C

CAP Theorem

Alan Fekete
University of Sydney, Sydney, NSW, Australia

Definition

The CAP theorem states that it is impossible to design a distributed data management platform that provides always consistent (C) data accessed through always available (A) operations if there is the possibility that the set of nodes may partition (P), that is, where there may be nodes which are unable to communicate with each other.

Key Points

The CAP theorem was stated in 1999 by Fox and Brewer [2], and it was proved mathematically by Gilbert and Lynch in 2002 [3]. The CAP theorem is often used to justify NoSQL data platforms that do not keep data consistent, but rather provide eventual consistency (q.v.) or other weak consistency models for replicated data (q.v.).

A common description of the CAP theorem is that a system can have at most two out of the three properties: consistency (C), availability (A), and partition tolerance (P). However, consistency and availability are properties that the system might choose to offer to users, or not; but P is a property of the communication environment, and so it is not up to the system designer to decide about this.

The main idea of the theorem is to suppose (for contradiction) that a system has C, A, and P. Now consider a write (changing x from 0 to 1, say) that is done on one side of a partition; because the system is always available, this write will succeed. Now consider a read of x which is requested on the other side of the partition; this must also succeed (by A), but there is no way to the information from the write to get to the response; so the read will not be able to return 1 but can only report on the previous value 0. Thus the execution doesn't reflect a view of a single consistent data item, contradicting the assumption of C.

Even when communication between nodes is possible, it can take a relatively long time compared to local computation. Abadi [1] extended the CAP idea to PACELC: “if there is a partition (P), how does the system trade off availability and consistency (A and C); else (E), when the system is running normally in the absence of partitions, how does the system trade off latency (L) and consistency (C)?”

Recommended Reading

1. Abadi D. Consistency tradeoffs in modern distributed database system design: CAP is only part of the story. *IEEE Comput.* 2012;45(2):37–42.
2. Fox A, Brewer E. Harvest, yield and scalable tolerant systems. In: *Proceedings of Workshop on Hot Topics in Operating Systems*; 1999. p. 174–8.
3. Gilbert S, Lynch N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News.* 2002;33(2):51–9.