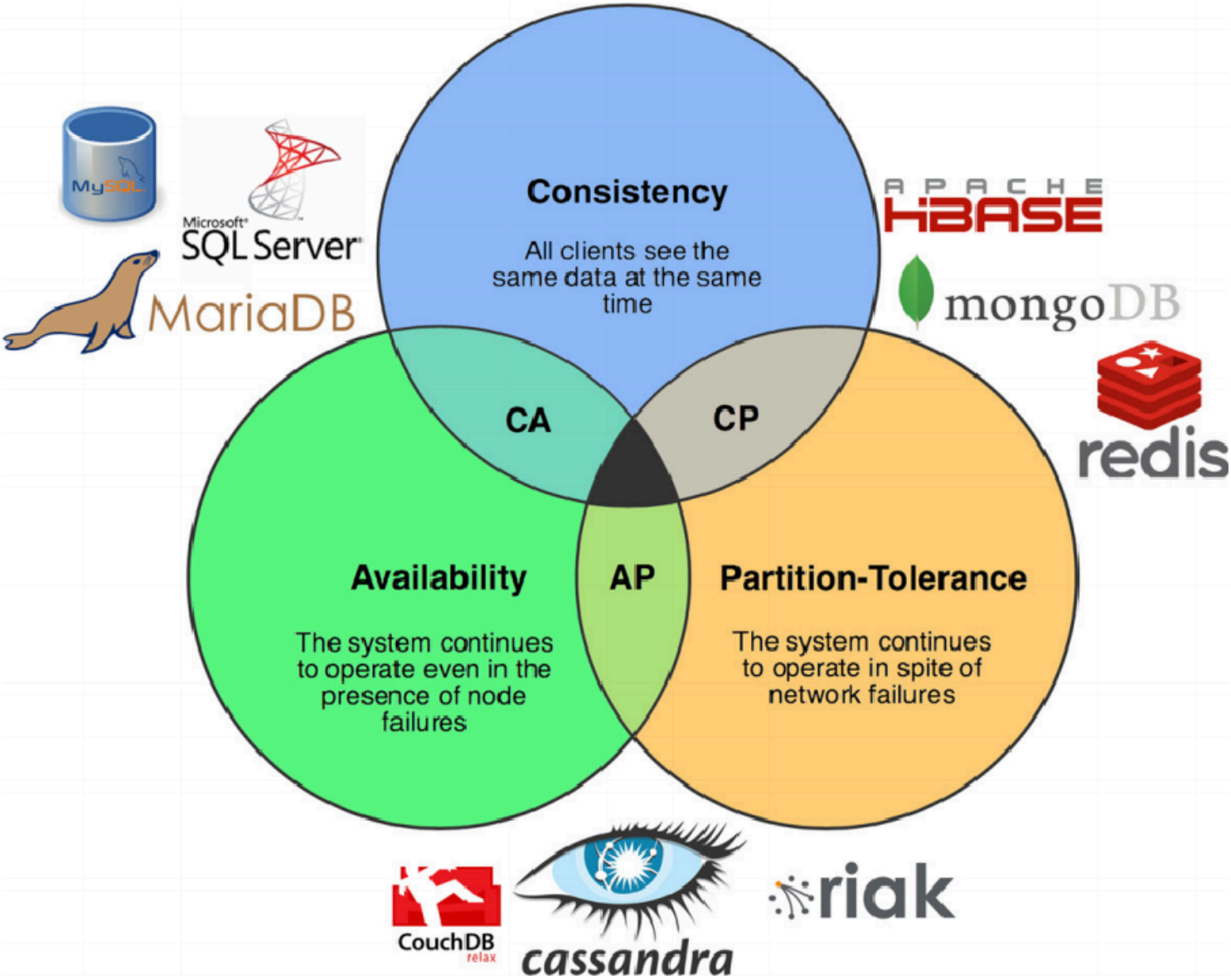
Introduction

A distributed system is a network that stores data on more than one node. The nodes can be either physical or virtual and the characteristics of the selected DBMS systems can vary based on their requirements. Today I'm going to explain the CAP Theorem and its relevance in choosing a database; either SQL or NoSQL. Later, I will explain how the CAP Theorem is applied in MongoDB. So let's get to it.

What is CAP Theorem?

Coined by Eric Brewer in 2000, CAP Theorem introduces the standard way of maintaining a distributed system. **CAP** is an acronym for **C**onsistency, **A**vailability, and **P**artition tolerance. According to the CAP Theorem, a distributes system can only have two out of the three characteristics mentioned previously.



