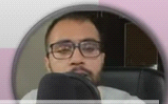


WHAT IS SYSTEM DESIGNING ?

System design is the process of designing the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system.



HLD

Describes the main components that would be developed for the resulting product.

The system architecture details, database design, services and processes, the relationship between various modules, and

LLD

Describes the design of each element mentioned in the High-Level Design of the system.

Classes, interfaces, relationships between different classes, and actual logic of the various components

Monolithic Architecture

Architecture → Internal design details for building the applications.

5 seconds

If all the components and functionalities of a project are entangled and combined in a single codebase, then that is a monolithic application.

Monolithic Architecture has less complexity => Easier to understand => Higher productivity

Monolithic Architecture is also known as centralized system.

When it is good,

1. when we are just starting
2. In monolithic architecture, all the modules are present in the single system, so they require fewer network calls as compared to other architectures.
3. It is comparatively easier to secure monolithic system
4. Integration testing is easier
5. Less confusion

YouTube

Like this video

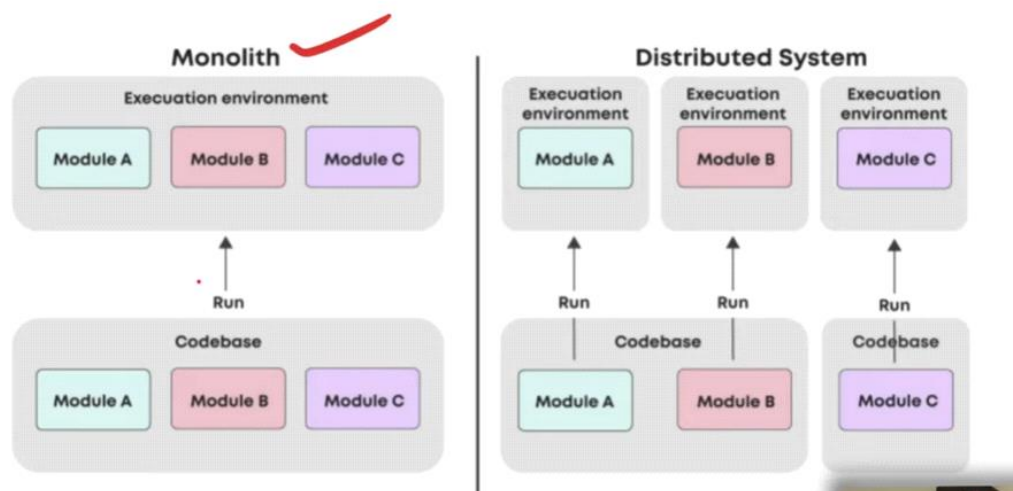


- Single point failure
- Not Scalable

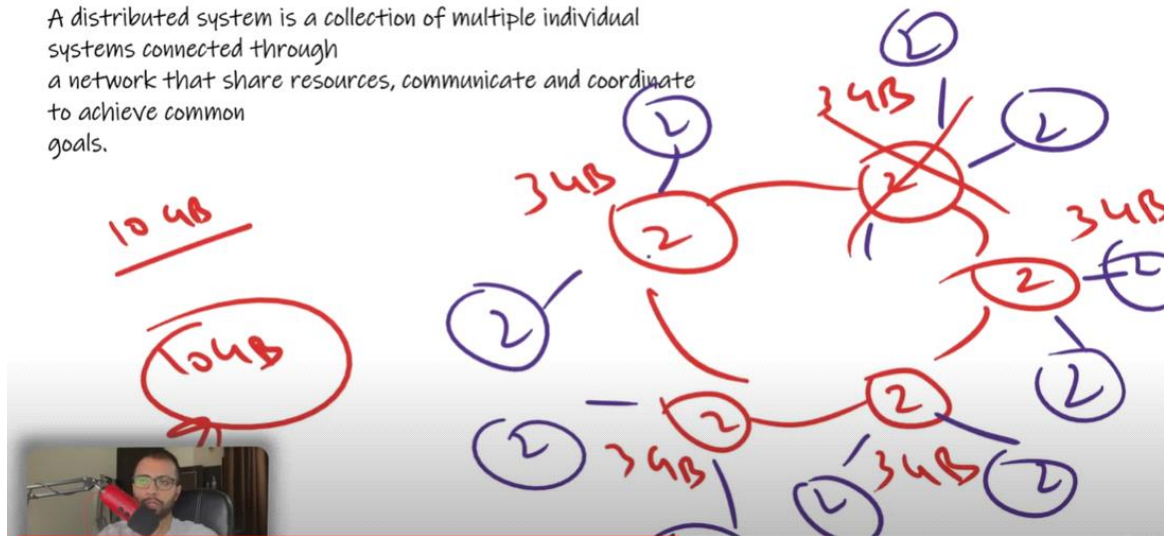
Disadvantages of monolithic architecture

