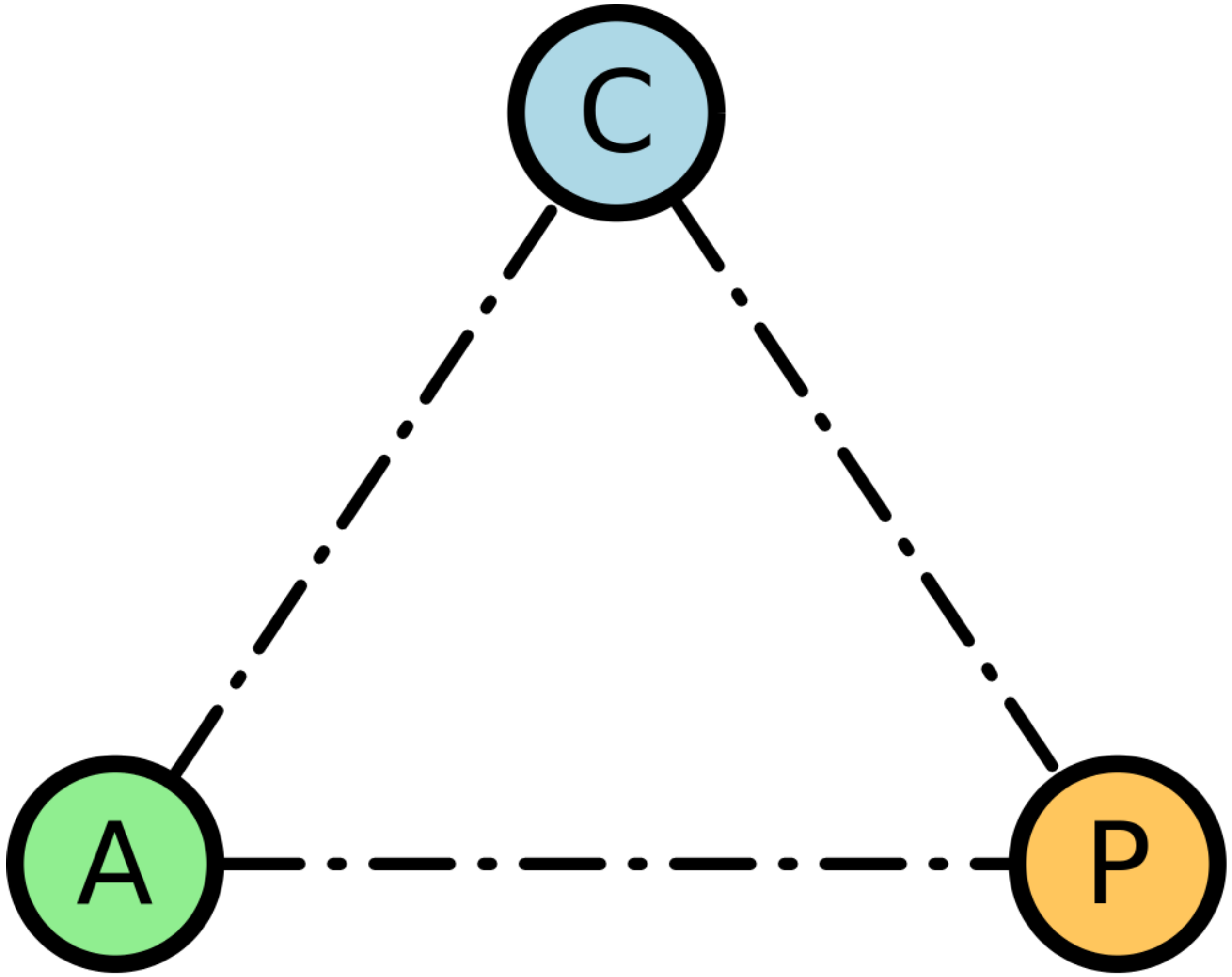
CAP Theorem

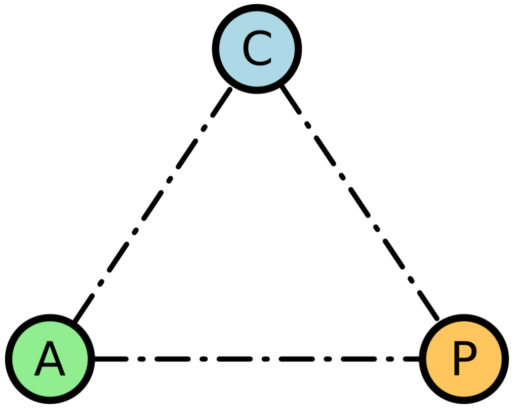


CAP Theorem

Conjecture postulated: Eric Brewer in 2000

Formally proven: Gilbert & Lynch in 2002

http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.20.1495



- C: Consistency
- A: Availability
- P: Partition Tolerance

The CAP Theorem basically describes the relationship between Consistency, Availability and Partition Tolerance

