Jahong Liu

System Design Basics 17: CAP Theorem

CAP Theorem

CAP stands for Consistency, Availability, and Partition Tolerance. The concept suggests that a distributed system can only ensure two out of these three guarantees simultaneously – either Partition Tolerance and Availability, or Partition Tolerance and Consistency, but never all three at once.

In essence, trade-offs are inevitable. The CAP theorem is largely applicable to NoSQL databases, as these databases inherently facilitate replication. As such, a key consideration of this theorem is that it is only applicable to partitioned databases.

This theorem is often misunderstood, largely due to its mode of teaching. Nonetheless, for those keen on further exploration, it is recommended to peruse Martin Kleppmann's blog on the CAP theorem. **python manage.py makemigrations**

Partition Tolerance

A partition in a distributed system arises when a communication breakdown between the leader and follower nodes prevents the leader from updating the follower. Various factors can trigger this, such as network failures, hardware issues, or any other incidents that inhibit some nodes from interacting with others.

If a system demonstrates partition tolerance, it implies that it can persist in functioning despite network failures, thereby avoiding a total system collapse.

In the CAP theorem, partition tolerance is assumed as a given.

Consistency

Consistency, in the context of the CAP theorem, refers to the uniformity of data between the leader and follower nodes. This should not be confused with consistency in ACID.

Consistency ensures that all nodes within the system perceive the data identically at any given moment. In the event of a data update, the clients will always access consistent data.

Regardless of the node from which they fetch the data, the information will remain consistent across all potential data-reading nodes. Every read will retrieve the most recent written data.

If our system remains partitioned and data is written to the leader database, a client reading from the leader database will receive the most recent data. However, since updates to the follower database are blocked, reading from it could yield outdated data. A possible solution could be to render the follower node redundant, ensuring no outdated data is read. This introduces the concept of availability.

Availability

Availability, in the context of the CAP theorem, differs from its definition in ACID. Here, availability signifies that every system request receives a response, be it successful or a failure, regardless of system faults. It ensures that the system stays operational and can manage requests even amid failures.

As mentioned earlier, under the CAP theorem, a system can only possess two of the three properties- Consistency, Availability, and Partition tolerance. Enhancing the system's availability compromises its consistency, meaning every individual node would respond to every valid request.

