Database CAPabilities

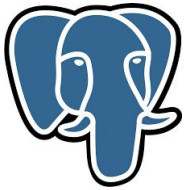
How to choose the right database for your service?

Tore Sæstad | September 2023

Agenda

- From single to a distributed database
- Replication factor
- Consistency level
- Versions and deletes
- The CAP theorem
- The PACELC theorem

RDS



PostgreSQL

Redis

DynamoDB

MySQL

Elasticsearch

MariaDB

S3

SQLite

Timestream

Cassandra

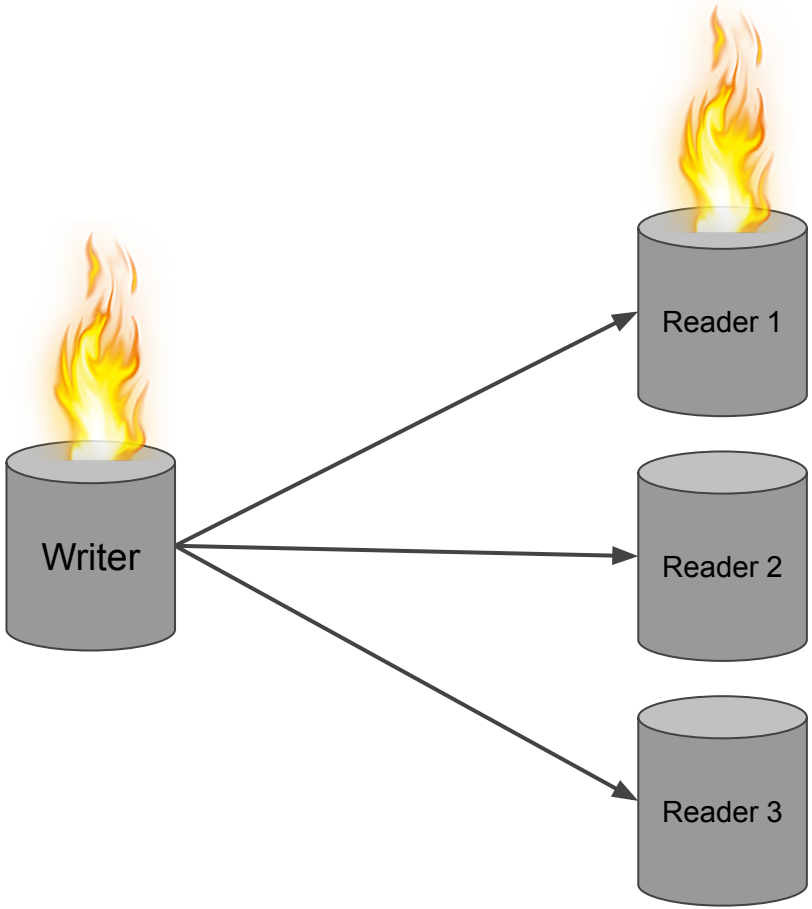
MongoDB

It all started out quite simple

- Consistent data
- No overhead
- How to scale it?
- How to handle upgrades?
- What if it fails?

