Impossibility Results:
CAP, PRAM, SNOW, PORT, & FLP

COS 418: Distributed Systems
Lecture 20

Wyatt Lloyd
Network Partitions Divide Systems
Network Partitions Divide Systems
How Can We Handle Partitions?

- Atomic Multicast?
- Bayou?
- Dynamo?
- Paxos?
- RAFT?
- COPS?
- Spanner?
How About This Set of Partitions?
Fundamental Tradeoff?

• Replicas appear to be a single machine, but lose availability during a network partition.

• OR

• All replicas remain available during a network partition but do not appear to be a single machine.
CAP Theorem Preview

- You cannot achieve all three of:
  1. Consistency
  2. Availability
  3. Partition-Tolerance

- Partition Tolerance => Partitions Can Happen
- Availability => All Sides of Partition Continue
- Consistency => Replicas Act Like Single Machine
  - Specifically, **Linearizability**
Linearizability (refresher)

• All replicas execute operations in some total order

• That total order preserves the real-time ordering between operations
  • If operation A completes before operation B begins, then A is ordered before B in real-time
  • If neither A nor B completes before the other begins, then there is no real-time order
    • (But there must be some total order)
CAP Conjecture [Brewer 00]

- From keynote lecture by Eric Brewer (2000)
  - History: Eric started Inktomi, early Internet search site based around “commodity” clusters of computers
  - Using CAP to justify “BASE” model: Basically Available, Soft-state services with Eventual consistency

- Popular interpretation: 2-out-of-3
  - Consistency (Linearizability)
  - Availability
  - Partition Tolerance: Arbitrary crash/network failures
CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm A provides all of CAP
CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm A provides all of CAP

Write eventually returns (from A)

Partition Possible (from P)
CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm A provides all of CAP

Write eventually returns (from A)

Partition Possible (from P)

Read begins after write completes
Read eventually returns (from A)
CAP Theorem [Gilbert Lynch 02]

Assume to contradict that Algorithm A provides all of CAP

Write eventually returns (from A)

Not consistent (C) => contradiction!

Read begins after write completes
Read eventually returns (from A)

Partition Possible (from P)
CAP Interpretation Part 1

• Cannot “choose” no partitions
  • 2-out-of-3 interpretation doesn’t make sense
  • Instead, availability OR consistency?

• i.e., fundamental tradeoff between availability and consistency
  • When designing system must choose one or the other, both are not possible
CAP Interpretation Part 2

• It is a theorem, with a proof, that you understand!

• Cannot “beat” CAP Theorem

• Can engineer systems to make partitions extremely rare, however, and then just take the rare hit to availability (or consistency)
Consistency Hierarchy

- Strict Serializability
  - e.g., Spanner

- Linearizability
  - e.g., RAFT

- Sequential Consistency

- Causal+ Consistency
  - e.g., Bayou

- Eventual Consistency
  - e.g., Dynamo
Impossibility Results Useful!!!!

- Fundamental tradeoff in design space
  - **Must** make a choice

- Avoids wasting effort trying to achieve the impossible

- Tells us the best-possible systems we can build!
**PRAM** [Lipton Sandberg 88] [Attiya Welch 94]

- $d$ is the worst-case delay in the network over all pairs of processes [datacenters]

- Sequentially consistent system

- read time + write time $\geq d$

- Fundamental tradeoff between consistency and latency!