1. In monolithic architecture, every module is combined in a single system, so if there is an error or bug in a single module, it can destroy the complete system.
2. In monolithic architecture, whenever a single module is updated, the whole system needs to be updated to reflect the changes to the users. All modules are present in a single system and are connected to one another, so the whole system needs to be updated.
3. In monolithic architecture, if there is any change in a single module's programming language or framework, it will affect the entire system. The entire system needs to be changed because every module is interlinked and tightly coupled.

Microservices architecture



A distributed system is a collection of multiple individual systems connected through a network that share resources, communicate and coordinate to achieve common goals.



Advantages:-

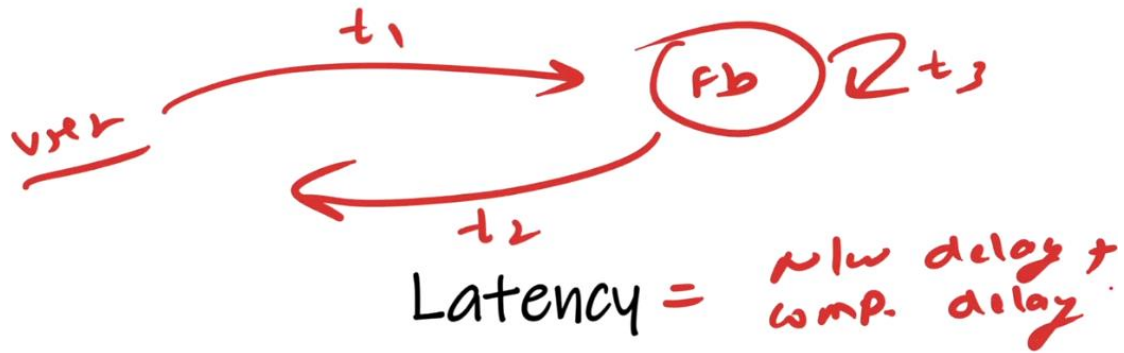
- Scalable :- system can be scalable in horizontally so if there one job was done by one machine then may be 2 machines distributes the job in half-half.
- SPOF:- Achieve the solution of single point of failure.
- Low latency