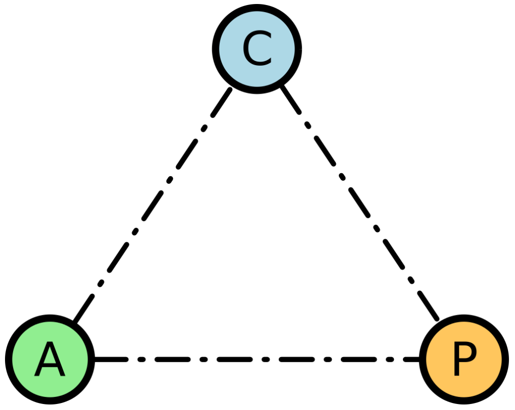
CAP Theorem

- C: Consistency

  All clients read the same data from all nodes
- A: Availability

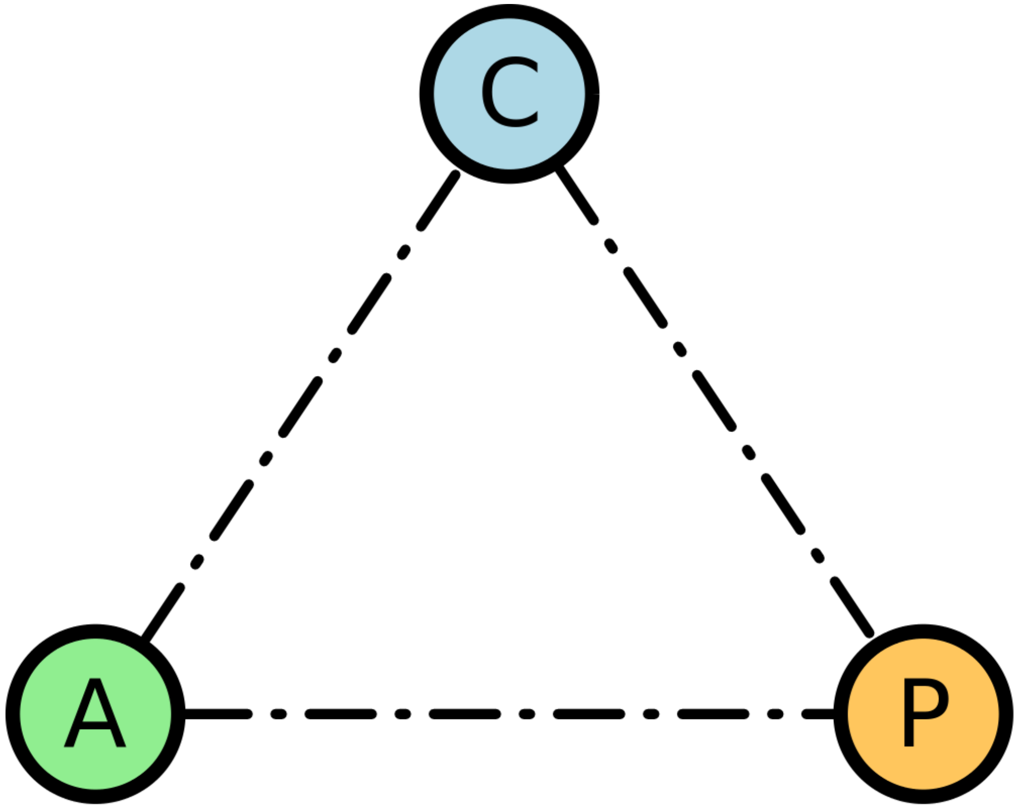
  A node, that is not faulty, always responds to queries
- P: Partition Tolerance

  A cluster is tolerant to communication failures



Consistency means, that all clients read the same data from all nodes, hence, the meaning is different from the C in ACID. Availability is given, when a non-faulty node always responds to a query. Finally, a "Partition Tolerant" cluster has some way to deal with communication problems.

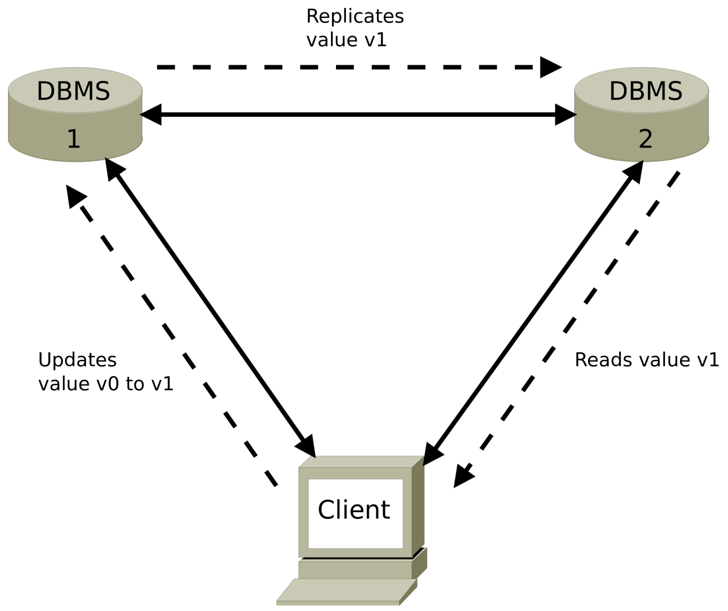
CAP Theorem



Central message: Only two characteristics can be fulfilled.

