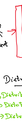
Day 4 CAP Theorem

Person Movement



→ If Person walks past Warehouse, he can do work for Warehouse
→ If Person walks past Warehouse, he finds a Warehouse

Distributed Systems:

→ Distributed DBs
→ Distributed File System
→ Distributed Messaging System

→ Instead of independent computers having own local

Case: Read > write

→ we should prevent Stale Data

Distributed Database

Read → OK
DB
Work on Single DB

Read Replicas

1/3 seconds lag, so ok in any website

Eventual Consistency

This banking account cannot lags.

This is Strong Consistency.
→ we verify with what only write synchronous across DBs. → we are synchronous to all replicas before replying write Read

Consistency → Eventual
→ Strong
→ Synchronous (consistency vs availability)
→ Asynchronous (consistency vs availability)

Eventual: Read Write Write

Strong: → R/W (Read Write with consistency)
→ If 1 command is read during write any new Write command, others can delay. This is used in social media

Consistency: Properties of a system.
→ Time same under Non functional requirement

→ Functional Requirements (FR)
→ Non Functional Requirements (NFR)
→ Consistency
→ Availability

Availability → Operational/ accessibility
→ Why does we
Datacenter cause when I stale data
Set read data

Measured in → # of uptime under a given period

SLA → Service Level Agreement
Ex website → How Reliable your system is

Two Nines → 99% uptime
365 days
→ 3.65 days of downtime per year

Three Nines → 99.9% uptime
365 days
→ 8.76 hours of downtime per year

Four Nines → 99.99% uptime
365 days
→ 52.6 minutes of downtime

Five Nines → 99.999%
365 days
5.26 minutes of downtime
Healthcare, military

Six Nines → 99.9999%

Space

How to Increase Availability?
→ Redundancy → at various levels
mount, blue
mode, red
mode, green
mode, DB
Work load on resources
→ Load Balancer → Round Robin
→ Geographical Distribution → CDN (various regional endpoints)
→ Rolling Maintenance
→ Scheduled Maintenance
→ Monitoring → This is more of prevention
→ Circuit Breaking → Stop Taking

This operation can be done without Failure
Two ways of handling
→ Stale Data
→ Stale Data
→ Consistency

→ No Read
→ Unavailable

CAP Theorem

→ Partition Tolerance
→ Availability
Consistency

In real economics, network failures are hard to happen

In Distributed Systems
→ Max 2 of these properties
→ Impossible to guarantee all three properties
→ Practice → Choose Consistency or Availability

CAP:
→ C
→ A
→ P

→ C → In real system partition can cause the guarantee of C → C is not possible
If the connection is gone, the store
Having another connection for Partition is really costly

Consistency | Availability

	Consistency	Availability
Relational DBs	✓	✗
Eventual Consistency	✗	✓
Redis	✓	✓

Kafka	✓	✓
Mongo → Default	✓	✗
→ Switch	✗	✓
HDFS	✓	✓
Cassandra → Default	✗ (Default)	✓
→ Switch	✓	✓
Google Pub/Sub	✓	✗

Real systems are either CP or AP Systems

Strong Consistency
Low Availability

High Availability
Eventual Consistency

Ex: Max 4 people

Messaging system / whatsapp
whatsapp

Step 1 → Find nearest Load Balancing.

Last Day: Course Find

→ Latency → Time between when you initiated operation → finish operation
→ Throughput → Rate
→ in a given time frame
→ Bandwidth: Maximum Throughput
→ Fault: Any Error / Unwanted Behavior
→ Failure: Fault leads to Failure
Fault causes Failure

Fault: Disk Failure, Bad memory, Power Failure
Fail: Disk Failure, Bad memory, Power Failure
Fault causes Failure

→ Resiliency → Recovery
→ Redundancy → Duplication

Stateful System	Stateless system
	→ Each interaction / request is independent of previous requests
Bad → Inefficient → Memory expensive	Bad → Simple → Scalable
Good → Simple → Scalable	Good → Easy Related data

Revision