Disadvantages:-

Disadv

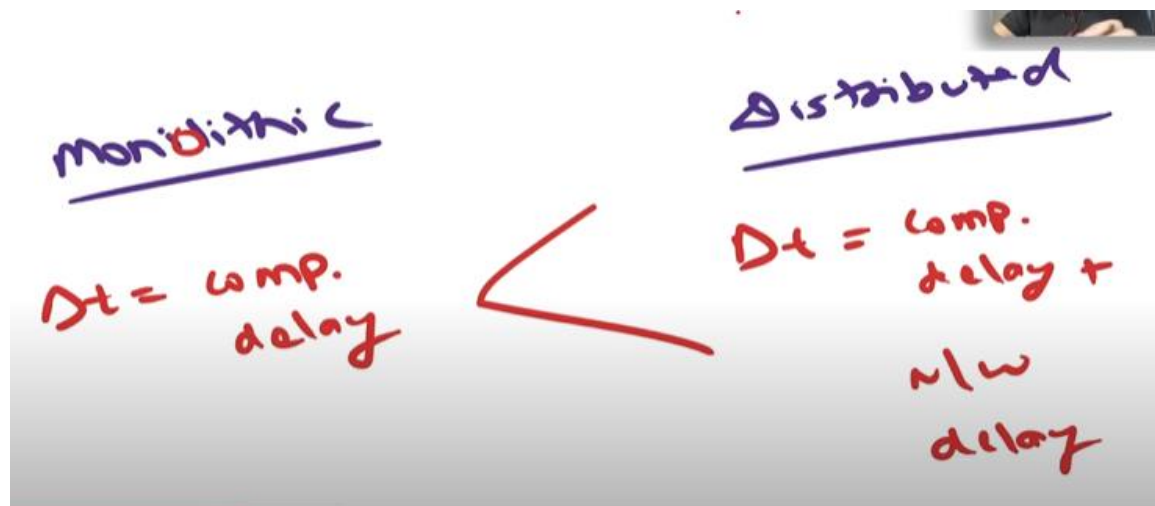
- ① Complex
- ② Management req.
- ③ Difficult to secure
- ④ Message may be lost in b/w nodes

Latency



$$Dt = t_1 + t_2 + t_3$$

Latency = Network Delay + Computational Delay



Reduce Latency

- > Caching
- > CDN (Content Delivery Network)

CDNs are geographically distributed networks of proxy servers and their objective is to serve content to users more quickly. Caching is the process of storing information for a set period of time on a computer.

- > Upgrade System

Throughput

The volume of work or information flowing through a system.

Throughput is the amount of data transmitted per unit of time. It is the process flow rate.

Throughput is measured in bits per second i.e. bps.

Monolithic



Distributed



→ no limit on resources
→ load balancing

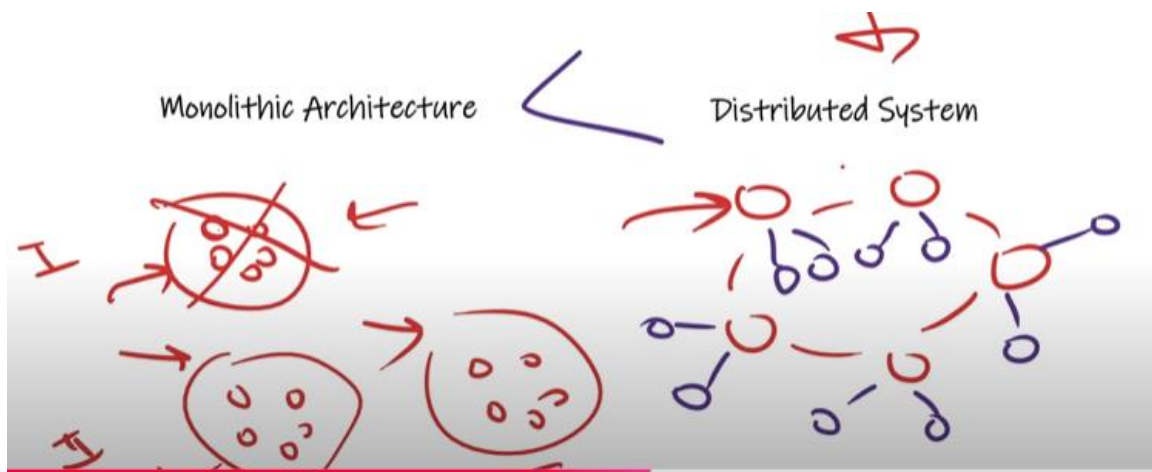
Causes of Low Throughput

- > Latency
- > protocol overhead
- > Congestion

Improving Throughput

- > CDN
- > Caching
- > DS
- > Load Balancer
- > Improve Resources

Availability



Fault Tolerance \propto Availability

How to increase the availability

-> Replication

-> DS

--> Redundancy

How to increase the availability?