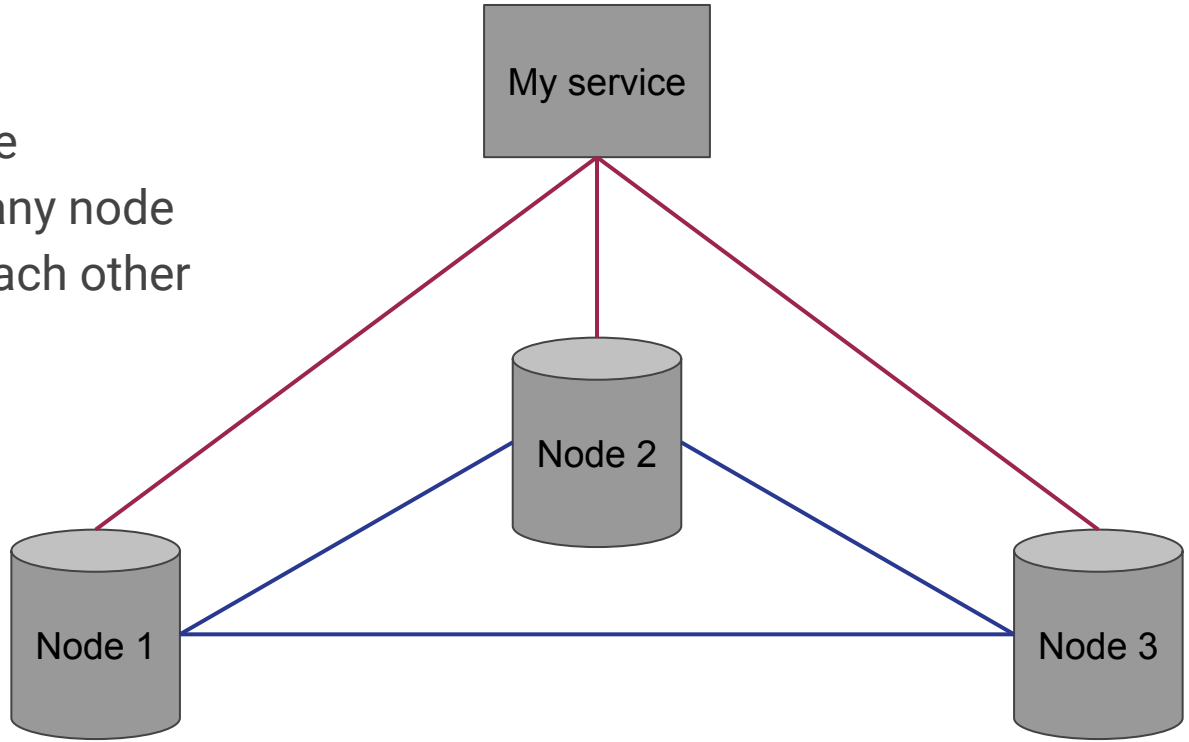
Replication to the rescue!

- One writer, multiple readers
- More readers → more read capacity
- Can still read when writer is down
- Synchronous vs. asynchronous
- What is our strategy?



Let’s build a distributed database!

- All nodes are equal
- No single point of failure
- Our service can talk to any node
- The nodes can talk to each other

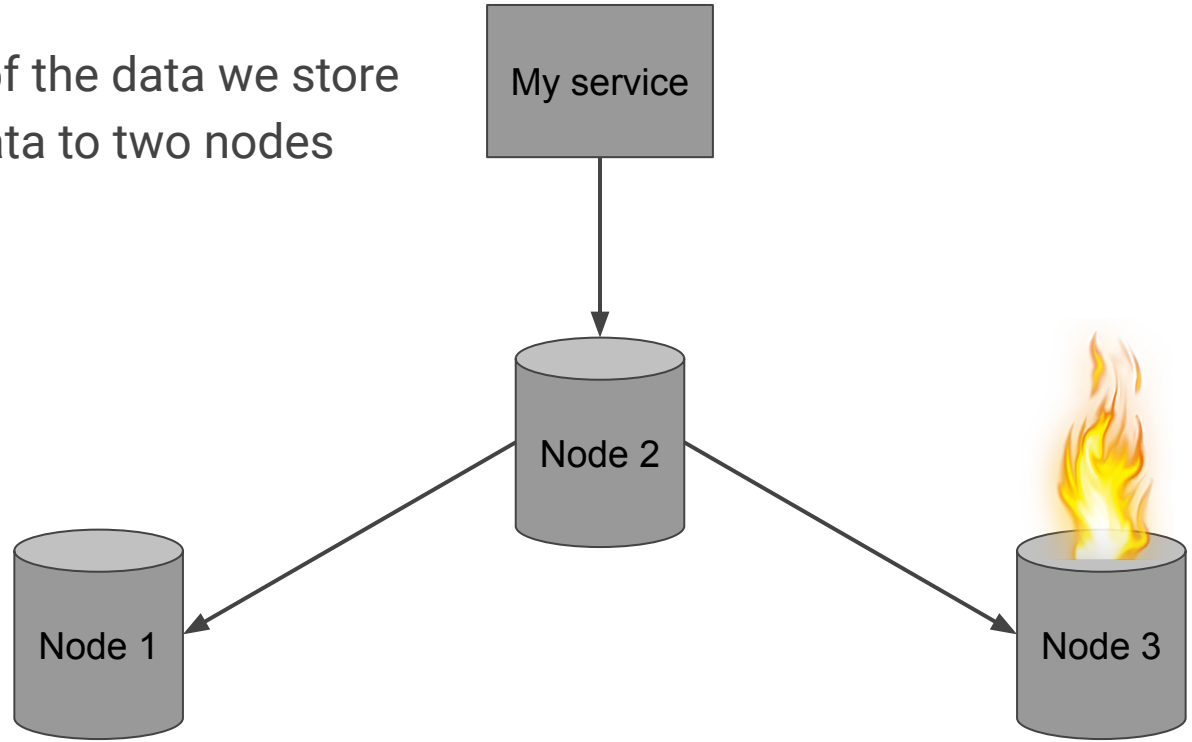


Why not just two nodes?