Consistency means, when we consider a single time frame, every client should read the same data. To achieve this, when a certain client update info, the updates should be instantly forwarded to all the nodes. Other than that, partially corrupted transactions won't save. This means, if a client couldn't complete a transaction successfully, it won't be saved and will be rolled-back.

Availability means the system must remain operational all the time. Even if some nodes are down, the rest of the nodes must respond to the requests in a reasonable amount of time.

Partition tolerance means, when a part of the network gets compromised and some nodes become unreachable, the whole system must continue its work properly and should take necessary measurements to recover as quickly as possible.

CA, CP or AP?

As I mentioned previously, according to CAP Theorem a distributed system can only have two out of three characteristics; consistency, availability, partition tolerance.

CA Database (Consistency and Availability): These systems focus on consistency and availability. Clients can see the latest data while the availability of the data is very high. But the entire system can go down if a part of the system gets compromised.

Example: RDBMS (MySQL, SQL Server)

CP Database (Consistency and Partition tolerance): These systems focus on consistency and partition tolerance. If a node or some nodes go down the requests to the downed node(s) won't get responded.

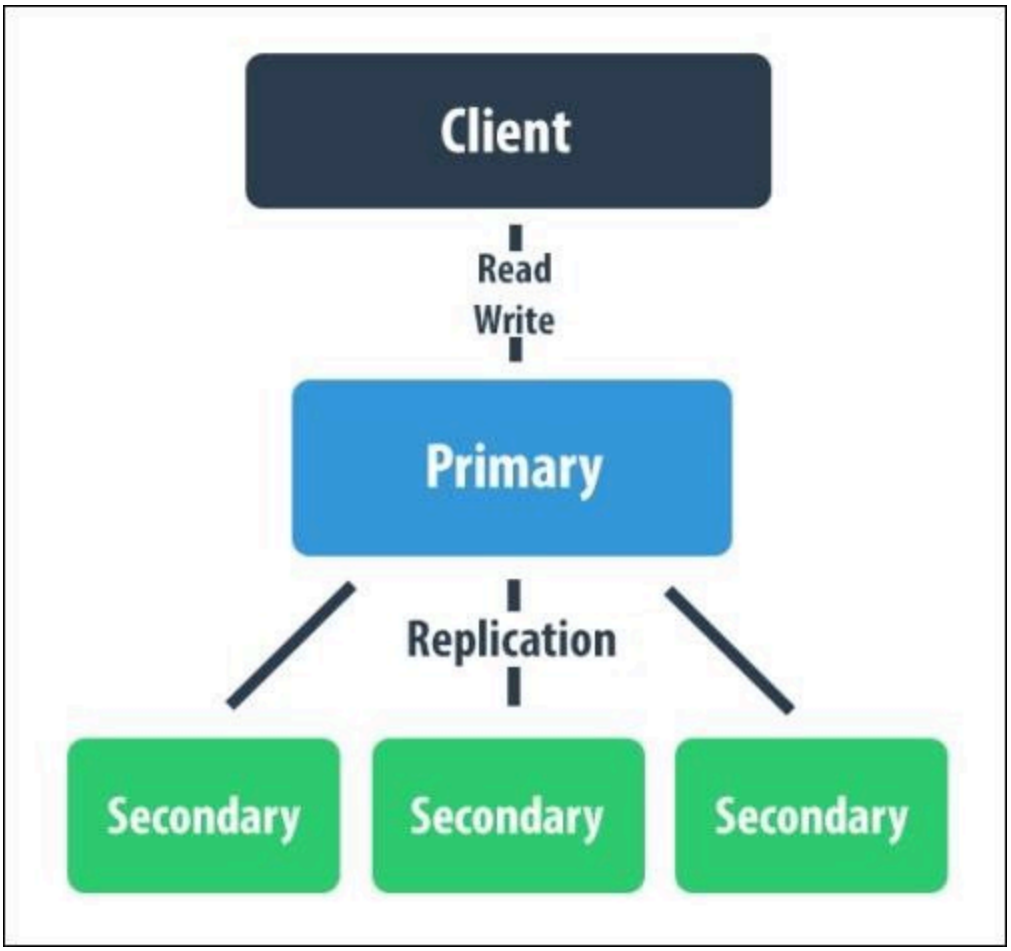
Example: MongoDB

AP Database (Availability and Partition tolerance): These systems focus on availability and partition tolerance. Every request will get responded and the system won't go down even if there are network outrages. But some clients may be exposed to older versions of data.

Example: Cassandra

CAP Theorem and MongoDB

MongoDB is one of the most popular document based NoSQL databases. It's very much used for real-time applications, which are running at different locations. According to the CAP Theorem MongoDB is a CP database, which means the data is consistent across the entire system and also focusses on partition tolerance.

