This course is outdated and will be removed from our platform. Try our new and improved version of this course.

Visit New Course

PACELC Theorem (New)

Let's learn about the PACELC theorem and its usage.

We'll cover the following ⌃

- Background
- Solution
- Examples

Background#

We cannot avoid partition in a distributed system, therefore, according to the CAP theorem, a distributed system should choose between consistency or availability. **ACID** (Atomicity, Consistency, Isolation, Durability) databases, such as RDBMSs like MySQL, Oracle, and Microsoft SQL Server, chose consistency (refuse response if it cannot check with peers), while **BASE** (Basically Available, Soft-state, Eventually consistent) databases, such as NoSQL databases like MongoDB, Cassandra, and Redis, chose availability (respond with local data without ensuring it is the latest with its peers).

One place where the CAP theorem is silent is what happens when there is no network partition? What choices does a distributed system have when there is no partition?

there is no partition.

Solution#

The PACELC theorem states that in a system that replicates data:

- if there is a partition ('P'), a distributed system can tradeoff between availability and consistency (i.e., 'A' and 'C');
- else ('E'), when the system is running normally in the absence of partitions, the system can tradeoff between latency ('L') and consistency ('C').
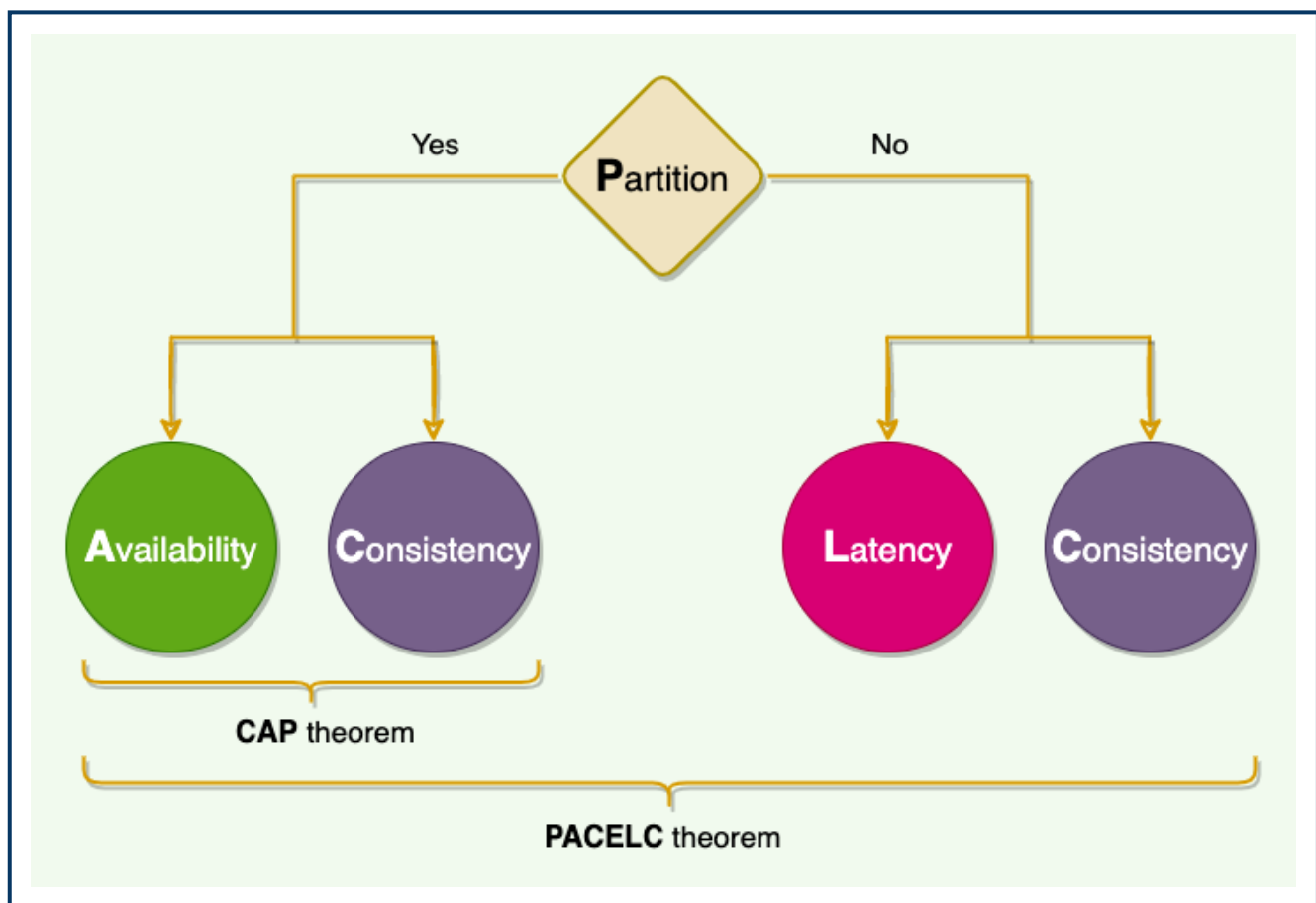
PACELC theorem

The first part of the theorem (**PAC**) is the same as the CAP theorem, and the ELC is the extension. The whole thesis is assuming we maintain high

ELC is the extension. The whole thesis is assuming we maintain high
availability by replication. So, when there is a failure, CAP theorem [clipboard icon] [settings icon]
But if not, we still have to consider the tradeoff between consistency and
latency of a replicated system.

Examples#

- **Dynamo and Cassandra** are **PA/EL** systems: They choose availability
  over consistency when a partition occurs; otherwise, they choose lower
  latency.
- **BigTable and HBase** are **PC/EC** systems: They will always choose
  consistency, giving up availability and lower latency.
- **MongoDB** can be considered **PA/EC** (default configuration): MongoDB
  works in a primary/secondaries configuration. In the default
  configuration, all writes and reads are performed on the primary. As all
  replication is done asynchronously (from primary to secondaries), when
  there is a network partition in which primary is lost or becomes isolated
  on the minority side, there is a chance of losing data that is unreplicated
  to secondaries, hence there is a loss of consistency during partitions.
  Therefore it can be concluded that **in the case of a network partition,
  MongoDB chooses availability, but otherwise guarantees
  consistency**. Alternately, when MongoDB is configured to write on
  majority replicas and read from the primary, it could be categorized as
  PC/EC.

← **Back**

CAP Theorem

Next →

Consistent Hashing (New)

☑ Mark as Completed