Log In Join

Back To Module Home

Distributed Systems

0% completed

Search Module

Conclusion
□

Mark Module as Completed

The CAP Theorem

In this lesson, we explain the CAP theorem with its proof and its extended PACELC theorem.

We'll cover the following□

The CAP Theorem is one of the most fundamental theorems in the field of distributed systems. It outlines an inherent trade-off in the design of distributed systems.

Initial statement of the CAP theorem#

According to the initial statement of the CAP theorem, it is impossible for a distributed data store to provide more than two of the following properties simultaneously: **consistency**, **availability**, and **partition tolerance**.

Consistency#

Consistency means that every successful read request receives the result of the most recent write request.

> The concept of consistency in the CAP theorem is completely different from the concept of consistency in ACID transactions. The notion of consistency as presented in the CAP theorem is more important for distributed systems.

Availability#

Availability means that every request receives a non-error response, without any guarantees on whether it reflects the most recent write request.

Partition Tolerance#

Partition tolerance means that the system can continue to operate despite an arbitrary number of messages being dropped by the network between nodes due to a **network partition**.

> It is very important to understand that *partition tolerance* is not a property we can abandon.

In a distributed system, there is always the risk of a network partition. If this happens, the system needs to decide either to continue operating and compromise *data consistency*, or stop operating and compromise *availability*.

However, there is no such thing as trading off *partition tolerance* to maintain both *consistency* and *availability*. As a result, what this theorem really states is the following.

Final statement of the CAP theorem#

According to the final statement of the CAP theorem, a distributed system can be either *consistent* or *available* in the presence of a network partition.

Proof#

Let’s attempt to prove this theorem simplistically and schematically. Let’s imagine a distributed system consisting of two nodes, as shown in the illustration.

Handling a network partition in a distributed system

This distributed system can act as a plain register with the value of a variable *X*.

Now, let’s assume that there is a network failure that results in a network partition between the two nodes of the system at some point. A user of the system performs a write, and then a read—even two different users may perform these operations.

We will examine the case where a different node of the system processes each operation. In that case, the system has two options:

- It can fail one of the operations, and break the *availability* property.
- It can process both the operations, which will return a stale value from the read and break the *consistency* property.

It cannot process both of the operations successfully, while also ensuring that the read returns the latest value that is written by the write operation. This is because the results of the write operation cannot be propagated from node A to node B due to the network partition.

Importance of the CAP theorem#

The CAP theorem is really important because it helped establish the basic limitations of all distributed systems.

The CAP theorem forces designers of distributed systems to make explicit trade-offs between *availability* and *consistency*. Once the engineers become aware of these properties, they choose the appropriate system.

Categorization of distributed systems based on the CAP theorem#

When we read the literature and documentation of distributed systems, we notice that systems are usually classified into two basic categories: **CP** and **AP**. This classification depends on which property the system violates during a network partition.

Categories of distributed systems according to the CAP theorem

> There is another important thing to note about the CAP theorem: the choice between *consistency* and *availability* needs to be made *only* during a network partition.

Both *consistency* and *availability* properties can be satisfied when the network partition is not present.

Trade-off between latency and consistency#

When no network partition is present during normal operation, there's a different trade-off between *latency* and *consistency*.

To guarantee *data consistency*, the system will have to delay write operations until the data has been propagated across the system successfully, thus taking a *latency* hit.

An example of this trade-off is the single-master replication scheme. In this setting, the *synchronous replication* approach would favor *consistency* over *latency*. Meanwhile, *asynchronous replication* would reduce *latency* at the cost of *consistency*.

PACELC theorem#

The **PACELC theorem** is an extension of the CAP theorem that is captured in a separate article. It states the following:

> "In the case of a *network partition* (P), the system has to choose between *availability* (A) and *consistency* (C) but *else* (E), when the system operates normally in the absence of network partitions, the system has to choose between *latency* (L) and *consistency* (C)."

Categorization of distributed systems based on PACELC theorem#

Each branch of the PACELC theorem creates two sub-categories of systems.

The first part of the theorem defines the two categories we have already seen: CP and AP.

The second part defines two new categories: **EL** and **EC**.

These sub-categories are combined to form the following four categories:

- AP/EL
- CP/EL
- AP/EC
- CP/EC

A system from the AP/EL category prioritizes *availability* during a network partition and *latency* during a normal operation.

In most cases, systems are designed with an overarching principle in mind: usually either performance and availability, or data consistency. As a result, most of the systems fall into the AP/EL or CP/EC categories.

> There are still systems we cannot strictly classify into these categories. This is because they have various levers that can tune the system differently when needed. Still, this theorem serves as a good indicator of the various forces at play in a distributed system.

We can find a table with the categorization of several distributed systems along these dimensions in the associated Wikipedia page.

ACID Transactions

Next

Consistency Models

Mark as Completed

Report an Issue