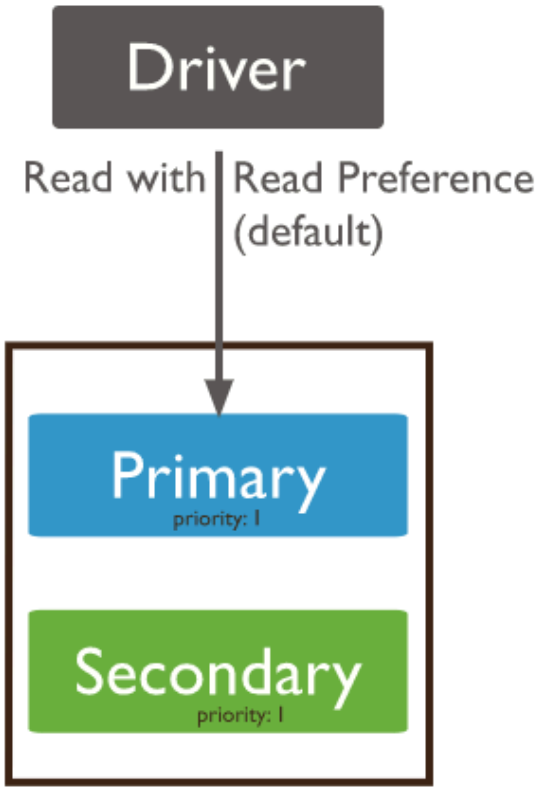


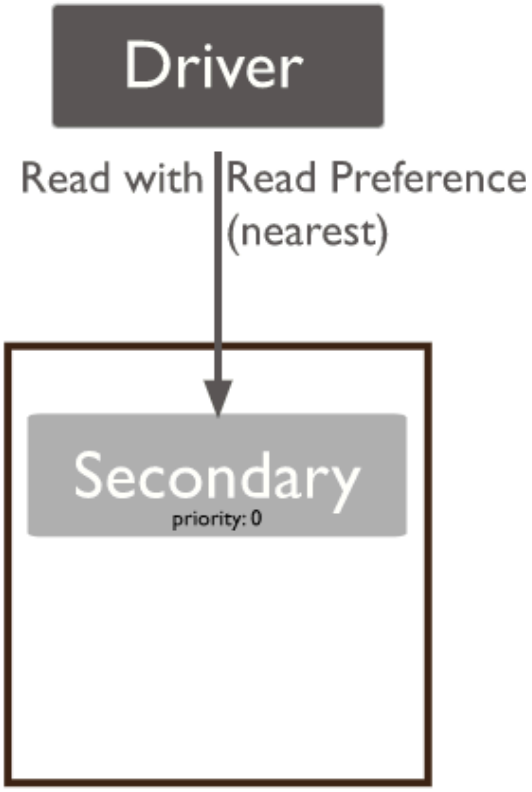
MongoDB can have multiple replicas but a single primary node. If a primary node goes down, a new node will be elected as the primary node, which means these replica sets in MongoDB helps to achieve the partition tolerance.

> "A replica set is a group of `mongod` instances that maintain the same data set. A replica set contains several data bearing nodes and optionally one arbiter node. Of the data bearing nodes, one and only one member is deemed the primary node, while the other nodes are deemed secondary nodes."

Reference: ***https://docs.mongodb.com/v3.4/replication/ (https://docs.mongodb.com/v3.4/replication/)***
Consider these situations, If a primary node is disconnected from the cluster or a client is disconnected from the primary node. In default configurations, the MongoDB driver sends all the requests (both read and write) only to the primary node, so both of the above situations cause unavailability.

But if someone wants, there is an option called "read-preference" which allows clients to directly READ (not write) from a secondary node. But the secondary nodes may not have the latest data, so there will be a lack of consistency.





To solve this "write-concerns" can be used. By using this clients can write to multiple nodes, which will help to increase the data consistency.

> "Write concern describes the level of acknowledgment requested from MongoDB for write operations to a standalone `mongod` or to replica sets or to sharded clusters."
>
> *Reference:* ***https://docs.mongodb.com/manual/reference/write-concern/ (https://docs.mongodb.com/manual/reference/write-concern/)***

Even though you can use some workarounds to increase the availability of read data, you can't make write data available. So when a new primary node being elected and the client driver is disconnected from the primary node, there will always be a downtime.

Conclusion

So now you have an idea about the CAP Theorem and how it is applied in MongoDB. Hope my article was clear and you could understand the explanations.

Thank you for reading. Feel free to drop a **feedback (https://rukshanjayasekara.wordpress.com/feedback/)** 🙂

43 Claps