① Replication

② DS

③ Redundancy

includes redundancy, but involves the copying of data from one node to another or the synchronization of state between nodes

Redundancy is the duplication of nodes, in case of some of them are failing

Consistency

When more than one client requests the system, for all such requests, it will be called consistent when each client gets the same data. The data should always be consistent, regardless of who is accessing it.

Usually Monolithik System is More consistant than Microlitjik beause in monolithic all structure and database available on one server

When more than one client requests the system, for all such requests, when different clients get different responses due to some recent update that has not been committed to all systems yet, this reading operation will be called dirty read.

Factors Improving Consistency

Improving Network Bandwidth

Stop the read

Replication based on Distance aware strategies

Types of Consistency

Strong Consistency

when the system doesn't allow read operation until all the nodes with replicated data are updated.



Eventual Consistency

User Read requests are not halted till all the replicas are updated rather the update process is eventual. Some users might receive old data but eventually all the data is updated to the latest data.

Weak Consistency



CAP Theorem

CAP Theorem

C - Consistency

A - Availability

P - Partition Tolerance

Consistency - > Synchronized

Availability -> High fault Tolerance

Partition Tolerance -> made up with two words Partition (In Distributed System or microlithic architecture application is divided in components and deployed on different server but when we any server goes down it is called Network Partitining) Tolerance (It basically means we have to make less chances of failur)

CAP Theorem

C - Consistency

A - Availability

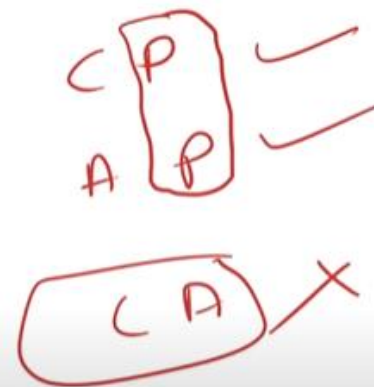
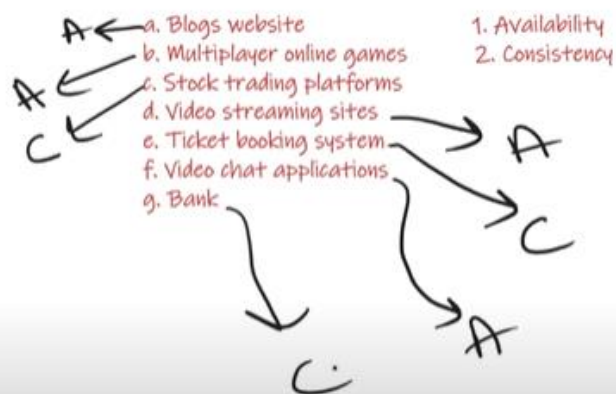
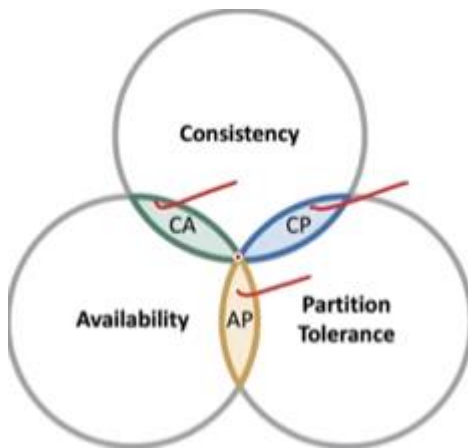
P - Partition Tolerance



For a distributed system, the CAP Theorem states that it is possible to attain only two properties and the third would be always compromised.

The system requirements should define which two properties should be chosen over the rest.

- The system designer can select Consistency and Partition Tolerance but the availability would be compromised then.
- The system designer can select Partition Tolerance and Availability but the consistency would be compromised then.
- The system designer can select Availability and Consistency but the Partition Tolerance would be compromised then.



Scalability

1> Vertical Scaling : In one machine increase only hardware specification.

