CONSISTENCY TRADEOFFS IN MODERN DISTRIBUTED DATABASE SYSTEM DESIGN

DANIEL J. ABADI, YALE UNIVERSITY

Presented by Shu Zhang

PRIMARY DRIVERS

- **Modern applications require increased data and transactional throughput, which has led to a desire for elastically scalable database systems.**

- **Increased globalization and pace of business has led to the requirement to place data near clients who are spread across the world.**

EXAMPLES

basho

Amazon DynamoDB
Amazon SimpleDB

Cassandra

APACHE HBASE

mongoDB

VOLTDB

Project Voldemort
A distributed database.

WHAT IS CAP THEOREM

Eric Brewer(2000) conjectured that a distributed system cannot simultaneously provide all three of the following properties:

Consistency: A read sees all previously completed writes.

(each server returns the right response to each request)

Availability: Reads and writes always succeed.

(each request eventually receive a response)

WHAT IS CAP THEOREM (CONT'D)

Partition tolerance: Guaranteed properties are maintained even when network failures prevent some machine from communicating with others.

(communication among the servers is not reliable, and the servers may be partitioned into multiple groups that cannot communicate with each other)

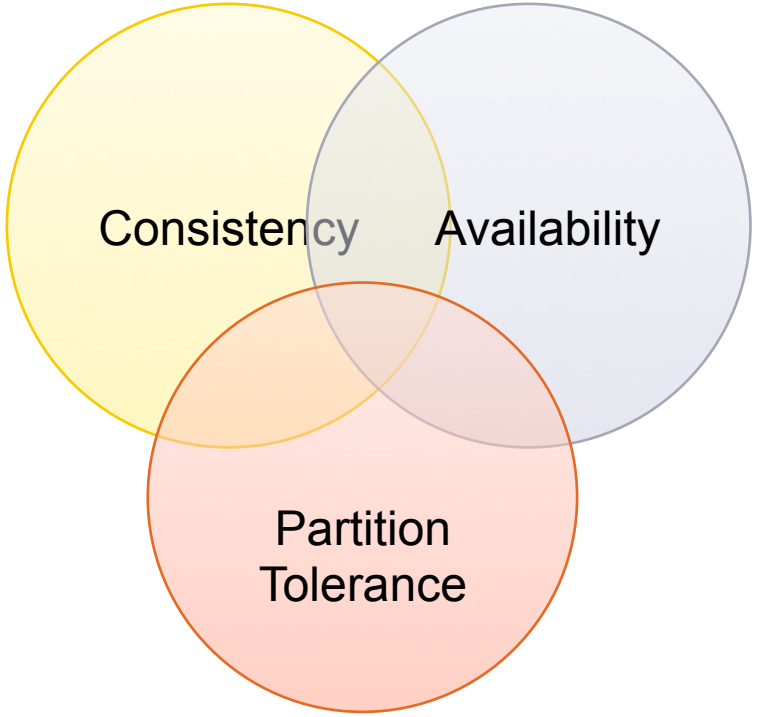
Seth Gilbert and Nancy A. Lynch (2002) proved this in the asynchronous and partially synchronous network models.

CAP IS FOR FAILURES

CA (consistent and highly available, but not partition-tolerant)

CP (consistent and partition-tolerant, but not highly available)

AP (highly available and partition-tolerant, but not consistent)



WHAT'S WRONG

Many modern DDBSs do not by default guarantee consistency.

It is wrong to assume that DDBSs that reduce consistency in the absence of any partitions are doing so due to CAP-based decision-making.

It is wrong to assume that because any DDBS must be tolerant of network partitions, the system must choose between high availability and consistency.

CONSISTENCY/LATENCY TRADEOFF

Amazon Dynamo: e-commerce platform

Facebook Cassandra: Inbox Search

LinkedIn Voldemort: online updates

Yahoo PNUTS: store user data, shopping data

Latency is critical in online interaction.

Tradeoff between consistency, availability and latency exists even when there are no network partitions.

Reason for tradeoff is that a high availability requirement implies that the system must replicate data.

DATA REPLICATION

3 alternatives for implementing data replication

(1) Data updates sent to all replicas at the same time

- Updates do not first pass through a preprocessing layer (lack of consistency)
- Updates first pass through a preprocessing layer (increase latency)

(2) Data updates sent to an agreed-upon location first

- Synchronous (master node waits until all updates sent to replicas)
- Asynchronous (system treats the updates as if it were completed)
- Combination of both (system sends updates to some subset of replicas synchronously, and the rest asynchronously)

DATA REPLICATION (CONT'D)

(3) Data updates sent to an arbitrary location first

- Different from (2) in that the location the system sends updates to is not always the same.

TRADEOFF EXAMPLES

Dynamo, Cassandra, and Riak use a combination of (2)(c) and (3).

PNUTS uses (2)(b)(ii)

PACELC

If there is a partition, how does the system trade off availability and consistency; else, when the system is running normally in the absence of partitions, how does the system trade off latency and consistency.

- **Dynamo, Cassandra, and Riak are PA/EL systems.**
  - If a partition occurs, give up consistency for availability.
  - Under normal operation, give up consistency for lower latency.

- **VoltDB/H-Store and Megastore, HBase are PC/EC systems.**
  - Refuse to give up consistency, will pay the availability and latency costs to achieve it.

PACELC (CONT'D)

- **MongoDB is PA/EC system.**
  - Guarantees reads and writes to be consistent.

- **PNUTS is PC/EL system.**
  - Gives up consistency for latency. If a partition occurs, it trades availability for consistency.

CONCLUTION

Tradeoffs involved in building distributed database systems are complex, and neither CAP nor PACELC can explain them all. Nonetheless, they are worth to be considered when designing modern DDBS.

THE END

Thank you