The central message of the CAP theorem is, that only two characteristics of the three can be fulfilled, so one must choose one of the three hides of the triangle.

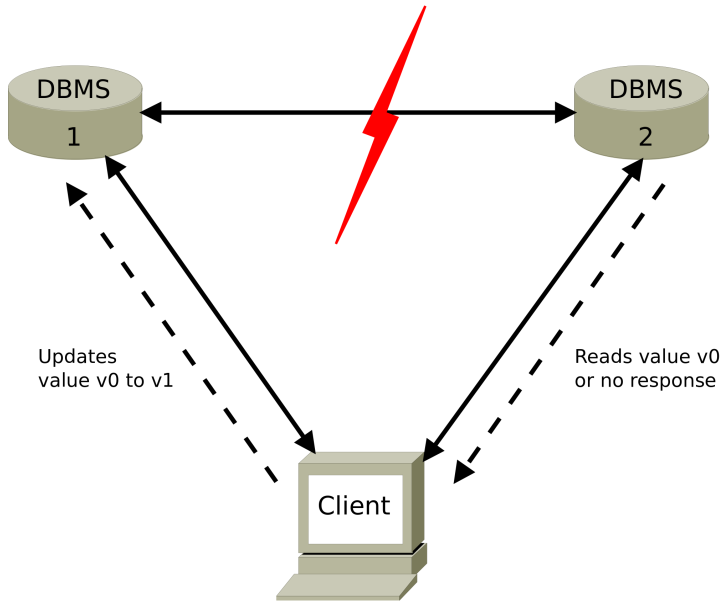
CAP Theorem



Normal operations: Client updates v0 to v1. New value is replicated across the cluster. Clients read v1 from all nodes.

In normal operations,

CAP Theorem



Communication failure: Client updates v0 to v1. Value is requested from another node, which either responds with an old value or denies the query.

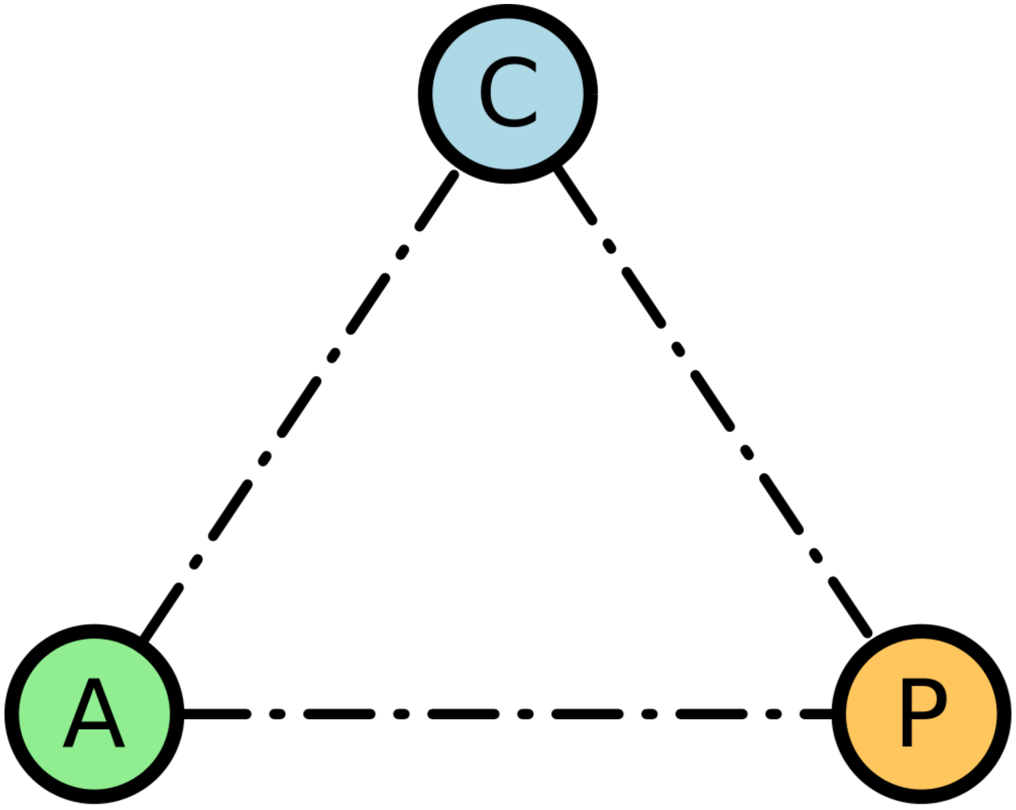
In case of a communication failure, some scenarious are feasible. Either the update itself cannot take place because the replication is not working, or DBMS 2 has to decide: Respond with a possibly outdated value or not at all.