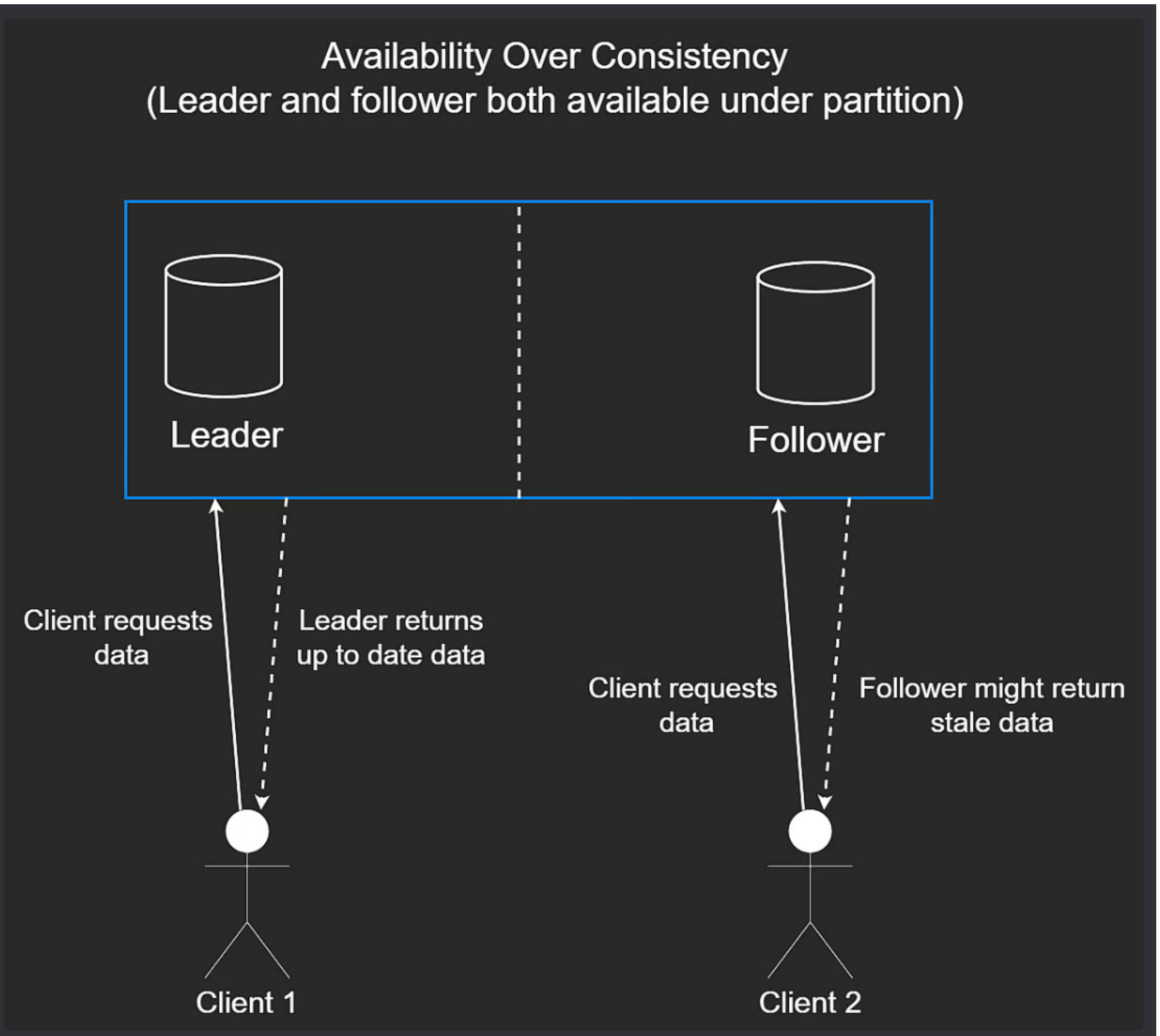
Consistency or availability

Indeed, the decision between prioritizing consistency or availability largely depends on the specific requirements of the application and its associated goals.

Consistency or availability

In systems such as a university's Learning Management System (LMS), high availability might be more crucial. For instance, if a student is attempting to submit an assignment, the LMS must be available to accept the submission. Even if there's a minor delay in updating the grade, it is unlikely to impact the operation negatively.

