Inadequacies of CAP Theorem

Omkar Patinge
VESIT, University of Mumbai

Vineet Karkhanis
VESIT, University of Mumbai

Amey Barapatre
VESIT, University of Mumbai

ABSTRACT
In today's technical world, we are witnessing a strong and increasing desire to scale systems to successfully complete workloads in a reasonable time frame. As a result of this scaling, an additional penalty of complexity is incurred in the system. In this paper, we have explained the tradeoffs that have to be taken into consideration while designing databases using CAP theorem and the consequences of this tradeoff. Problems of the CAP theorem itself and its limitations are discussed. CAP theorem which was able to meet the demands, back when it was proposed, can't catch up to the current requirements. The problem lies in the open-ended definitions of CAP which are subject to interpretations. PACELC is an alternative to CAP and is able to overcome some of its current problems. PACELC builds on the CAP theorem and it goes one step ahead of CAP by stating that a trade-off also exists, this time between latency and consistency, provides a more complete portrayal of the potential consistency tradeoffs for distributed systems.

General Terms
CAP, PACELC.

Keywords
Consistency, Availability, Partition tolerance, Latency, Tradeoff.

1. INTRODUCTION
Since the inception of CAP theorem 16 years ago it has played a fundamental role in designing and modeling modern distributed database systems. In today's fast-paced world where applications have to scale globally in order to reach masses, briskly and effectively the designing of distributed database is an intricate task. While designing these complex database systems CAP theorem states that there will be tradeoffs between consistency, availability, and partition-tolerance.[1] Since the CAP theorem was first formulated nearly two decades ago, the networked world has changed significantly creating new tradeoffs to explore and new challenges to explore.

Brewer's CAP theorem states that it's not possible to provide all three - consistency, availability and partition-tolerance by distributed computer system. The terminologies associated with CAP theorem are as follows:

1.1 Consistency
The consistency property describes a consistent view of data on all nodes of the distributed system. A consistent system ensures all operations are atomic in nature and the alterations made in any node are reflected in all nodes ensuring a consistent data is maintained. In a replicated distributed database, consistency can be maintained in three ways: the data updates requests are sent to all replicas at the same time, the updates request are sent to a single master node which resolves the request, or to a single arbitrary node first.[2]

1.2 Availability
Availability ensures that every request is answered, even in the case of failures. This must be true for both read and write operations. This property is often zeroed down to bounded responses in reasonable time. Availability gives the notion of 100% uptime; there are limitations to its availability. If there is only a single piece of data on four nodes and if all four nodes die, that data is gone and any request which required it in order to be processed cannot be handled.

1.3 Partition Tolerance

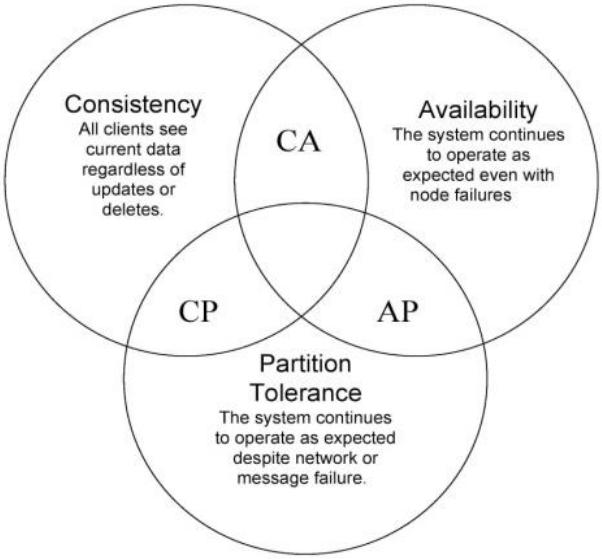

Fig 1: CAP Theorem

Partition tolerance must be possessed by a system to deal with messages losses in a distributed computing system. A partition is a split within the systems in a distributed system which leads to complete loss of communications between affected nodes.[3] Algorithms to deal with maintaining consistency must also deal with effects of partitioning and for a system to be available it must ensure that every node in the partitions should respond to a request.

2. TRADEOFFS
You cannot build a distributed database system that is continually available, sequentially consistent and tolerant to partition pattern. You can build one that has any two of these three properties.

2.1 Consistency-Availability Tradeoff
In modern applications that require distributed database, the primary non-functional requirement is response time. To achieve low response time a sufficient number of replicated databases is needed. This creates an inevitable trade-off between consistency and availability.[4] Since the number of

replicas increases so does the difficulty of maintaining consistency.

If we want to maintain consistency the data is controlled by a single node. If this node fails availability is compromised. Hence this trade-off also occurs when there are no network partitions. In contrary to the trade-off caused by replication, this trade-off is just caused by the possibility of a failure.

2.2 Availability Partition Tolerance Tradeoff

When consistency is fixed the number of queries is directly proportional to the partitioning of a database. The partitioning of database leads to an increase in the number of distributed queries.[5] Hence the availability of data on each node decreases. To overcome this problem the compartmentalization of data should be limited leading to an increase in availability of data on each node. This solution causes a decrease in the need for multiple distributed queries.

2.3 Consistency Partition Tolerance Tradeoff

Availability and partition tolerance is used when users require applications to be responsive in all situations. The responses may not be correct always. In this nodes remain online even if one node can't communicate with another and will resynchronize data once the partition is resolved, but it is guaranteed that all nodes will have the same data. Availability and consistency, data is consistent between all nodes - as long as all nodes are online - and one can read/write from any node and be sure that the data is the same, but if one ever develops a partition between nodes, the data will be out of sync (that is it won't re-sync once the partition is resolved).