CAP Theorem

CP: Consistent & Partition tolerant

AP: Available & Partition tolerant

CA: Consistent & Available



A system must choose between being consistent and partition tolerant or available and partition tolerant. In a distributed system, it is not very meaningful to rule out partition tolerance, so this option is grayed out.

CAP Theorem

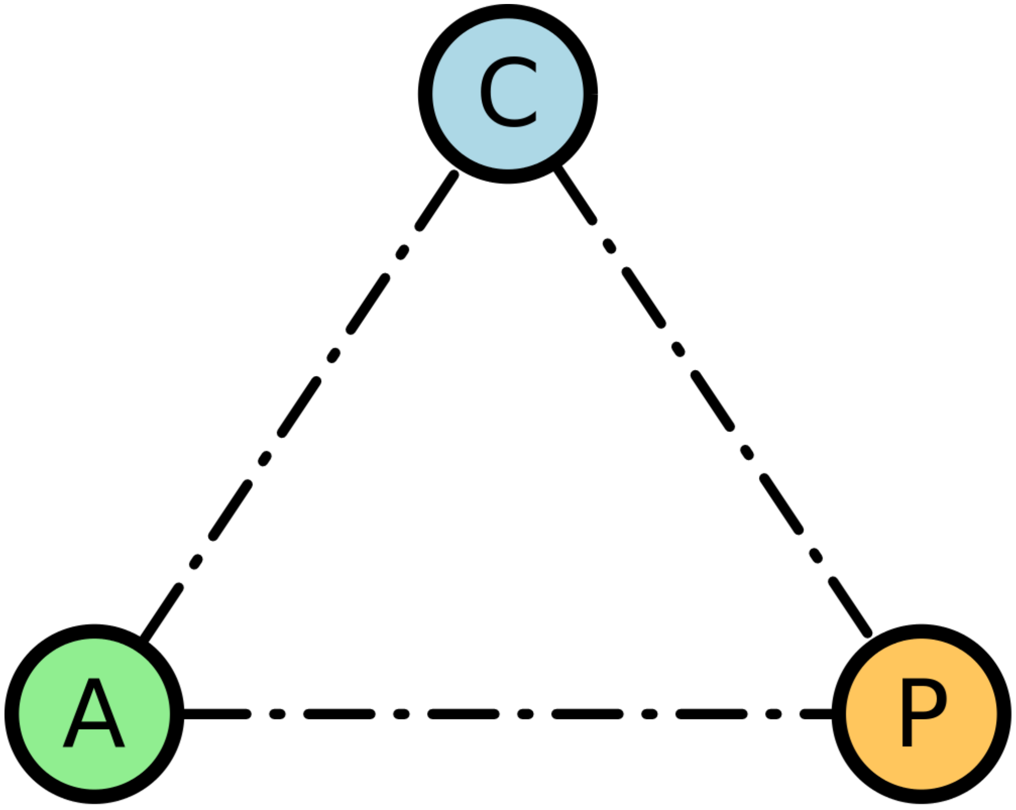
CP: Consistent & Partition tolerant

- MongoDB
- Hadoop

AP: Available & Partition tolerant

- Dynamo
- DNS (AXFR)