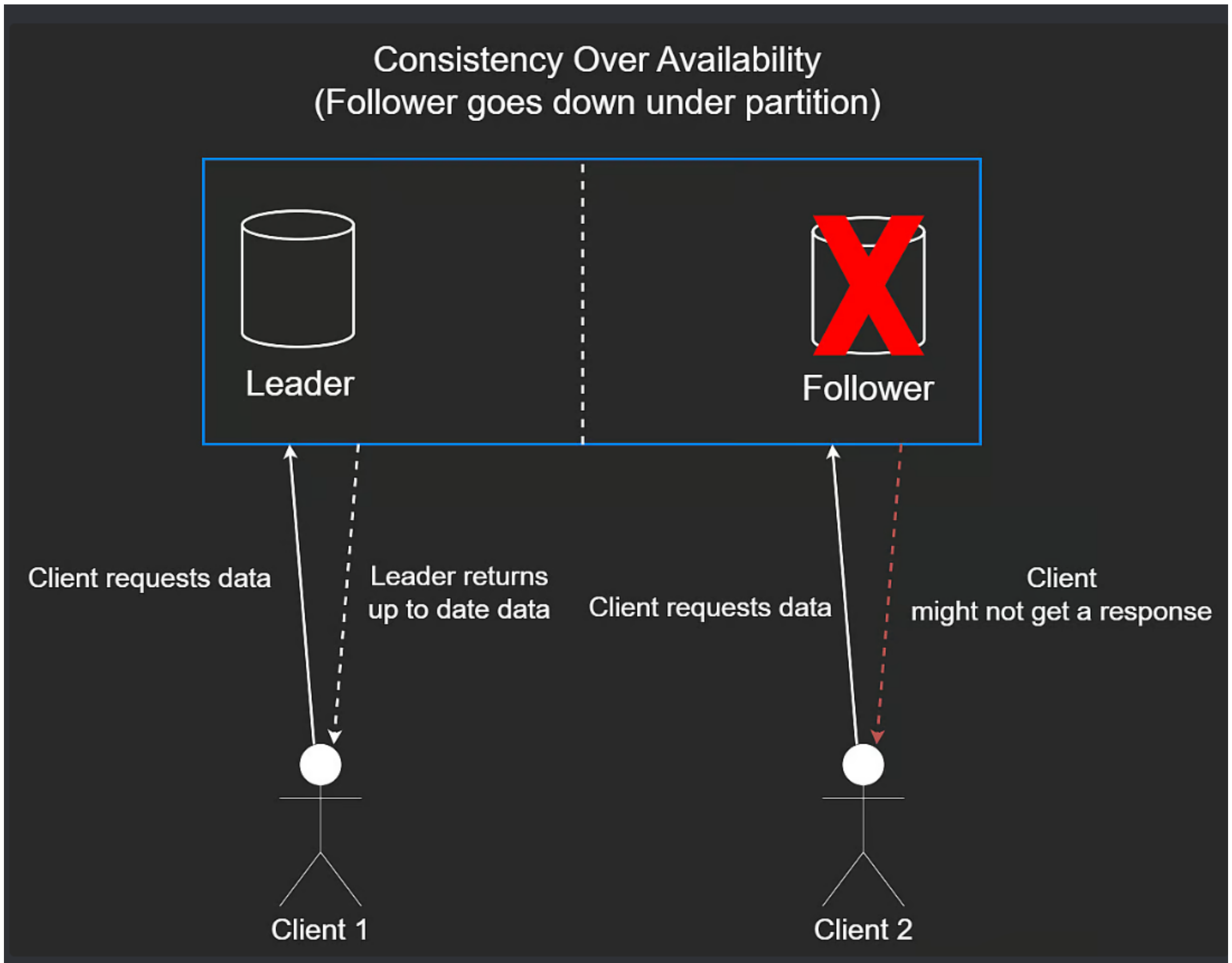
Many modern databases aim to strike a balance between consistency and availability, rather than strictly adhering to being CP or AP. These databases often provide "tunable consistency", permitting te system to dynamically adjust between being more consistent and less available, or more available and less consistent.


Availability Over Consistency
(Leader and follower both available under partition)
Leader
Follower
Client requests data
Leader returns up to date data
Client requests data
Follower might return stale data
Client 1
Client 2

The visual above shows a system that prioritizes availability over consistency. Notice that client 2 receives a response even in the case of a partition between the leader and the follower node.