Pros :

Easy implementation
 Less power in mentaince
 Management is easy

Cons:

SPOF
 Limit
 price

2> Horizontal Scaling: We increase By same machine acc. To need

Pros:

Reduces SPOF

No limit
Price is low comparing with vertical scaling

Cons:

Power Consumption is more

Implementation is tough

Management is difficult

Difference between Redundancy and Replication

Redundancy

Redundancy

Redundancy is simply the duplication of nodes or components so that when a node or component fails, the duplicate node is available to service customers.

Types Of Redundancy

1.

Active Redundancy is considered when each unit is operating/active and responding to the action. Multiple nodes are connected to a load balancer, and each unit receives an equal load.

2.

Passive Redundancy is considered when one node is active or operational and the other is not operating. During the breakdown of the active node, the passive node maintains availability by becoming the active node.

Replication

Replication redundancy + synchronization

At same time data should be same on all the servers

Basically Replication use in database

1. **Active replication** is a computing technique that processes the same request at every replica. In an active replication environment, nodes can replicate data to other nodes in a cluster. This allows all nodes to handle read and write operations simultaneously, unlike traditional master-slave configurations.

2. **Passive replication**

every read - write → master

If master goes down one slave becomes master

Master-slave replication can be either synchronous or asynchronous. The difference is simply the timing of propagation of changes. If the changes are made to the master and slave at the same time, it is synchronous. If changes are queued up and written later, it is asynchronous

load balancer

Load Balancing is the process of efficient distribution of network traffic across all nodes in a distributed system.

****Roles of Load Balancer****

1. The load distribution is equal over every node.
2. Health check (if the node is not operational, the request is passed to another node that is up and running).
3. Load balancers ensure high scalability, high throughput and high availability

****When to use it and when not to ?****

In monolithic or in case of vertically scaled system we don't need it. But in microservice architecture we need it.

****Challenges of Load Balancing****

Single Point Of Failure - During a load balancer malfunctioning, the communication between clients and servers would be broken. To solve this issue, we can use redundancy. The system can have an active load balancer and one passive load balancer

****Advantages of using load balancers:****

1. ***Optimisation***: Load Balancers help in resource utilisation and lower response time, thereby optimising the system in a high traffic environment.
2. ***Better User Experience***: Load balancers help in reducing the latency and increasing availability, making the user's request go smoothly and error-free.
3. ***Prevents Downtime***: Load Balancers maintain a record of servers that are non-operational and distribute the traffic accordingly, therefore ensuring security and preventing downtime, which also helps increase profits and productivity.
4. ***Flexibility***: Load Balancers have the flexibility to re-route the traffic in case of any breakdown and work on server maintenance to ensure efficiency. This helps in avoiding traffic bottlenecks.
5. ***Scalability***: When the traffic of a web application increases suddenly, load balancers can use physical or virtual servers to deliver the responses without any disruption.
6. ***Redundancy***: Load Balancing also provides in-build redundancy by re-routing the traffic in case of any failure.