- Cost, can’t consume more than 50% of the resources
- Updates/failures
- To achieve a quorum, avoids split brain situations

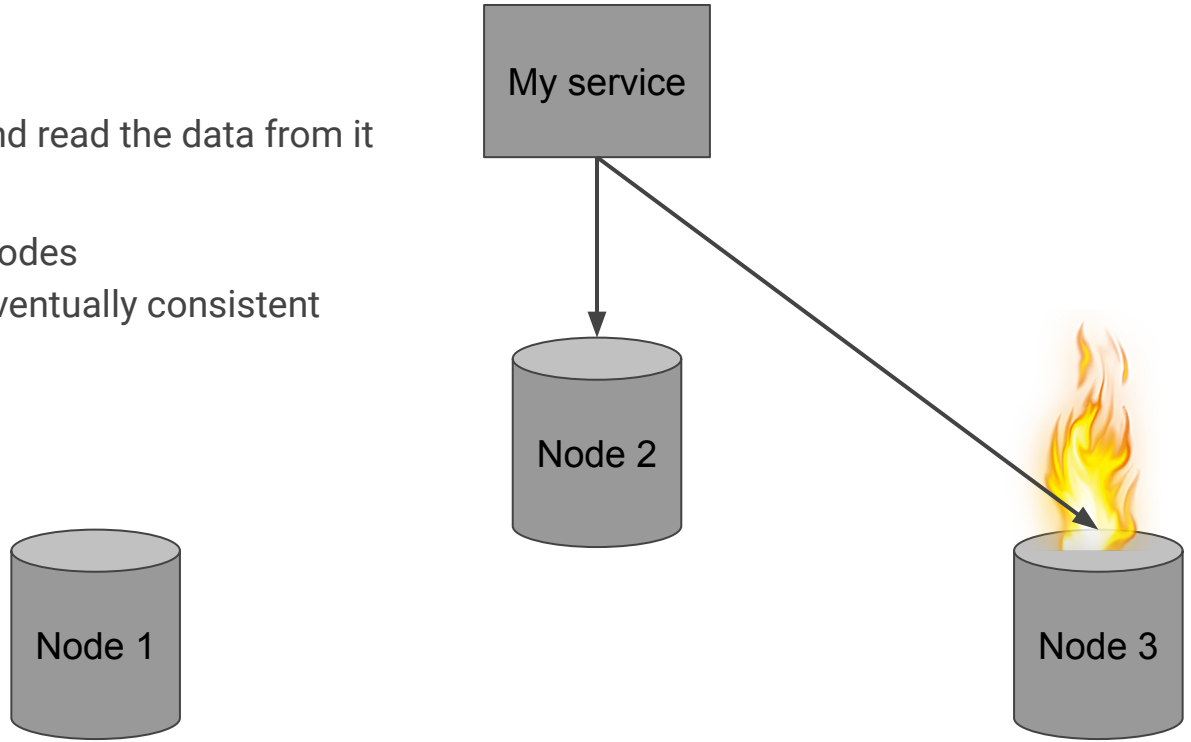
Replication factor

- The number of copies of the data we store
- DynamoDB will store data to two nodes before returning OK



