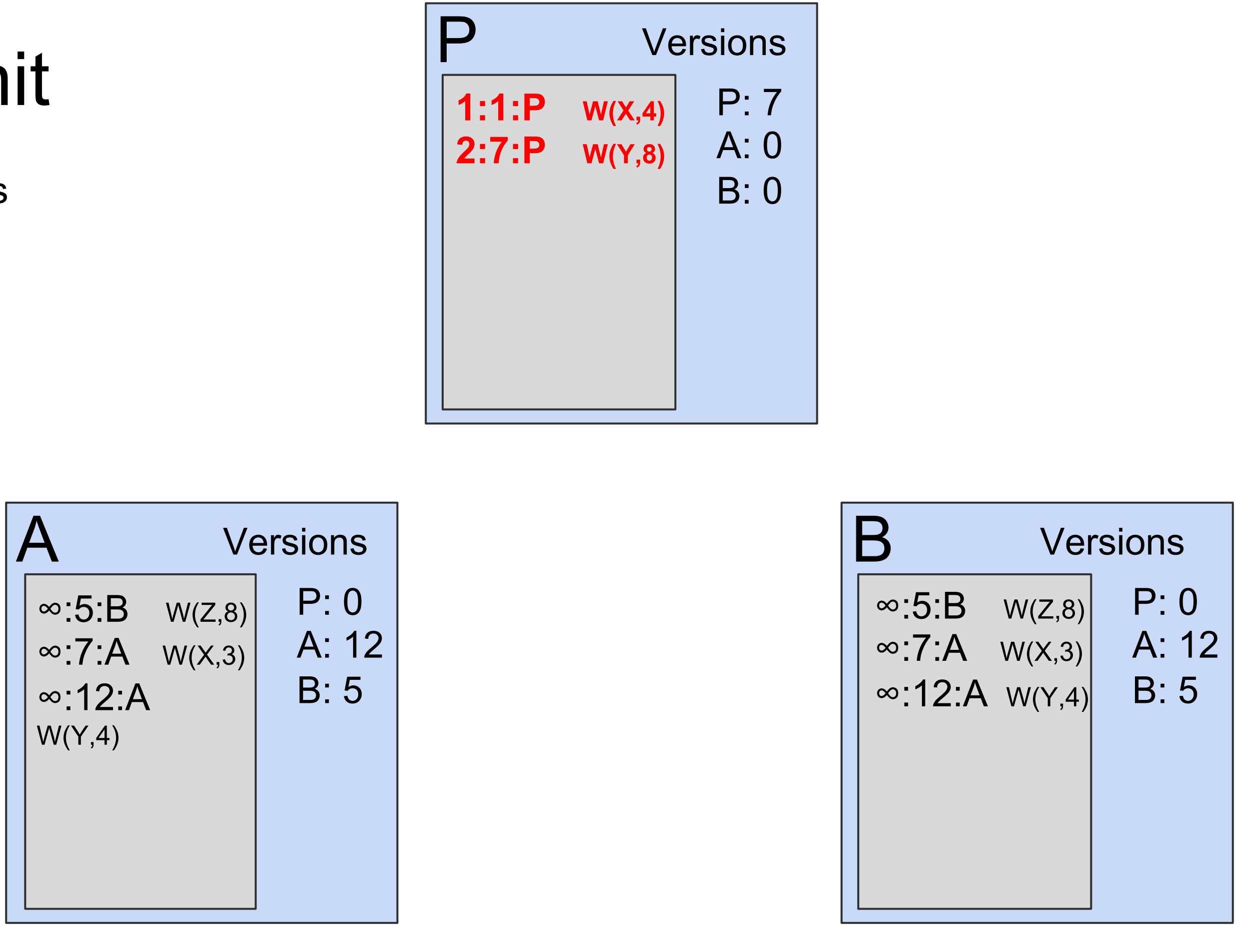
	Versions
∞:5:B W(Z,8)	P: 0
∞:7:A W(X,3)	A: 12
∞:12:A W(Y,4)	B: 5