Choosing consistency over availability

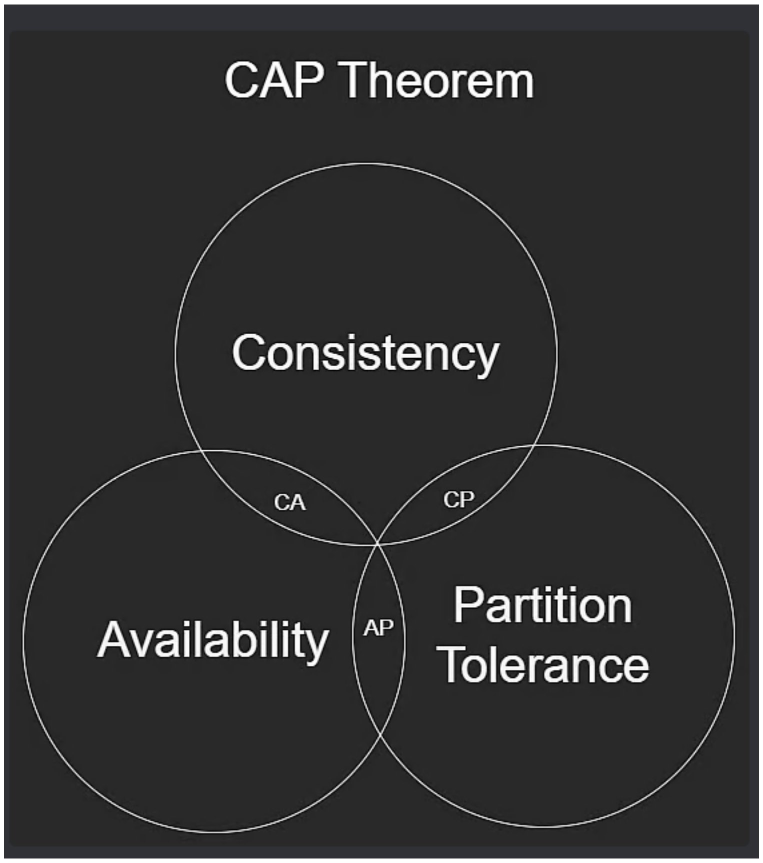
In scenarios where the accuracy of data is absolutely crucial, such as banking and healthcare systems, consistency should be prioritized. In a banking system, the account balance must be correct and consistent across all nodes. If a network partition occurs, it might be acceptable to stop the operations, but still ensuring that the data remains consistent. In a similar manner, in a healthcare setting, having accurate and up-to-date medical records is a matter of life and death and inconsistent data will lead to severe consequences, so prioritizing consistency is critical


Consistency Over Availability
(Follower goes down under partition)
Leader
Follower
Client requests data
Leader returns up to date data
Client requests data
Client might not get a response
Client 1
Client 2

The visual above demonstrates a system that prioritizes consistency over availability. Notice that client 2 might not get a response. The system would not want to risk returning outdated or inconsistent data. This lack of response means we sacrifice availability for consistency.

CAP Theorem Visualized

People often visualize the CAP theorem using a Venn diagram. According to traditional understanding of the CAP theorem, it's not possible to achieve an intersection among all three aspects: availability, consistency, and partition tolerance. Therefore, what's possible is to have an intersection between consistency and partition tolerance, or availability and partition tolerance.



PACELC Theorem (an extension of the CAP theorem)

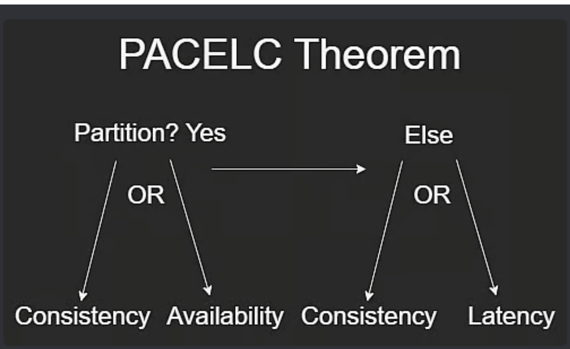
CAP theorem primarily addresses the behavior of a system in the absence of a network partition. However, the PACELC theorem provides further guidance by stating:

“Given P (a network partition), choose A (availability) or C (consistency). Else, favor Latency or Consistency.”

To delve deeper, when a network partition occurs, we have two choices: prioritize either availability or consistency.

Continuing within the leader-follower paradigm, in the absence of a network partition, we must opt to favor either latency or consistency. This means that users will either receive consistent data or data with low latency.

This concept can also be visualized as follows:



> It should be noted that consistency in CAP refers to ensuring the most up-to-date information is available across all nodes in a distributed system, whereas consistency in ACID refers to maintaining transactional integrity and ensuring that transactions do not violate any integrity constraints.

> Choosing between availability and consistency is really a matter of the system design and requirements. For example, some applications can afford to serve slightly outdated data