CAP Theorem

Big Data Systems
School of Data Science
University of Virginia

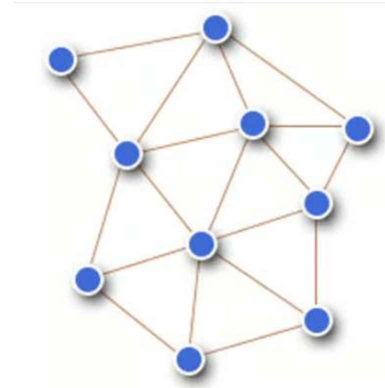
Last updated: January 20, 2026

Scaling and its Challenges

When we scale from single machine to distributed system,

it makes partitions inevitable (breaks in network)

We are faced with tradeoff explained by CAP Theorem

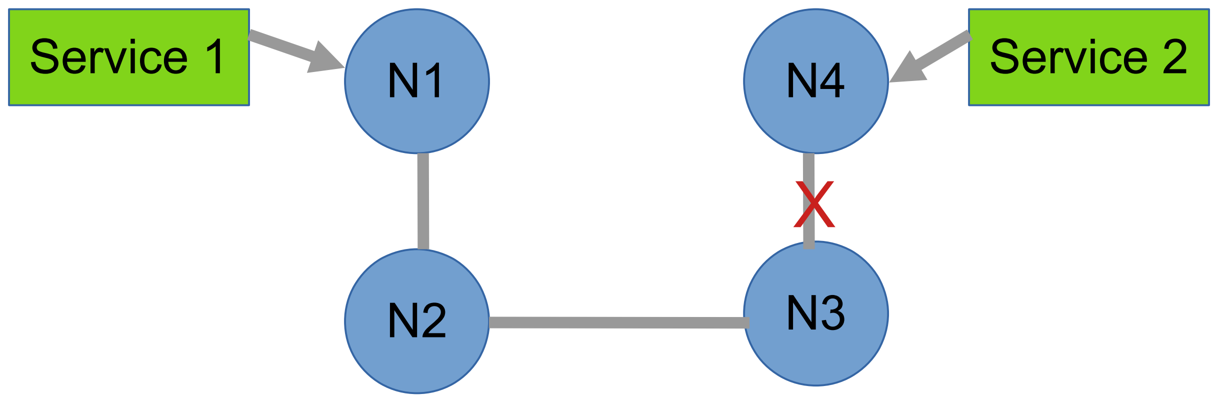


Partition Tolerance

Data processing system must continue to work even if network partition causes communication errors between subsystems

Partition Tolerance

Failure causes disconnect between N3 and N4



Request by Service 2 at N4 gets stale data

Background on DB Transaction Models

Database Transactions

Transactions are a change of state performed against db

Useful for preventing issues:

- System failure prevents some work from completing

Database Transactions

Transactions are a change of state performed against db

Useful for preventing issues:

- System failure prevents some work from completing
- Provide isolation between programs accessing db concurrently

ACID

Atomicity
All steps in db transaction fully completed or reverted to original state. Avoids partial completions.

Consistency
Guarantees that data meets predefined integrity rules

ACID

Isolation
New transaction waits until previous transaction finishes before starting

Durability
db maintains all committed records even if system fails

BASE

Basically available
Focus on concurrent accessibility by users at all times

Soft state
Data can have transient states that may change over time
e.g., several applications may update a record simultaneously

Eventually consistent
Record will be consistent when all concurrent updates complete

ACID vs. BASE

ACID databases
Prioritize strict data consistency for critical transactions

Examples: Most SQL databases e.g., MySQL, PostgreSQL, Oracle, SQL Server

ACID vs. BASE

ACID databases

Prioritize strict data consistency for critical transactions

Examples: Most SQL databases e.g., MySQL, PostgreSQL, Oracle, SQL Server

BASE databases

Prioritize scalability and availability

Suitable for large-scale, distributed systems

Examples: MongoDB, Cassandra, DynamoDB, Redis

CAP Theorem

CAP Theorem

A tradeoff faced by distributed data store

Can only guarantee two of these three:

Consistency: Each read receives most recent write or an error.
All users see same data at same time.

Availability: Each request receives response.
No guarantee it contains most recent write.