3. PROBLEMS

CAP theorem doesn't capture every fundamental tension in a distributed system. CAP theorem has inconsistencies and ambiguities in its definitions, and some problems in its formalization are discussed below. These problems cast doubt on the utility of CAP for its application in practical systems.

3.1 Binary Existence of Availability

Binary existence is convenient for proof purposes but does not closely match our intuitive notion of availability. The traditional CAP theorem's definition doesn't take into account a quantitative measure of network latency.[6] According to the availability property, if the response hasn't arrived, there is still hope that the response will arrive but it does not have an upper bound on latency.

3.2 Inconsequential Trivial Solution

Availability by definition requires only non-failed nodes to respond. In a networked partitioned area even if one node fails at times the availability of a system is hampered. In order to ensure complete availability, one of the solutions proposed is to forcibly make all nodes unavailable.[6] But this trivial solution is unacceptable since it is unnecessarily tampering with rest active nodes.

3.3 Failures

CAP theorem fails to encompass problems like node failure, loss or delay of messages and restart time lapse of nodes other than partition. Fair link loss is possessed by a link if it has a nonzero probability of packet loss. In such a link the lost packets will be delivered by a limited number of repeated attempts ensuring packets reach the destination. The fair link loss is closely associated with mobile networks which are integral to today's application. Problems like node failures, restarts are no longer accidental as much as they are due to attacks on a system. For instance, denial of service attack is common and is one of most notorious attacks on a network operation.

3.4 Probabilistic Consistency

It is also possible to define consistency as a quantitative metric rather than a safety property. For example, Fox and Brewer define harvest as "the fraction of the data reflected in the response, i.e. the completeness of the answer to the query," and investigate the probability outcome being stale, given various assumptions about the distribution of network latencies. However, these stochastic definitions of consistency are not the subject of CAP.

3.5 Partition Tolerance Cannot Be Skipped

In any distributed system partitioning is inevitable. If we assume that one node has 99.9% chance of not failing in a particular time period then five such nodes in a cluster will have a probability of 99.5% chance of failure.[7] Thus one cannot sacrifice partition tolerance. Therefore there is an inevitable choice between availability and consistency.

4. PACELC

In 2010 Daniel J. Abadi proposed PACELC overcoming shortcomings of CAP theorem. The CAP theorem fails to take into account the latency consistency trade off which is always prevalent.

Abadi states PACELC because "Ignoring the consistency/latency tradeoff of replicated systems is a major oversight (in CAP), as it is present at all times during system operations, whereas CAP is only relevant in the arguably rare case of a network partition."

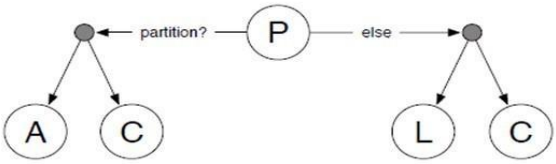

Fig 2 : PACELC Model

If there is partition (P) in the system there is a tradeoff between availability (A) and consistency (C) else (E) there will be a tradeoff between latency (L) and consistency. This gives four alternatives to any system- P+A with E+L/E+C and P+C with E+L/E+C.

DDBS	P+A	P+C	E+L	E+C
Dynamo	Y		Y	
Cassandra	Y		Y	
Riak	Y		Y	
Voldemort		Y		Y
Megastore		Y		Y
PNUTS		Y	Y	

Fig 3 : DDBS in PACELC metrics

In PA/EL systems if partition takes place then we choose availability over consistency and if network partition is absent then we choose latency over consistency. The example of

such system is dynamo, Cassandra and Riak etc. PA/EC system chooses availability over consistency when a partition occurs else it prefers to have consistency over latency. MongoDB is an instance of such a system. PC/EL systems prefer consistency over availability in presence of partition and latency in its absence. PC/EC systems like Megastore, Voldemort in absence of network partition selects consistency over latency and consistency over availability in presence of partitions.

5. CONCLUSION

The CAP theorem provided a base for designing and modeling of databases but it fails to meet the current needs. The problems are self-imposing, majorly because definitions of CAP theorem are open to interpretation. Taking into consideration present day database needs, PACLEC is introduced. PACELC builds on the CAP theorem. PACELC, however, goes further and states that a tradeoff exists between latency and consistency, even when partitions aren't present, thus providing an array of the potential of the potential consistency tradeoffs for distributed systems.

6. REFERENCES

[1] Simon S.Y. Shim,2012, The CAP Theorem's Growing Impact.

[2] Daniel J. Abadi, Yale University,2012, Consistency Tradeoffs in Modern Distributed Database System Design.

[3] Seth Gilbert, Nancy A. Lynch,2012 Perspectives on the CAP Theorem.

[4] Eric Brewer,2012, CAP Twelve Years Later: How the "Rules" Have Changed.

[5] Chao Kong, Ming Gao, Weining Qian, Rong Zhang* , Minqi Zhou, Xueqing Gong and Aoying Zhou,2015, ACID Encountering the CAP Theorem.

[6] A Critique of the CAP Theorem, University of Cambridge, A Critique of the CAP Theorem.

[7] Balla Wade Diack,Samba Ndiaye and Yahya Slimani, 2013, CAP Theorem between Claims and Misunderstandings: What is to be sacrificed?