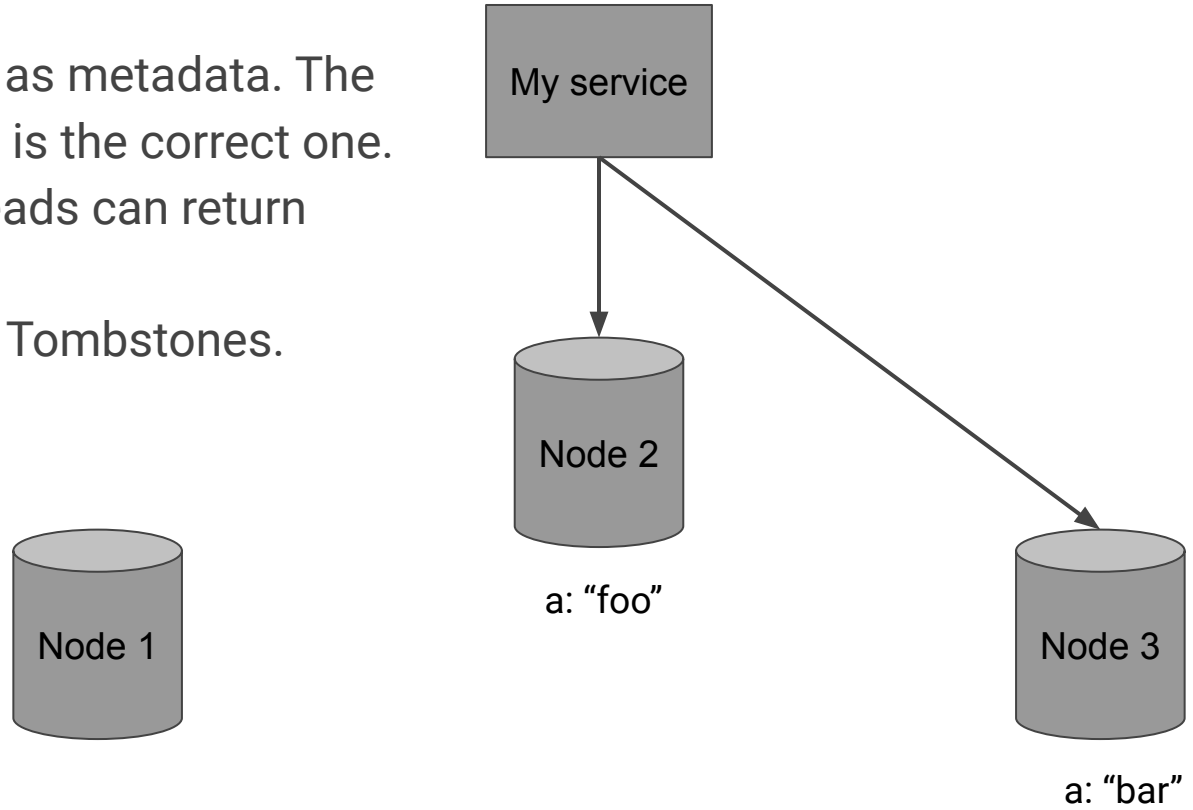
Consistency level

- Eventually consistency
  - Randomly pick a node and read the data from it
- Strong consistency
  - Read from at least two nodes
  - Twice as expensive as eventually consistent reads for DynamoDB