CA: Consistent & Available



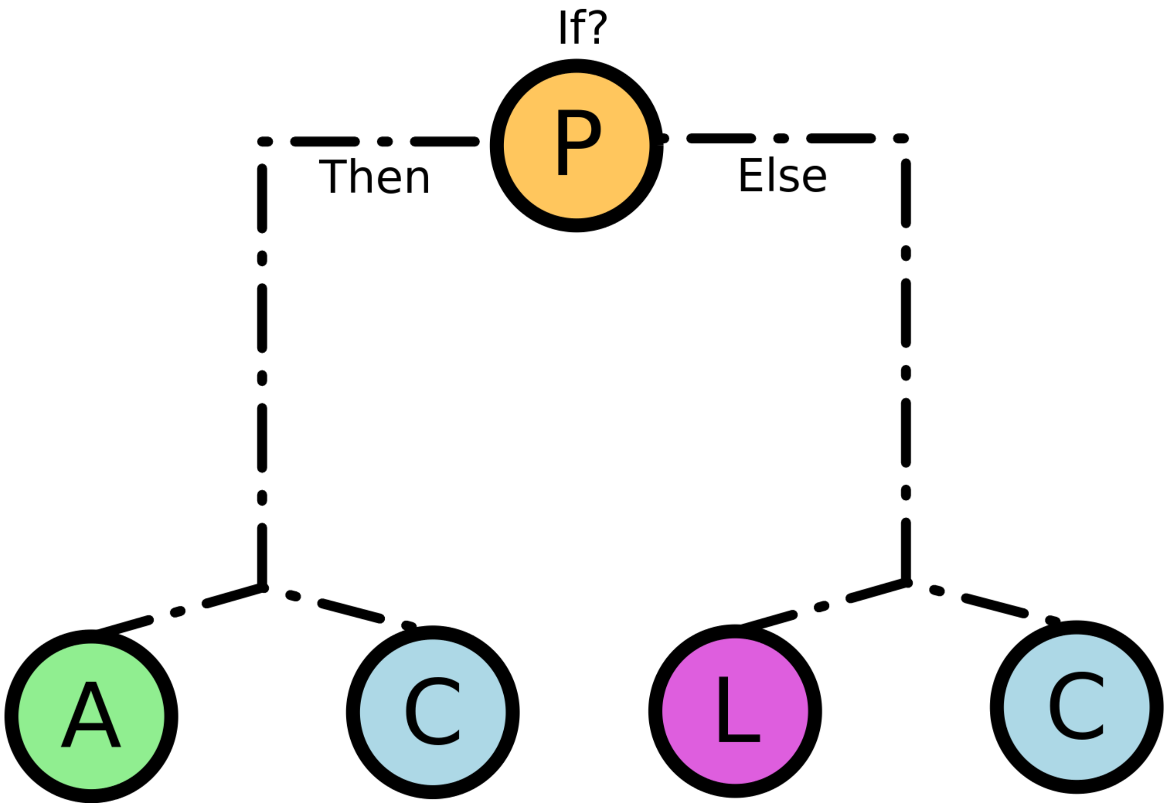
Extension: PACELC Theorem

Network **P**artitioned?

Choose between **A**vailability or **C**onsistency!