Partition tolerance: System operates despite dropped/delayed messages by network between nodes

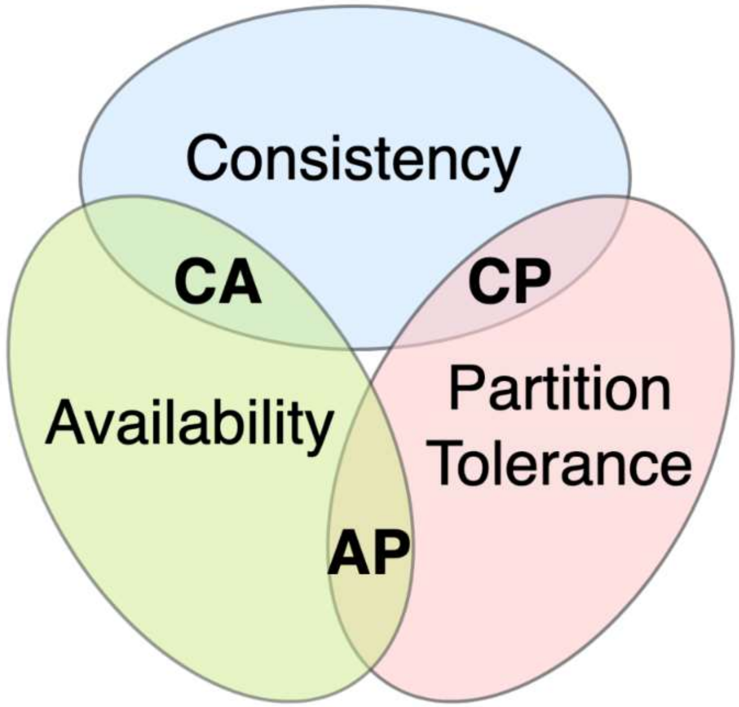
CAP Theorem, contd.

When network partition happens, must decide between:

- Cancel the operation (decrease availability, ensure consistency)

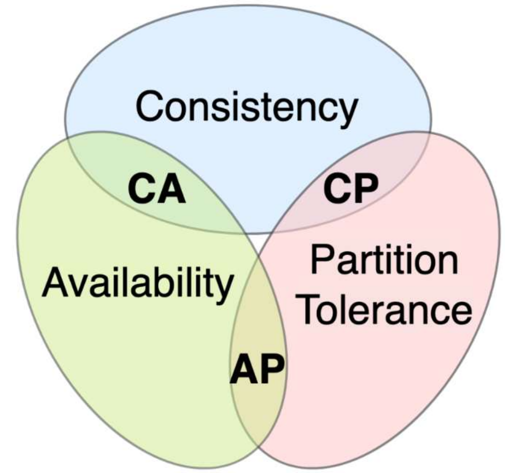
- Proceed with operation (ensure availability, risk inconsistency)

Options



Option CA: Data is not partitioned
In practice, CA distributed database cannot exist.

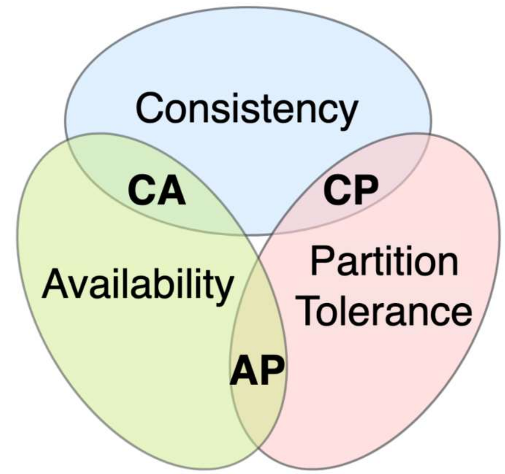
CP and AP Databases



CP: Consistency over Availability.
Traditional ACID guarantees.
Guarantee validity using transactions.
This is RDBMS like Postgres, MySQL, etc.

AP: Availability over Consistency. BASE philosophy.
BASE centers on eventual consistency.
Common in NoSQL (non-relational) database systems.

CP and AP Databases, contd.



What is required and what is the downside...

For CP?

For AP?

References

https://aws.amazon.com/compare/the-difference-between-acid-and-base-database/

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

https://www.linkedin.com/pulse/introduction-cap-theorem-rupesh-tiwari/

https://www.ibm.com/topics/cap-theorem