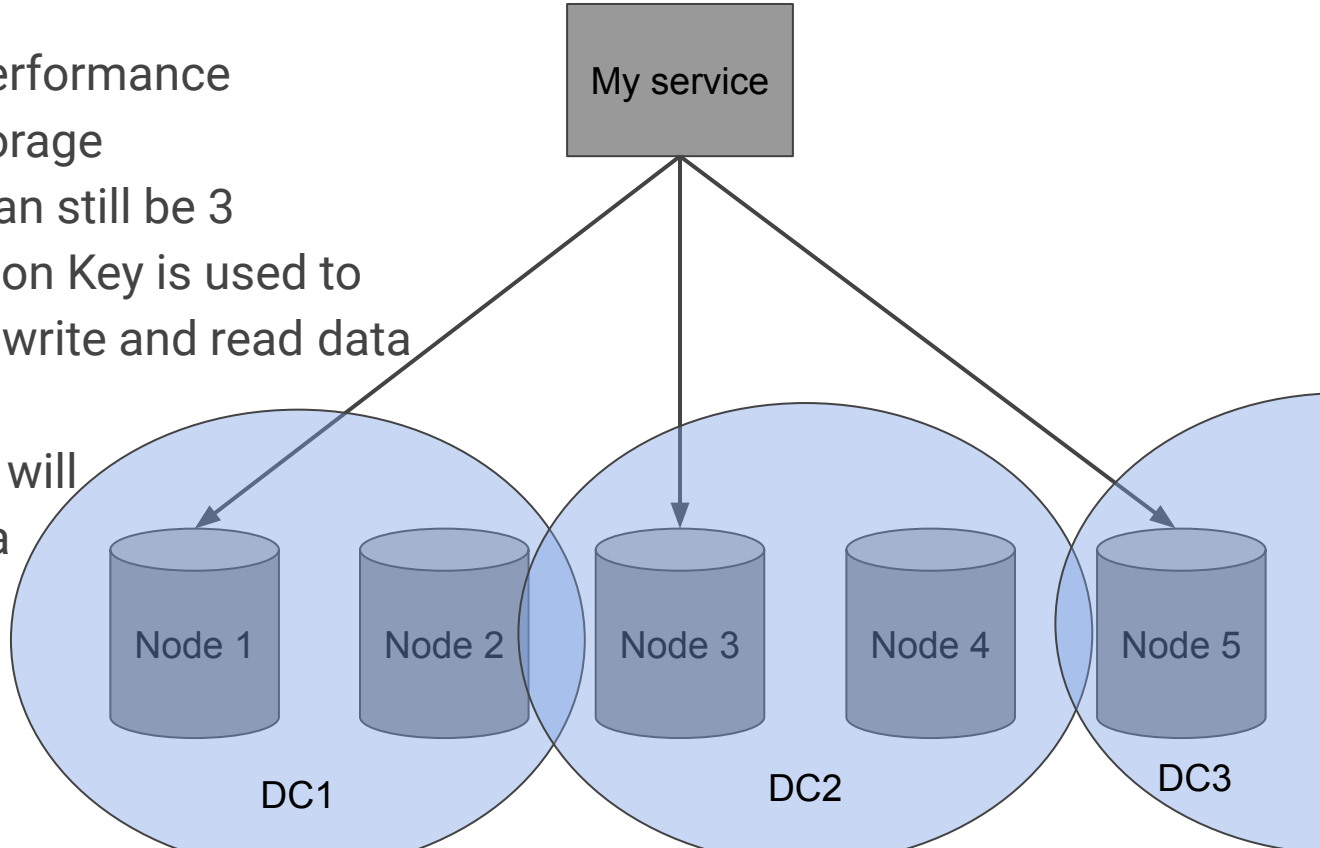
Which version is correct?

- Timestamps are stored as metadata. The most up to date version is the correct one.
- Eventually consistent reads can return stale (or no) data.
- How to handle deletes? Tombstones.



Scaling our distributed database

- Scaling for better performance
- Scaling for more storage
- Replication factor can still be 3
- A hash of the Partition Key is used to determine where to write and read data (sharding)
- Adding more nodes will redistribute our data

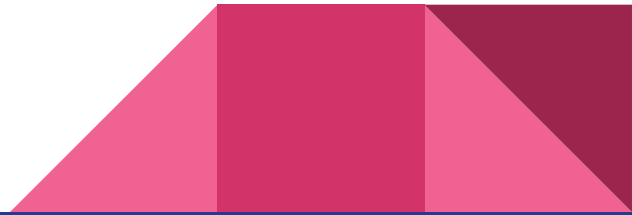


The CAP theorem

“Any distributed data store can only provide two of the three guarantees:

- Consistency
  - Every read receives the most recent write or an error.
- Availability
  - Every request receives a (non-error) response, without the guarantee that it contains the most recent write.
- Partition tolerance
  - The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.”

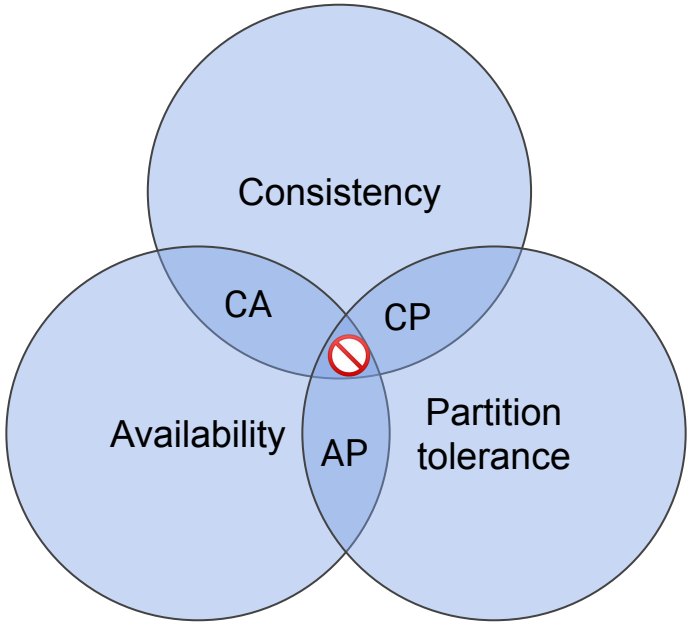
https://en.wikipedia.org/wiki/CAP_theorem