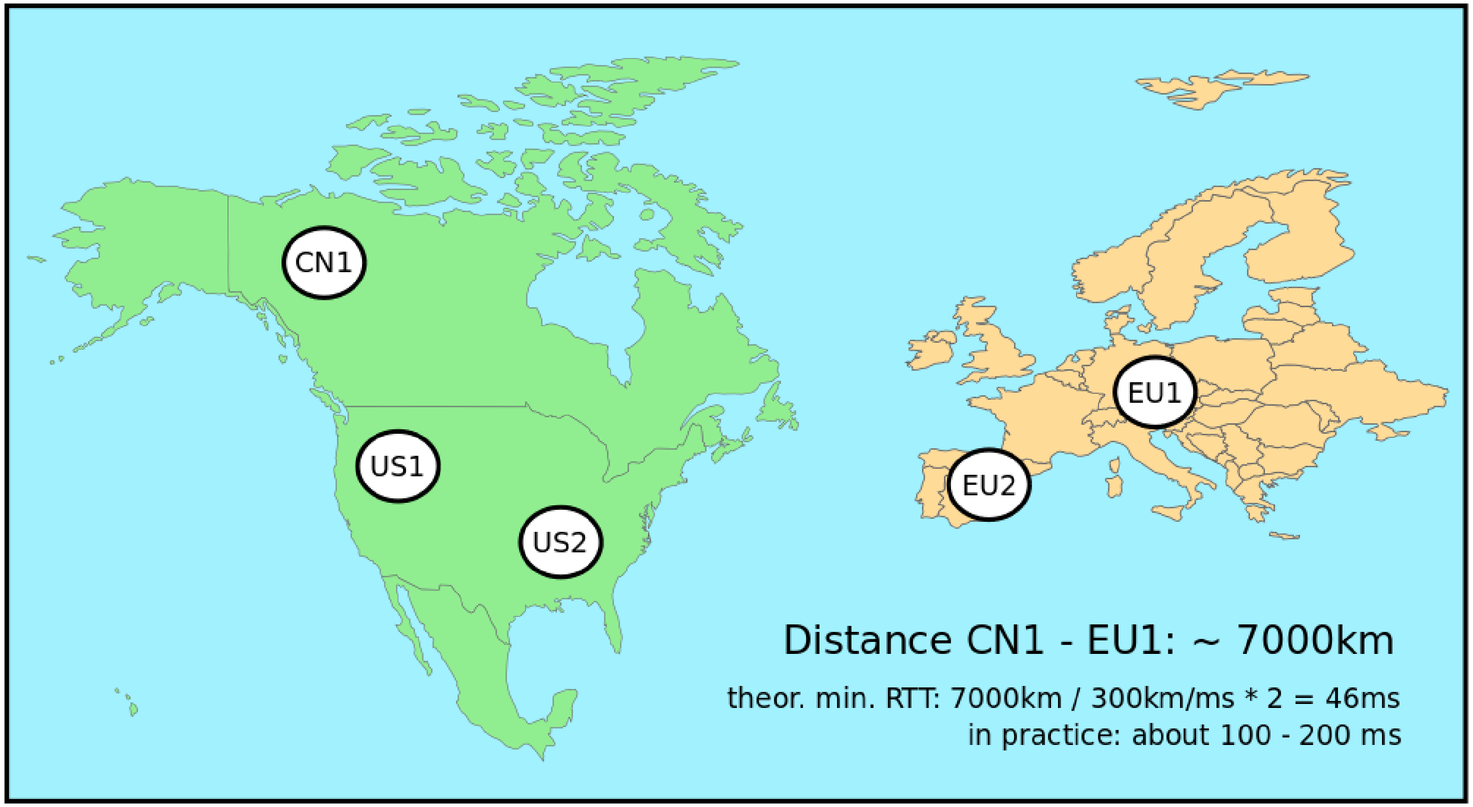
Else

Choose between **L**atency or **C**onsistency!



The PACELC theorem extends the CAP theorem by postulating, that even in case of normal network condition one has to choose between low latency and consistency in a database cluster (Trade-Off)

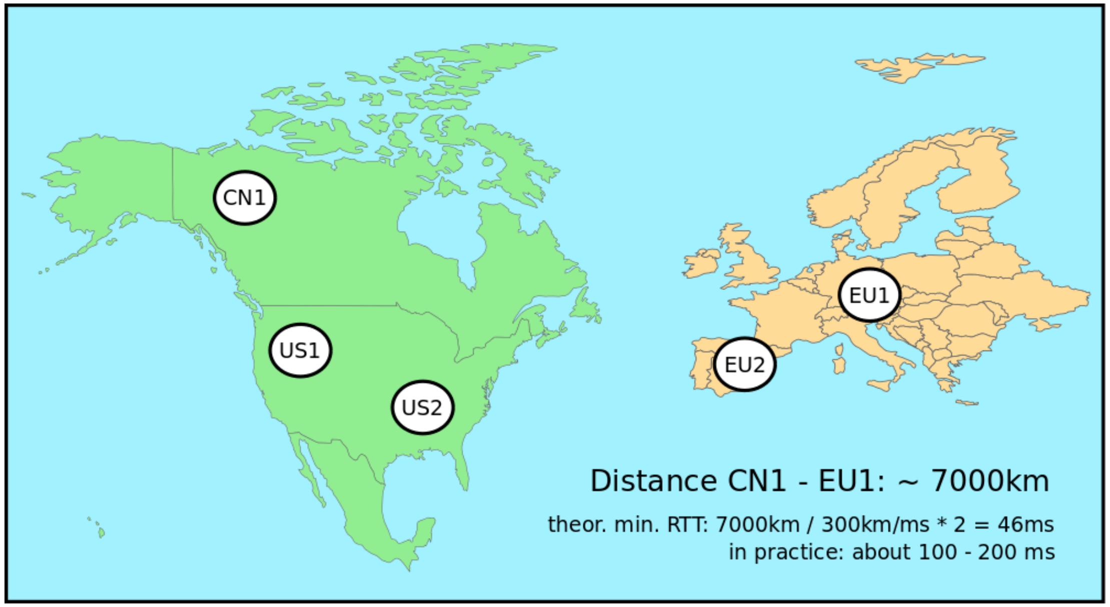
Extension: PACELC Theorem


CN1
US1
US2
EU1
EU2
Distance CN1 - EU1: ~ 7000km
theor. min. RTT: 7000km / 300km/ms * 2 = 46ms
in practice: about 100 - 200 ms

P(AC)/EC: High (write or read) latency (ca 100ms), e.g. with sync. replication

P(AC)/EL: Low latency, no strong consistency, e.g. with asynchronous replication.

Example: Apache Cassandra

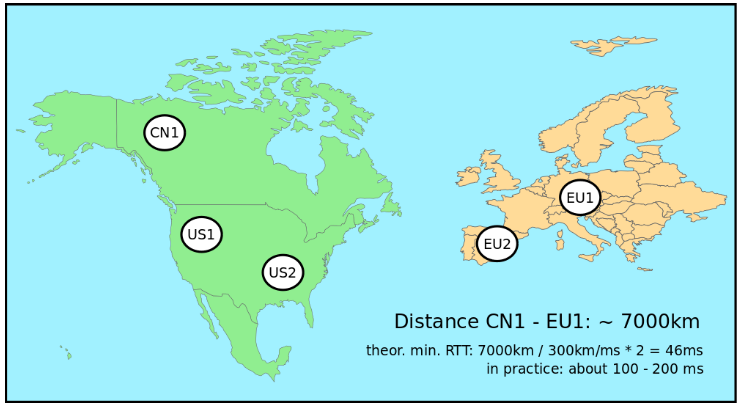

CN1
US1
US2
EU1
EU2
Distance CN1 - EU1: ~ 7000km
theor. min. RTT: 7000km / 300km/ms * 2 = 46ms
in practice: about 100 - 200 ms