B

	Versions
∞:5:B W(Z,8)	P: 0
∞:7:A W(X,3)	A: 12
∞:12:A W(Y,4)	B: 5

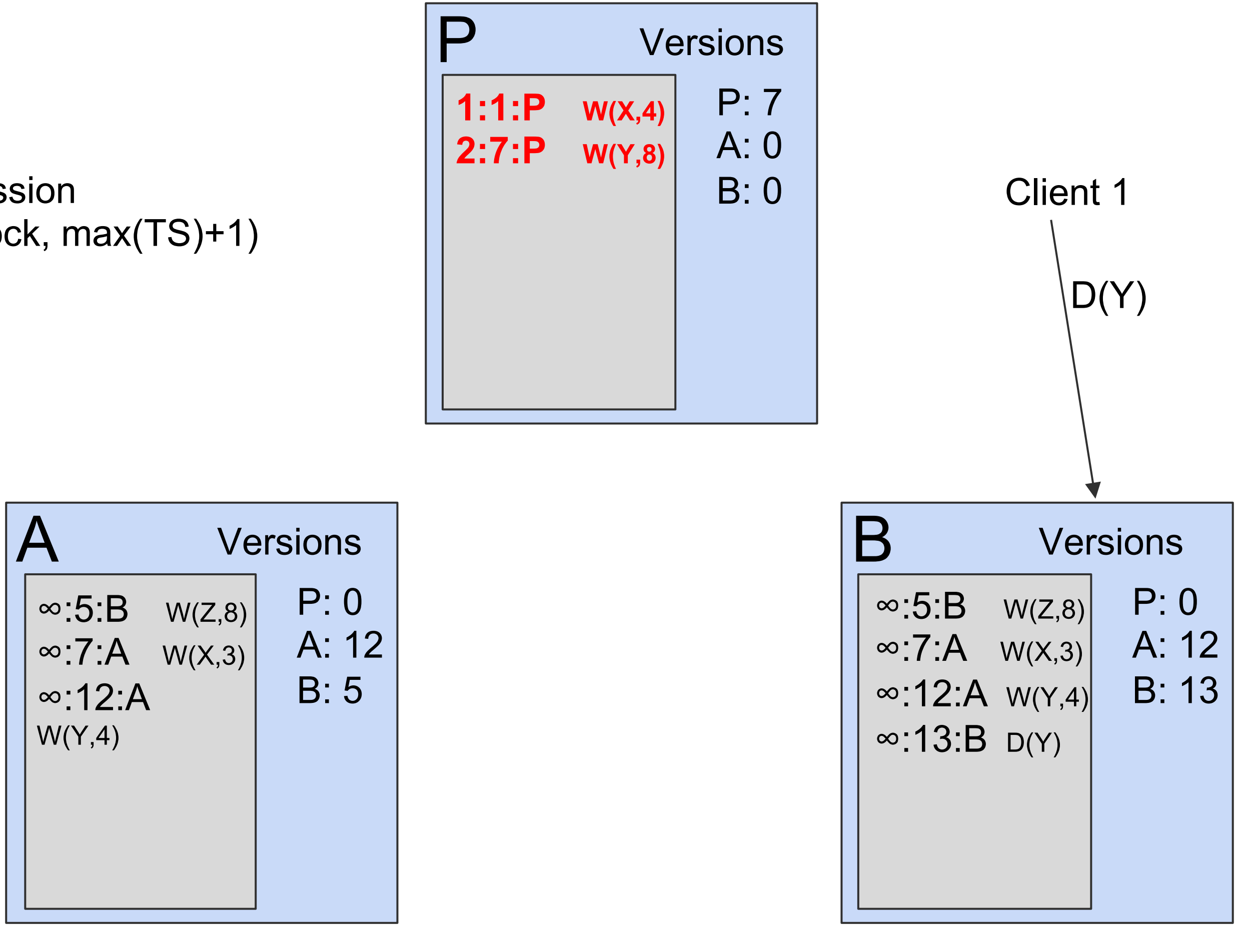
Bayou Commit

Primary commits its entries



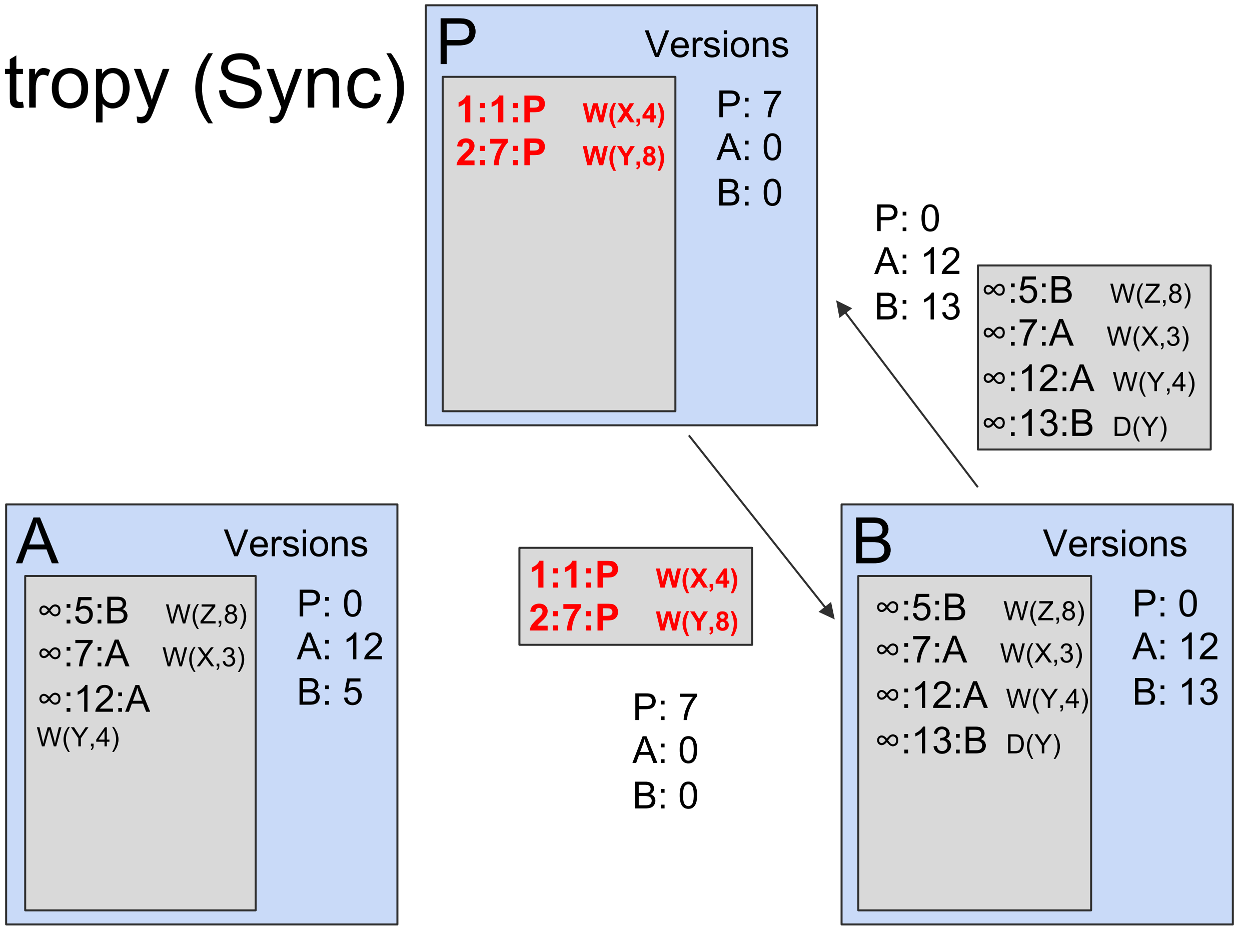
Bayou Write

Write after anti-entropy session
Write timestamp = max(clock, max(TS)+1)



Bayou Anti-Entropy (Sync)

Anti-entropy Session
P & B



Bayou Anti-Entropy

Anti-entropy Session
P & B
Primary respects causality

P

	Versions
1:1:P W(X,4)	P: 7
2:7:P W(Y,8)	A: 12
∞:5:B W(Z,8)	B: 13
∞:7:A W(X,3)	
∞:12:A W(Y,4)	
∞:13:B D(Y)	