The CAP theorem (by Eric Brewer in 1998)

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

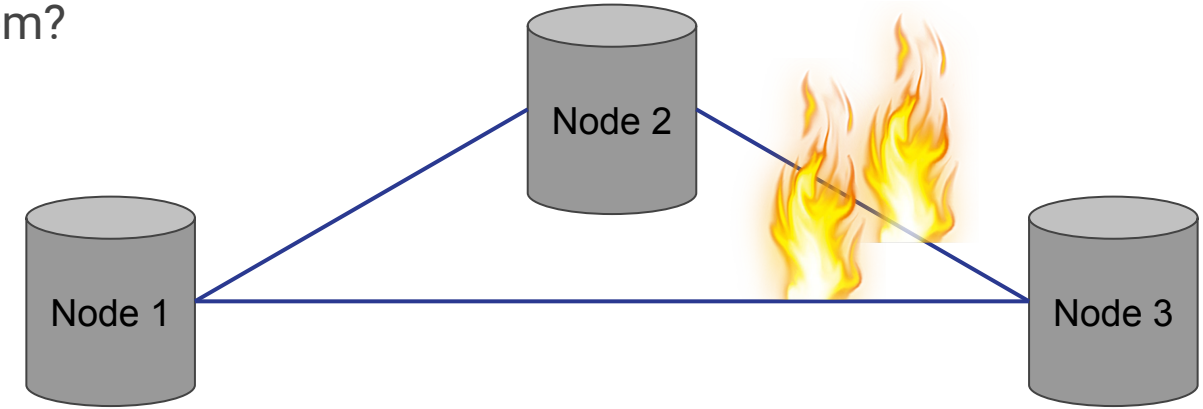
- cancel the operation and thus decrease the availability but ensure consistency
- proceed with the operation and thus provide availability but risk inconsistency.”



https://en.wikipedia.org/wiki/CAP_theorem

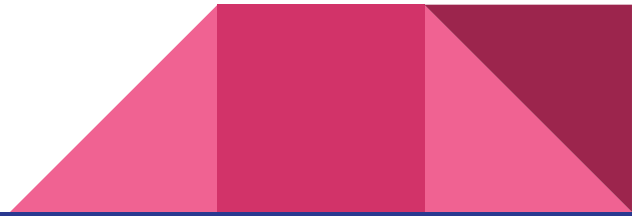
Network partition example

- CA (consistency and availability)
- CP (consistency and partition tolerance)
- AP (availability and partition tolerance)
- The choice is consistency vs. availability
- Any use cases for them?



CAP theorem for popular databases

- Availability, Consistency
  - MySQL, PostgreSQL, SQLite
- Consistency, Partition tolerance
  - MongoDB, Redis
- Availability, Partition tolerance
  - DynamoDB, Cassandra