All nodes have the same role

Consistency level can be set per query:

by setting the number of cluster nodes, that need to confirm a write or read request

Example: Apache Cassandra



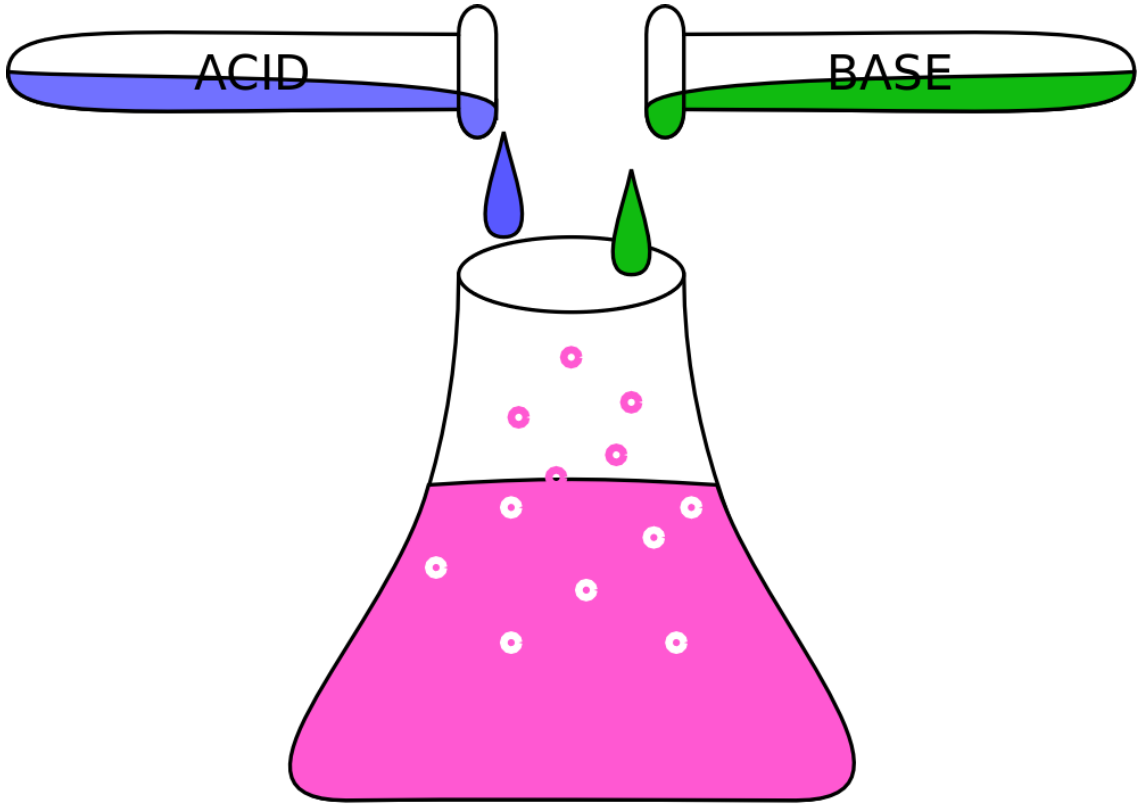
CN1
US1
US2
EU1
EU2
Distance CN1 - EU1: ~ 7000km
theor. min. RTT: 7000km / 300km/ms * 2 = 46ms
in practice: about 100 - 200 ms

Some consistency levels:

- One
- Two
- Three
- Quorum (more than half of the nodes)
- All

In general R + W > N needed for Strong Consistency

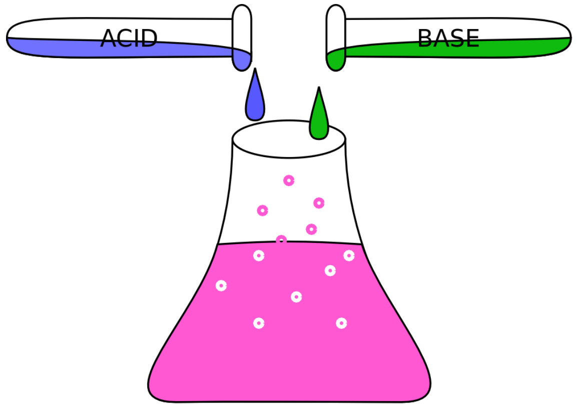
BASE Principle



The BASE principle is in contrast to the ACID transactional model

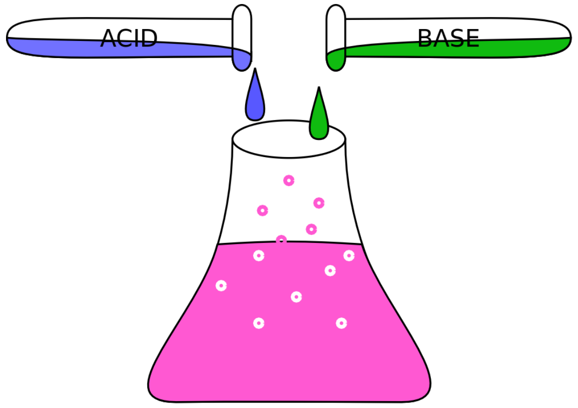
BASE Principle

- AP is hard to implement efficiently with ACID semantics
- BASE principle is a viable alternative
- Introduced by Eric Brewer
- Less strictly defined than ACID
- Best-Effort approach



It is not as strict defined as ACID and follows a best-effort approach instead of making any guarantees.