A

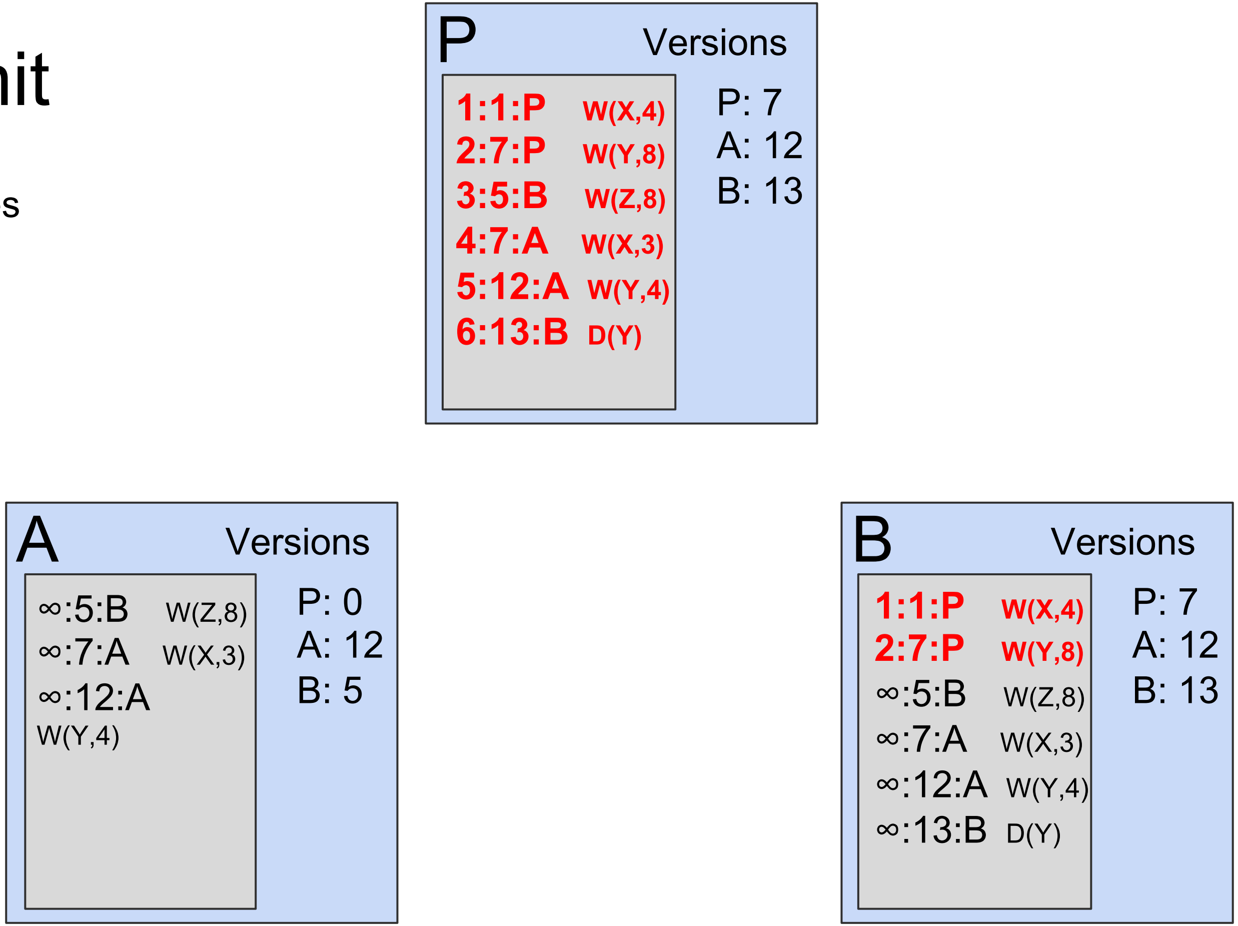
	Versions
∞:5:B W(Z,8)	P: 0
∞:7:A W(X,3)	A: 12
∞:12:A W(Y,4)	B: 5

B

	Versions
1:1:P W(X,4)	P: 7
2:7:P W(Y,8)	A: 12
∞:5:B W(Z,8)	B: 13
∞:7:A W(X,3)	
∞:12:A W(Y,4)	
∞:13:B D(Y)	

Bayou Commit

Primary commits Its entries



Bayou

After a number of commits and anti-entropy sessions (without further writes), all nodes converge on same state.

P

Versions

1:1:P	W(X,4)
2:7:P	W(Y,8)
3:5:B	W(Z,8)
4:7:A	W(X,3)
5:12:A	W(Y,4)
6:13:B	D(Y)

P: 7
A: 12
B: 13

A

Versions

1:1:P	W(X,4)
2:7:P	W(Y,8)
3:5:B	W(Z,8)
4:7:A	W(X,3)
5:12:A	W(Y,4)
6:13:B	D(Y)

P: 7
A: 12
B: 13

B

Versions

1:1:P	W(X,4)
2:7:P	W(Y,8)
3:5:B	W(Z,8)
4:7:A	W(X,3)
5:12:A	W(Y,4)
6:13:B	D(Y)

P: 7
A: 12
B: 13

Context for Chord: Key Value Stores

Amazon:
- Key: customerID
- Value: customer profile (e.g., buying history, credit card, ..)