****Load Balancing Algorithms****

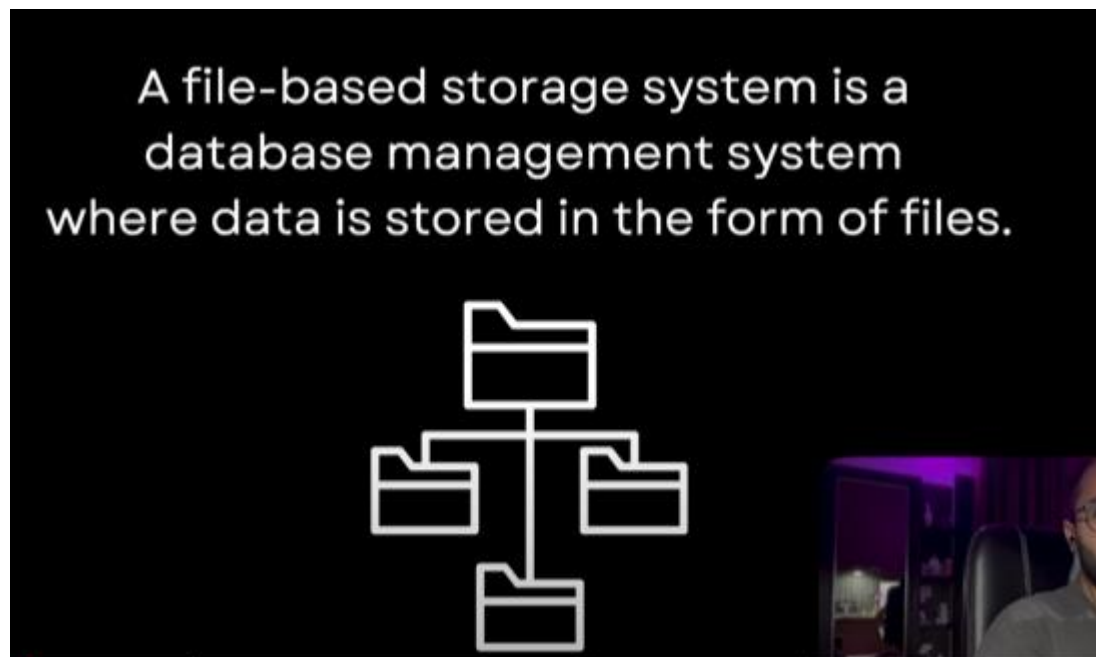
1. Round Robin (Static) - rotation fashion.
2. Weighted Round robin (Static) - It is similar to the Round Robin when the servers are of different capacities. (some node can have better resources, others might not have)
3. IP Hash Algorithm (Static) - The servers have almost equal capacity, and the hash function (input is source IP) is used for random or unbiased distribution of requests to the nodes.
4. Source IP Hash (Static) - combines the server and client's source and destination IP addresses to produce a hash key. The key can be used to determine the request distribution.
5. ~~Least Connection Algorithm~~ (Dynamic) - Client requests are distributed to the application server with the least number of active connections at the time the client request is received.
6. Least Response Time (Dynamic) - The request is distributed based on the server which has the least response time.

Caching

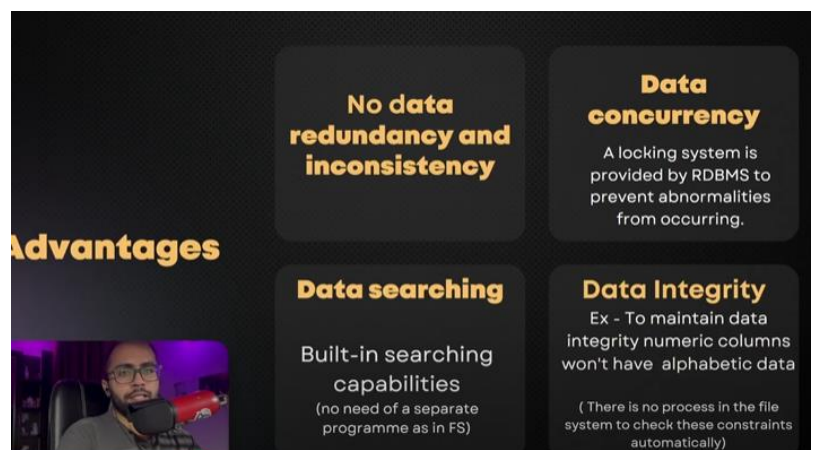
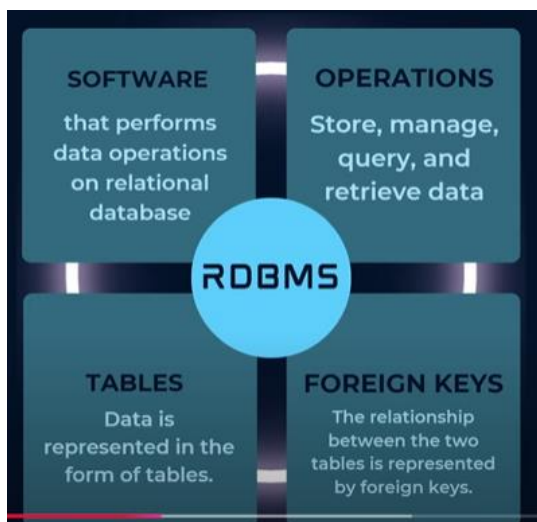
Caching is a technique that stores data in a cache, a hardware or software component, to make it available faster when requested again. Caching can improve a system's performance and scalability by making data more accessible