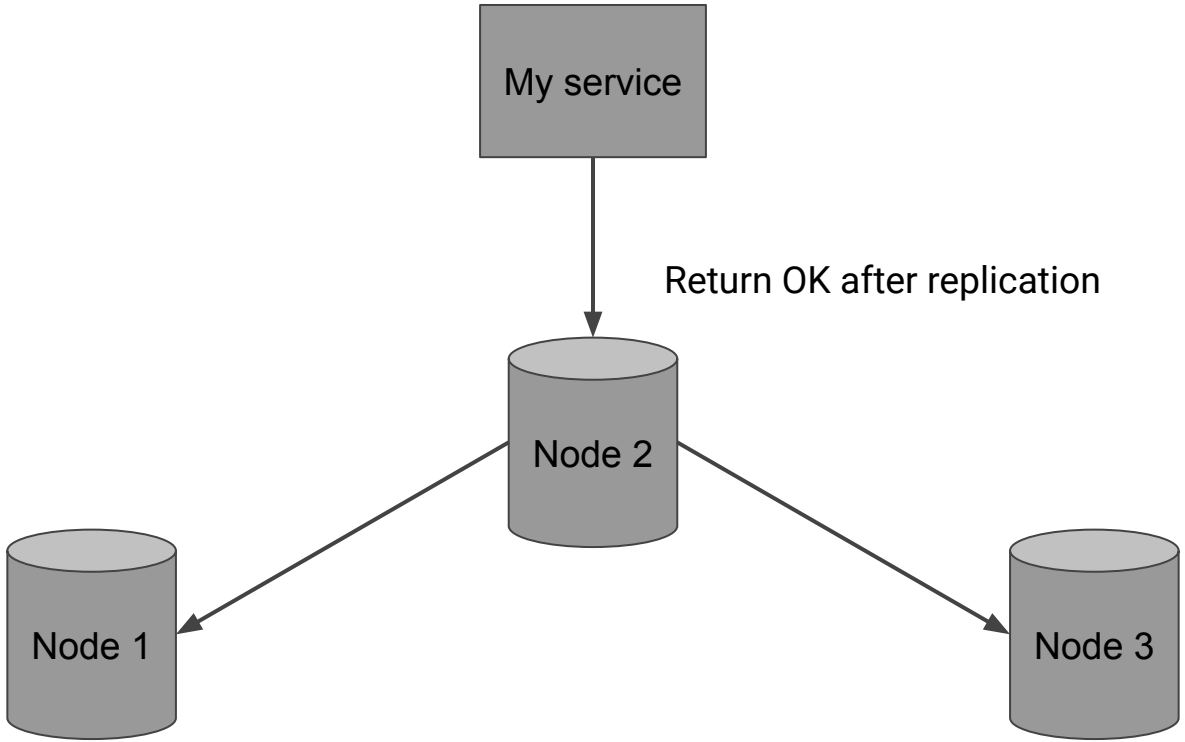
PACELC

- There is another trade-off for distributed data stores.
- “In case of network partitioning (**P**) in a distributed computer system, one has to choose between availability (**A**) and consistency (**C**) (as per the CAP theorem), but else (**E**), even when the system is running normally in the absence of partitions, one has to choose between latency (**L**) and consistency (**C**).”

