Consistency

Distributed and Pervasive Systems, MSc

Christian Fischer Pedersen
cfp@ece.au.dk

Department of Electrical and Computer Engineering
Aarhus University

Revised on February 13, 2022

Outline

Motivation

CAP and PACELC Theorems

Consistency models

Eventual Consistency

Outline

Motivation

CAP and PACELC Theorems

Consistency models

Eventual Consistency

Replication reasons

1. **Reliability:** E.g. if a file system has been replicated it may be possible to continue working after data corruption or one replica crashes
2. **Load balancing:** E.g. replicating a highly loaded server and dividing the workload among the processes accessing the server's data
3. **Geo scaling:** E.g. place a data copy close to the process using them to lower access time

Replication challenges

1. Having **multiple copies** may lead to **consistency problems**
2. **Consistency model:** a contract between processes and the data store: if processes agree to obey certain rules, then the store promises to work correctly

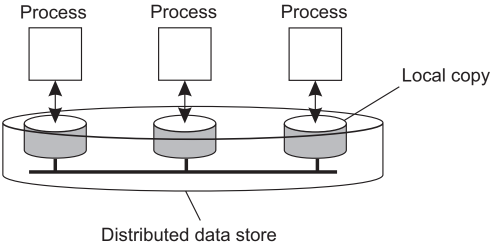


Figure: A logical data store physically distributed and replicated across multiple processes (M. Van Steen)

Outline

Motivation

CAP and PACELC Theorems

Consistency models

Eventual Consistency

Consistency

Definition (Consistency)

All nodes in the system agree on the current state of the system.

Availability

Definition (Availability)

The system is operational and instantly processing incoming requests.

Partition tolerance

Definition (Partition tolerance)

Partition tolerance is the ability of a distributed system to continue operating correctly even in the presence of a **network partition**.

Definition (Network partition)

A network partition is a failure where a network splits into at least two parts that cannot communicate with each other.

CAP theorem

Definition (CAP theorem)

If there is a network **P**artition, the system must choose between **A**vailability and **C**onsistency.

Proof-sketch of CAP theorem.

Assume two nodes, sharing some state. The nodes are in different partitions, thus they cannot communicate. Assume a request wants to update the state and contacts one of the nodes. The node may either:

1. update its local state $\Rightarrow$ inconsistent states, or
2. not update its local state $\Rightarrow$ system unavailable for updates. $\square$

CAP as classification tool

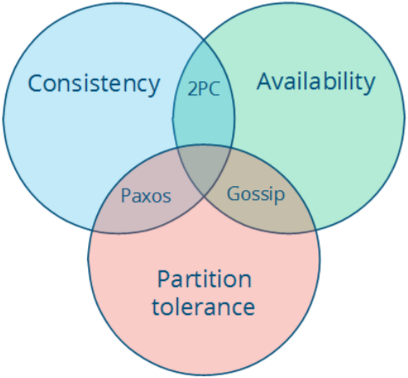


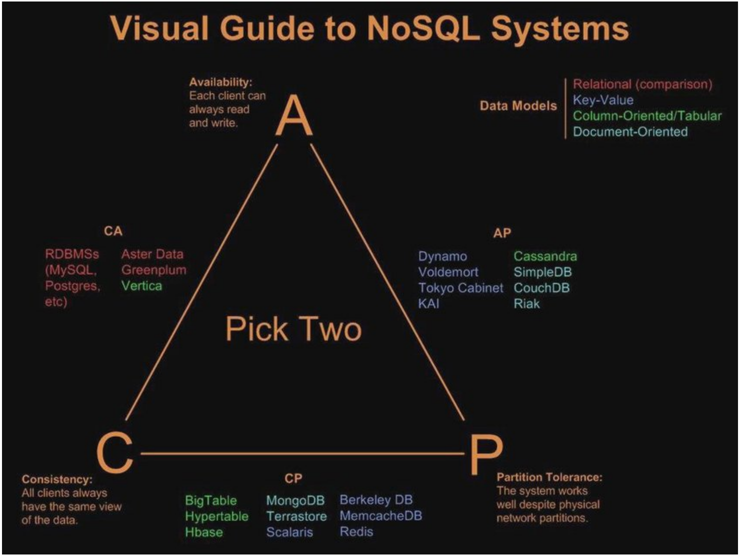
Figure: Center (having all three properties) not achievable (cf. CAP th.).

- **CA:** E.g. full strict quorum protocols
- **CP:** E.g. majority quorum protocols
- **AP:** E.g. protocols using conflict resolution

CAP classification of enterprise systems

- **CA**
  - Relational DB Mgmt Sys: Master/Slave replication, Sharding
  - Terracota: Quorum vote, majority partition survival
  - Google BigTable
- **AP**
  - Amazon Dynamo: Read-repair, application hooks
  - Apache Cassandra: Partitioning, Read-repair
  - Apache Zookeeper: Consensus protocol
  - Apache CouchDB

CAP classification of enterprise systems



PACELC

Definition (PACELC: Partitions, Availability, Consistency, Else, Latency, Consistency)

If there is a network **P**artition, the system must choose between **A**vailability and **C**onsistency (CAP Theorem). **E**lse, when the system is running normally in the absence of partitions, how does the system trade off latency **L** and consistency **C**?

Thus, PACELC theorem elaborates the CAP theorem by stating that even in the absence of partitioning, another trade-off between **L**atency and **C**onsistency occurs.

Outline

Motivation

CAP and PACELC Theorems

Consistency models

Eventual Consistency

Strong consistency model: Definition

Definition (Consistency)

All nodes in the system agree on the current state of the system.

Definition (Strong consistency)

Strong consistency models guarantee that an update is **immediately** propagated to all other nodes. Moreover, if two updates happen concurrently, the updates must be processed in the **same order** at all nodes. Thus, reads are guaranteed to return the most recent data regardless of which node delivers the data. That is, strong consistency guarantees that the system is in a **consistent state** when a transaction has finished and before the next transaction can be handled.

Strong consistency model: Example

Definition (Linearizable (a.k.a. atomic) consistency)

Under linearizable consistency, all operations appear to have executed atomically in an order that is consistent with the global real-time ordering of operations.

Weak consistency model: Definition

Definition (Consistency)

All nodes in the system agree on the current state of the system.

Definition (Weak consistency model)

Weak consistency models guarantee that the system will **eventually** become consistent. Thus, temporarily during the **inconsistency window**, reads are not guaranteed to return the most globally (across all nodes) recent update, but only the most locally (across a subset of nodes) recent update.

Weak consistency model: Example

Definition (Eventual consistency)

Eventual consistency guarantees that the state is eventually agreed upon, but the nodes may disagree temporarily.

Outline

Motivation

CAP and PACELC Theorems

Consistency models

Eventual Consistency

Eventual Consistency

Eventual consistency is a **weak** consistency model:

- A data storage system guarantees that if no new updates are made to a particular object, then **eventually** all accesses will return the last updated value. If no failures occur, the maximum size of the **inconsistency window** can be determined based on factors such as communication delays, system load, and the number of replicas involved in the replication scheme.

Eventual Consistency

The **inconsistency window** is determined via e.g.

- Communication delays
- System load
- Number of replicas involved in the replication scheme

DNS: Most widespread system with eventual consistency

- Name updates distributed according to configured pattern
- DNS server caches are time-controlled
- Eventually, all clients will see the update