WIKIPEDIA
The Free Encyclopedia

CAP theorem

In database theory, the **CAP theorem**, also named **Brewer's theorem** after computer scientist Eric Brewer, states that any distributed data store can provide only two of the following three guarantees:[1][2][3]

Consistency
: Every read receives the most recent write or an error. Note that consistency as defined in the CAP theorem is quite different from the consistency guaranteed in ACID database transactions.[4]

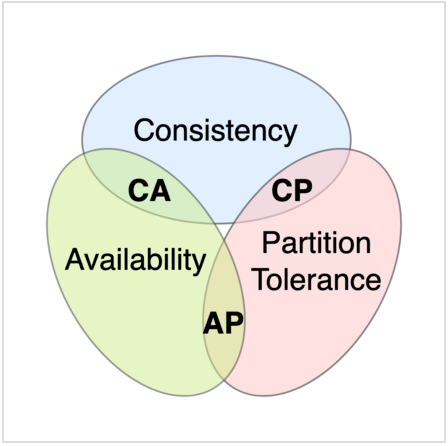
Availability
: Every request received by a non-failing node in the system must result in a response. This is the definition of availability in CAP theorem as defined by Gilbert and Lynch.[1] Note that availability as defined in CAP theorem is different from high availability in software architecture.[5]

Partition tolerance
: The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.

When a network partition failure happens, it must be decided whether to do one of the following:

- cancel the operation and thus decrease the availability but ensure consistency
- proceed with the operation and thus provide availability but risk inconsistency. Note this doesn't necessarily mean that system is highly available to its users.[5]

Thus, if there is a network partition, one has to choose between consistency or availability.



CAP theorem Euler diagram

Explanation

No distributed system is safe from network failures, thus network partitioning generally has to be tolerated.[6][7] In the presence of a partition, one is then left with two options: consistency or availability. When choosing consistency over availability, the system will return an error or a time out if particular information cannot be guaranteed to be up to date due to network partitioning. When choosing availability over consistency, the system will always process the query and try to return the most recent available version of the information, even if it cannot guarantee it is up to date due to network partitioning.

In the absence of a network partition, both availability and consistency can be satisfied.[8]

Database systems designed with traditional ACID guarantees in mind such as RDBMS choose consistency over availability, whereas systems designed around the BASE philosophy, common in the NoSQL movement for example, choose availability over consistency.[9]

Some cloud services choose strong consistency but use worldwide private fiber networks and GPS clock synchronization to minimize the frequency of network partitions. Finally, consistent shared-nothing architectures may use techniques such as geographic sharding to maintain availability of data owned by the queried node, but without being available for arbitrary requests during a network partition.

History

According to computer scientist Eric Brewer of the University of California, Berkeley, the theorem first appeared in autumn 1998.[9] It was published as the CAP principle in 1999[10] and presented as a conjecture by Brewer at the 2000 Symposium on Principles of Distributed Computing (PODC).[11] In 2002, Seth Gilbert and Nancy Lynch of MIT published a formal proof of Brewer's conjecture, rendering it a theorem.[1]

In 2012, Brewer clarified some of his positions, including why the often-used "two out of three" concept can be somewhat misleading because system designers only need to sacrifice consistency or availability in the presence of partitions; partition management and recovery techniques exist. Brewer also noted the different definition of consistency used in the CAP theorem relative to the definition used in ACID.[9][12]

A similar theorem stating the trade-off between consistency and availability in distributed systems was published by Birman and Friedman in 1996.[13] Birman and Friedman's result restricted this lower bound to non-commuting operations.

The PACELC theorem, introduced in 2010,[8] builds on CAP by stating that even in the absence of partitioning, there is another trade-off between latency and consistency. PACELC means, if partition (P) happens, the trade-off is between availability (A) and consistency (C); Else (E), the trade-off is between latency (L) and consistency (C). Some experts like Marc Brooker argue that the CAP theorem is particularly relevant in intermittently connected environments, such as those related to the Internet of Things (IoT) and mobile applications. In these contexts, devices may become partitioned due to challenging physical conditions, such as power outages or when entering confined spaces like elevators. For distributed systems, such as cloud applications, it is more appropriate to use the PACELC theorem, which is more comprehensive and considers trade-offs such as latency and consistency even in the absence of network partitions.[14]

See also

- Fallacies of distributed computing
- Lambda architecture (solution)
- PACELC theorem
- Paxos (computer science)
- Raft (computer science)
- Zooko's triangle
- Inconsistent triad