Facebook, Twitter:
- Key: UserID
- Value: user profile (e.g., posting history, photos, friends, …)

iCloud/iTunes:
- Key: Movie/song name
- Value: Movie, Song

Distributed file systems
- Key: Block ID
- Value: Block

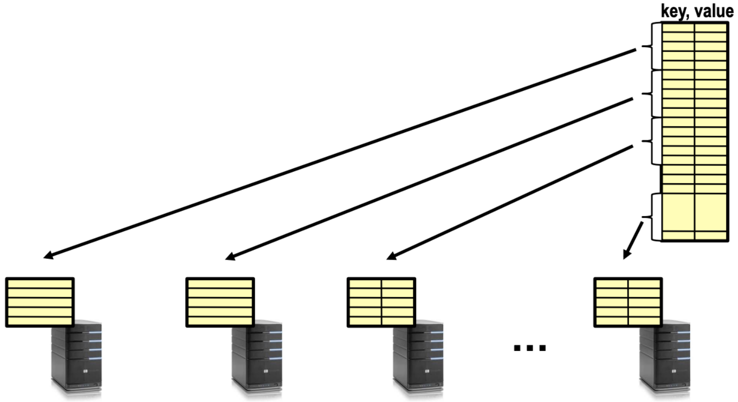
Credit: Ion Stoica's slide deck

Context for Chord: Key Value Stores

Key Value Store

Also called a Distributed Hash Table (DHT)

Main idea: partition set of key-values across many machines



Credit: Ion Stoica’s slide deck

Chord

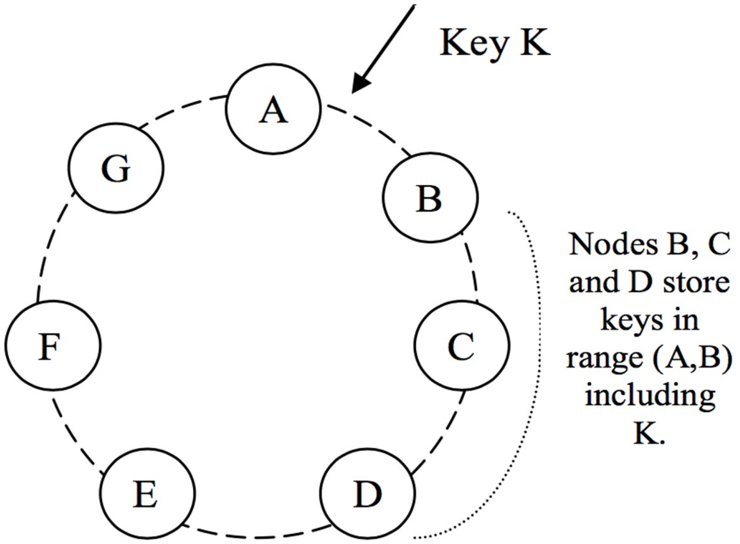
- Chord: “a distributed lookup protocol” for a peer-to-peer distributed hash table [Stoica ‘01]
- *Consistent hashing* for partitioning key space / lookup

Consistent Hashing

Assign each node a random position on the ring

Node owns the preceding key range

Virtual Nodes

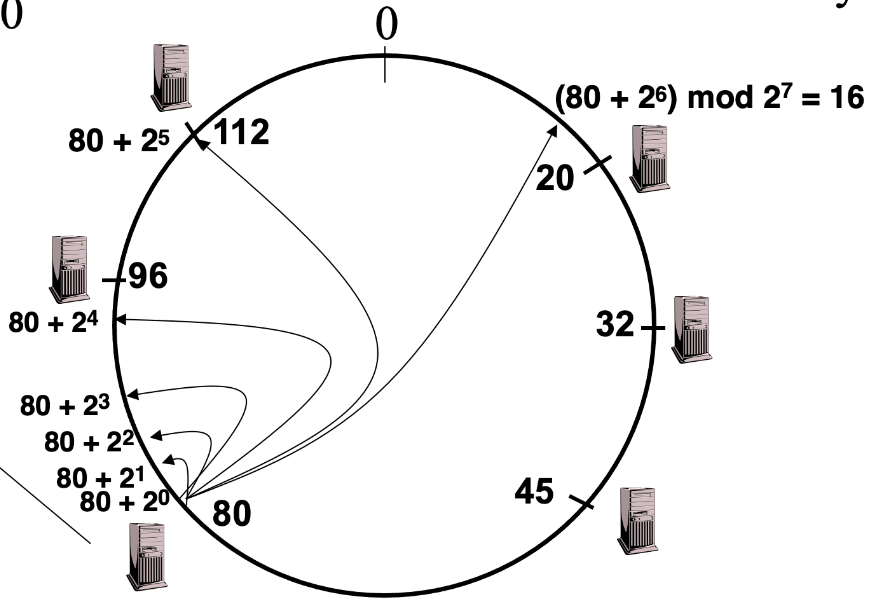


Achieving Efficiency: *finger tables*

Say m=7

Finger Table at 80

i	$ft[i]$
0	96
1	96
2	96
3	96
4	96
5	112
6	20



ith entry at peer with id n is first peer with id >= $n + 2^i \pmod{2^m}$

Credit: Ion Stoica's slide deck