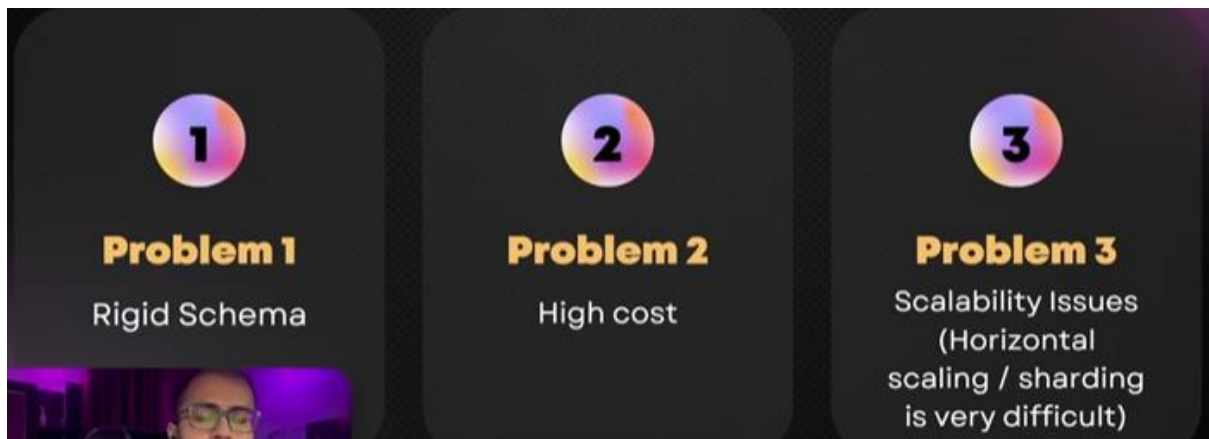
Read Some More From Google

File based storage system



RDBMS





The slide features three numbered problems in a dark-themed layout. Each problem is represented by a circular icon with a number, a title, and a description. A small video feed of a person is visible in the bottom-left corner of the slide.

1	2	3
Problem 1	Problem 2	Problem 3
Rigid Schema	High cost	Scalability Issues (Horizontal scaling / sharding is very difficult)

NoSQL Databases

NoSQL

It stands for “non-SQL” database
or we can say that it is a non-relational database

NoSQL is the umbrella term comprising of four different types of databases.