- (Skipping proof, see presenter notes or papers)
PRAM Theorem:

Impossible for sequentially consistent system to always provide low latency.
Consistency Hierarchy

- **Strict Serializability** → example: Spanner
- **Linearizability** → example: RAFT
- **Sequential Consistency**
- **Causal+ Consistency** → example: Bayou
- **Eventual Consistency** → example: Dynamo

---

**CAP**

**PRAM 1988 (Princeton)**
Sharding vs. Replication

Sharding Dimension

SNOW

Replication Dimension

CAP

PRAM

A-F

G-L

M-R

S-Z

A-F

G-L

M-R

S-Z

A-F

G-L

M-R

S-Z

21
The SNOW Theorem [Lu et al. 2016]

• Focus on read-only transactions

• Are the ‘ideal’ read-only transaction possible?
  • Provide the strongest guarantees
  • AND
  • Provide the lowest possible latency?
    • (Same as eventual consistent non-transactional reads)

• No 😞
The SNOW Properties

[S]trict serializability

[N]on-blocking operations

[O]ne response per read

[W]rite transactions that conflict
[S]trict Serializability

- Strongest model: real-time + total order
[S]trict Serializability

- Strongest model: real-time + total order

```
SACL  SPhoto

C_R   SACL   SPhoto  C_W

R starts  W starts

R finishes

“Public + Photo A”
“Photo B is private!”

“Public + Photo B”
“Photo A is private!”

W finishes

ACL := Private
Upload Photo B
```
Non-blocking Operations

• Do not wait on external events
  • Locks, timeouts, messages, etc.

• Lower latency
  • Save the time spent blocking
One Response

• One round-trip
  • No message redirection
    • Centralized components: coordinator, etc.
  • No retries
  • Save the time for extra round-trips

• One value per response
  • Less time for transmitting, marshaling, etc.
Write Transactions That Conflict

- Compatible with write transactions
  - Richer system model
  - Easier to program

- Spanner has W
- COPS does not have W
The SNOW Theorem:

Impossible for read-only transaction algorithms to have all SNOW properties

Must choose strongest guarantees OR lowest latency for read-only transactions
Why SNOW Is Impossible

[Intuition]

Assume SNOW

\[ \rightarrow R \]

Violates property S

\[ R_A = \text{new} \]
\[ R_B = \text{old} \]

\( W \) starts
\( \{ \)
\( A := \text{new} \)
\( B := \text{new} \)
\( \} \)

\( W \) invisible

\( W \) visible

\( W \) finishes
SNOW Is Tight

S+N+O:  COPS-DW
S+N+W:  Eiger
S+O+W:  Spanner-RO
N+O+W:  Spanner-Snap

Spanner’s read-only transaction interfaces provide both sides of tradeoff!
Consistency Hierarchy

Strict Serializability → Linearizability → Sequential Consistency → Causal+ Consistency → Eventual Consistency

- e.g., Spanner: SNOW
- e.g., RAFT: CAP
- e.g., Bayou: PRAM
- e.g., Dynamo: CAP
Latency vs. Throughput

• Latency: How long operations take
  – All results so far about latency/availability

• Throughput: How many operations/sec
The NOCS Theorem [Lu et al. 2020]

- Focus on read-only transaction’s latency and throughput

- Are the ‘ideal’ read-only transaction possible?
  - Provide the strongest guarantees
  - AND
  - Provide the lowest possible latency?
  - AND
  - Provide the highest possible throughput?

- No 😞
The NOCS Properties

[N]on-blocking operations

[O]ne response per read

[C]onstant metadata

[S]trict serializability

Same As Simple Reads
The NOCS Theorem:

**Impossible** for read-only transaction algorithms to have all NOCS properties

Must choose strongest consistency OR best performance for read-only transactions
No deterministic 1-crash-robust consensus algorithm exists with asynchronous communication

"FLP"
FLP is the original impossibility result for distributed systems!

• Useful interpretation: no consensus algorithm can always reach consensus with an asynchronous network
  – Do not believe such claims!

• Led to lots and lots of theoretical work
  – (Consensus is possible when the network is reasonably well-behaved)
Conclusion

- Impossibility results tell you choices you must make in the design of your systems

- CAP: Fundamental tradeoff between availability and strong consistency (for replication)

- PRAM: Fundamental tradeoff between latency and strong consistency (for replication)

- SNOW: Fundamental tradeoff between latency and strong guarantees (for sharding)

- NOCS: Fundamental tradeoff between performance (latency and throughput) and strong guarantees (for sharding)