BASE Principle



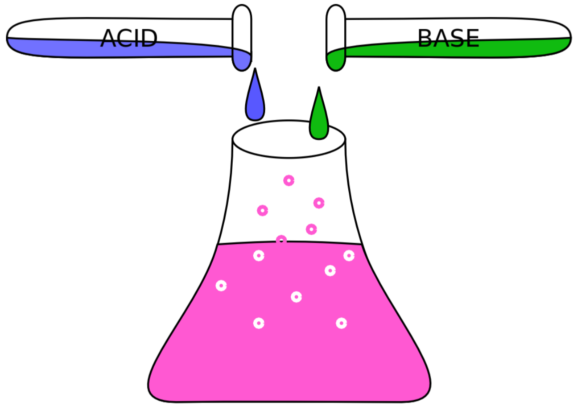
Basically

Available

Soft State

Eventually Consistent

BASE Principle



In contract to ACID,

... no hard criteria that must be met

... focus on availability, not consistency

... simpler to implement

... higher performance

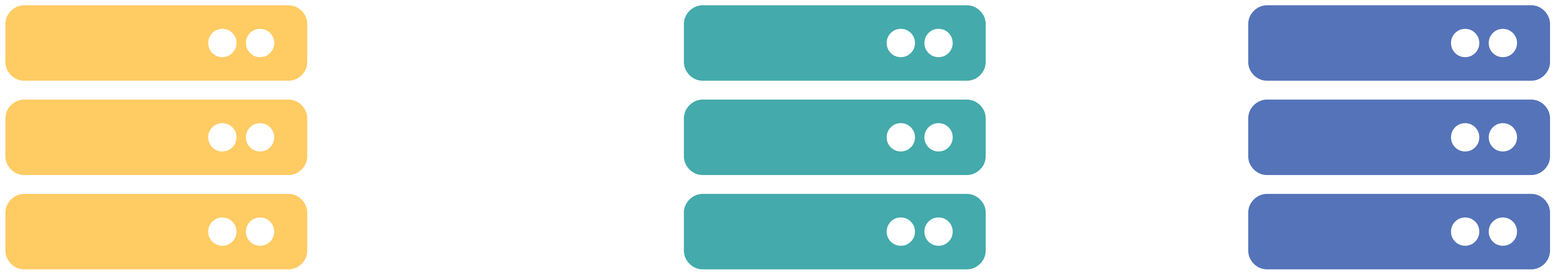
... but not suitable for all applications

Focus lies on availability and efficiency, with the drawbacks of a soft cluster state (the state may differ from node to node) and eventual consistency, which means, that consistency is not given at any time, but will be achieved at some point in the (near) future. It is often simpler to implement and performance is usually better, but it is not suitable for applications, that rely on strong consistency.

Partitioning / Sharding

Distributed storage of data across the cluster nodes

Range Partitioning



product_id ∈ [-∞, 999]　　[1000, 1999]　　[2000, ∞]

(+) supports equality (=) and range queries (<, ≥, …) on the shard key

(-) hot spots can occur

The difference between sharding and replication is that sharding does not duplicate the data. Here, data is divided and distributed across the cluster nodes. In this picture, each server stores a third of the data (assuming an equal distribution). A hot spot is a key or a range of keys which are much more often accessed than others. When data is partitioned by date or timestamp ranges, this is often the case because typically, most queries access the current data. Less often, historical data is accessed.

Partitioning / Sharding

Distributed storage of data across the cluster nodes

Hash Partitioning / Consistent Hashing

$$h(k) = \dots \; mod \; 2^{32}$$

h(MAC_ADDR) = 858 993 459 2 576 980 378 3 435 973 827

A primitive hash-partitioning approach would be choosing $h(k) = k \; mod \; n$ as a hash function, with n being the number of nodes in the cluster (in our example: 3). This would lead to an immense effort in repartitioning when a node is added because the hash function needs to be adjusted. Consistent hashing is an approach which uses the same hash function even when the cluster changes. With this hash function, the home node of each data item can be computed. Different from range partitioning, this approach does not support range queries on the partition key.

Summary

- Big Data
- 4 Vs: Volume, Velocity, Variety, Veracity
- Scale up / Scale out
- Replication: Master-Slave, (a)synchronous Repl.
- Strong consistency / Eventual consistency
- CAP / PACELC Theorem
- Sharding: Range Partitioning, Hash Partitioning