https://en.wikipedia.org/wiki/PACELC_theorem

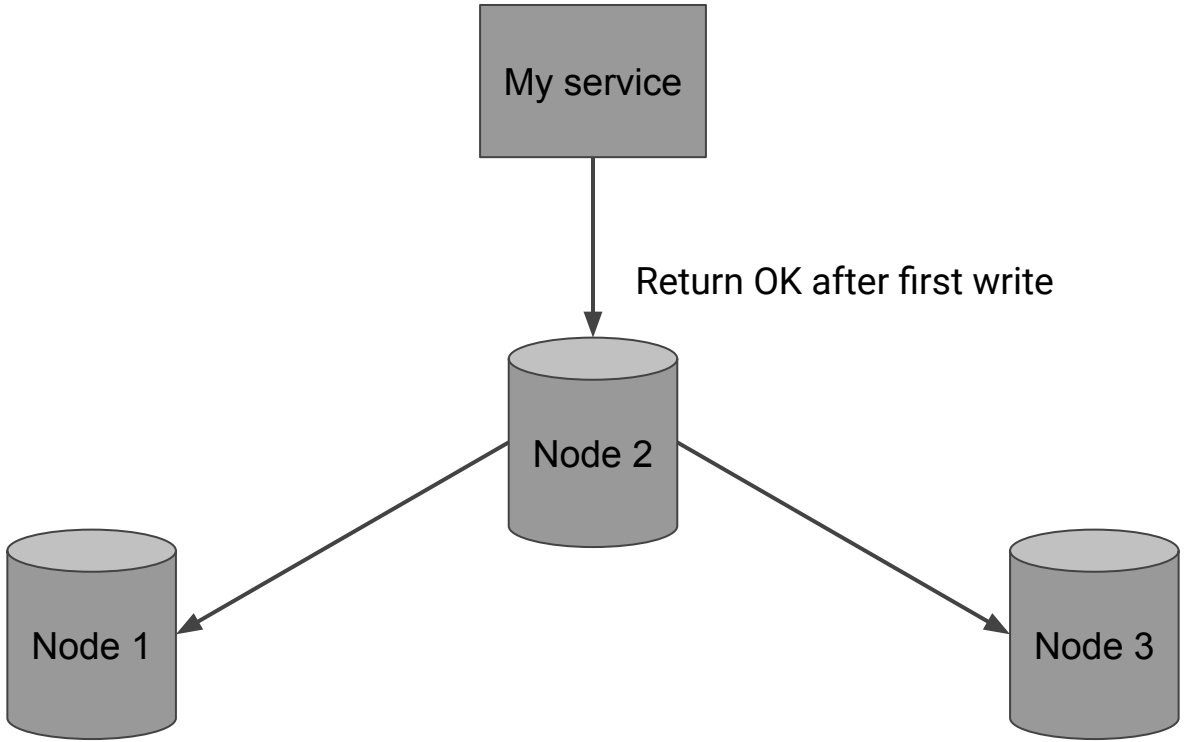
Consistency

- Strong consistency
- Longer latency



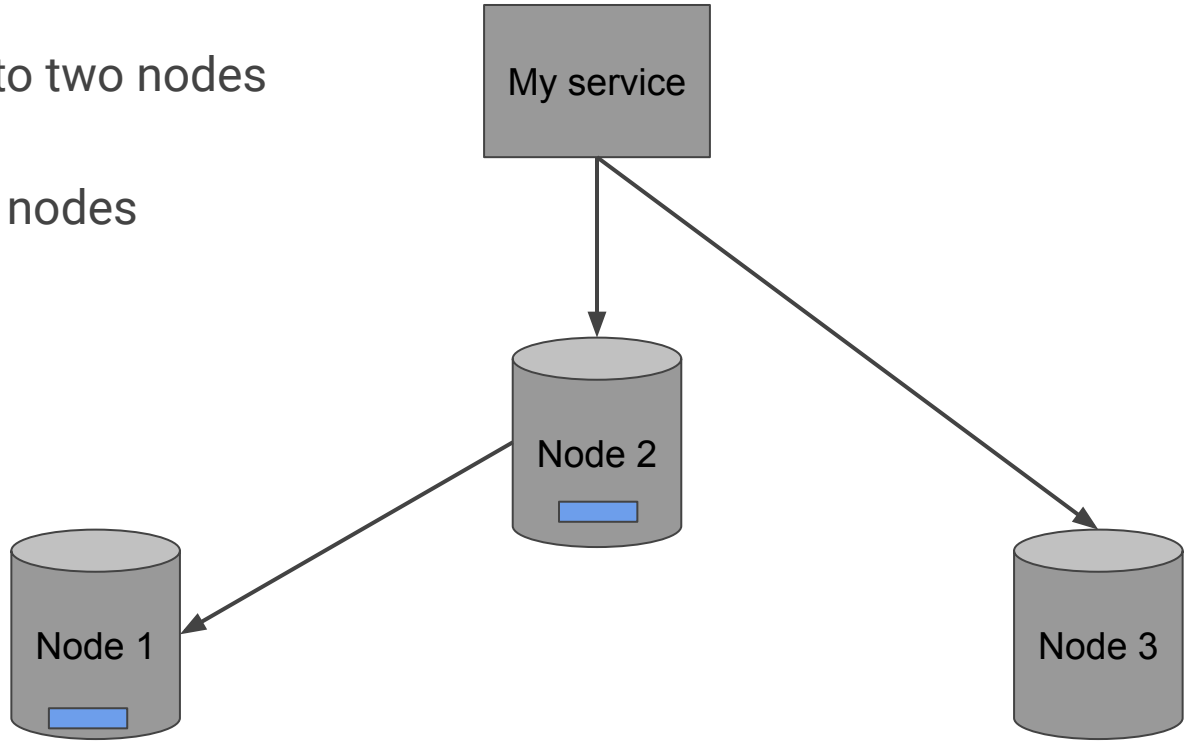
Latency

- Low latency
- Eventually consistent



Best of both worlds?

- Return OK after writing to two nodes (quorum)
- Read the data from two nodes (quorum)
- Medium latency
- Strong consistency
- DynamoDB example



Summary

- From single to a distributed database
- Replication factor
- Consistency level
- Versions and deletes
- The CAP theorem
- The PACELC theorem