Key value db



Document db



Columnar db



cassandra

Graph db



Key value db



key value database
generally used for caching

Document db



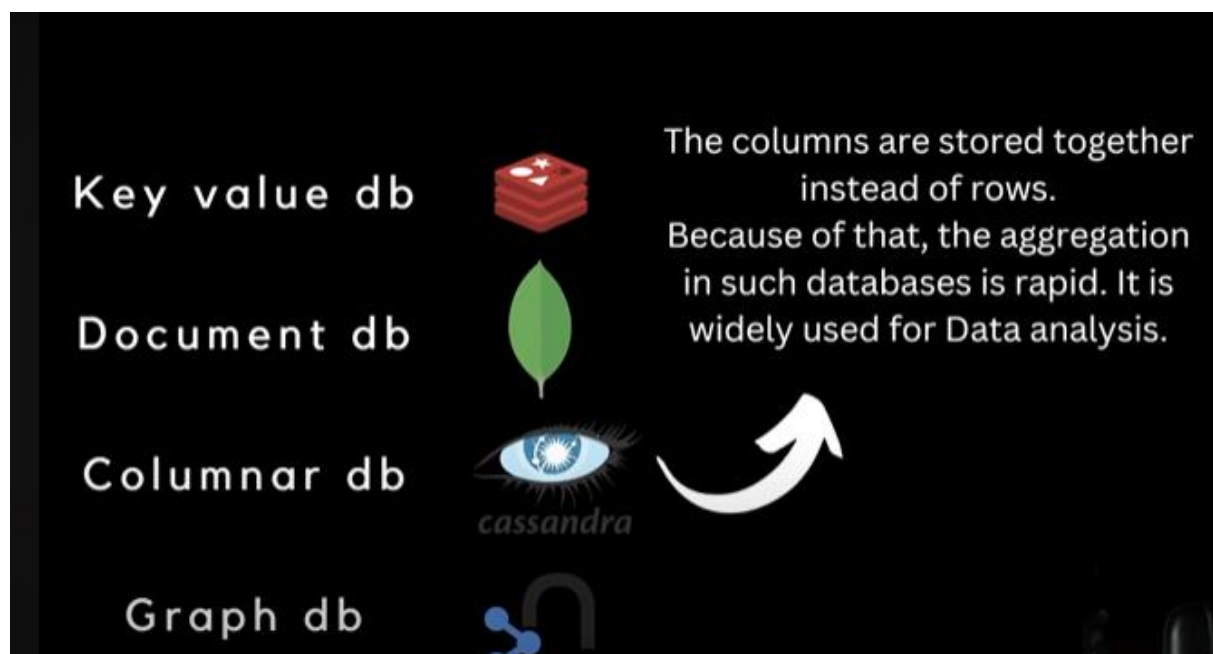
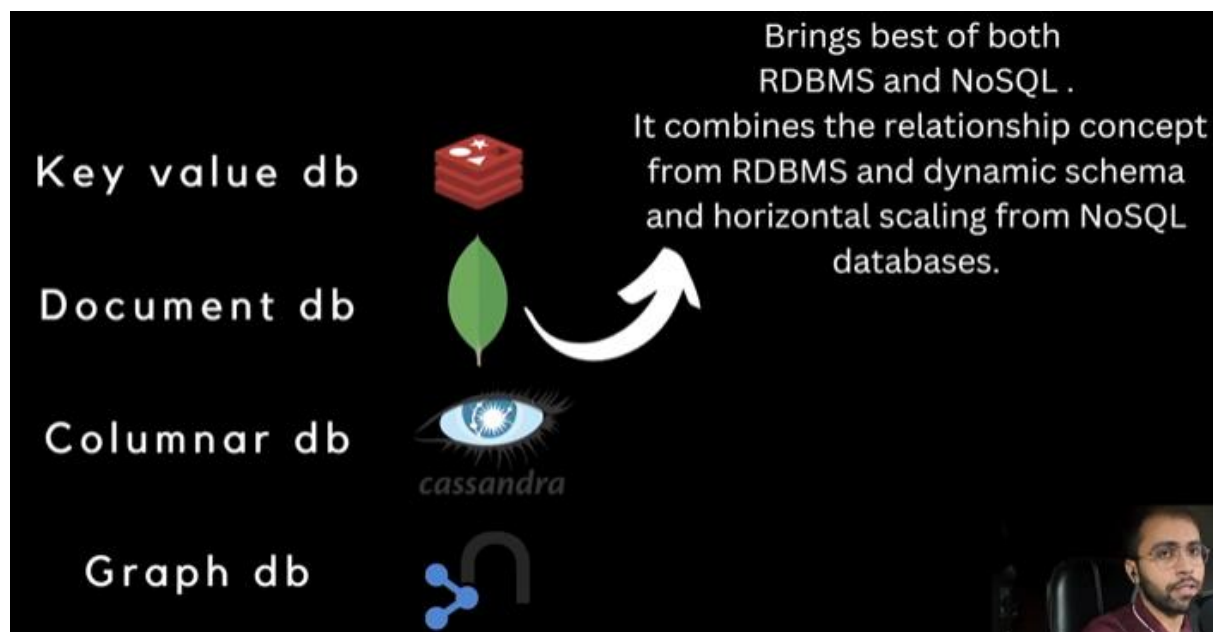
Columnar db



cassandra

Graph db

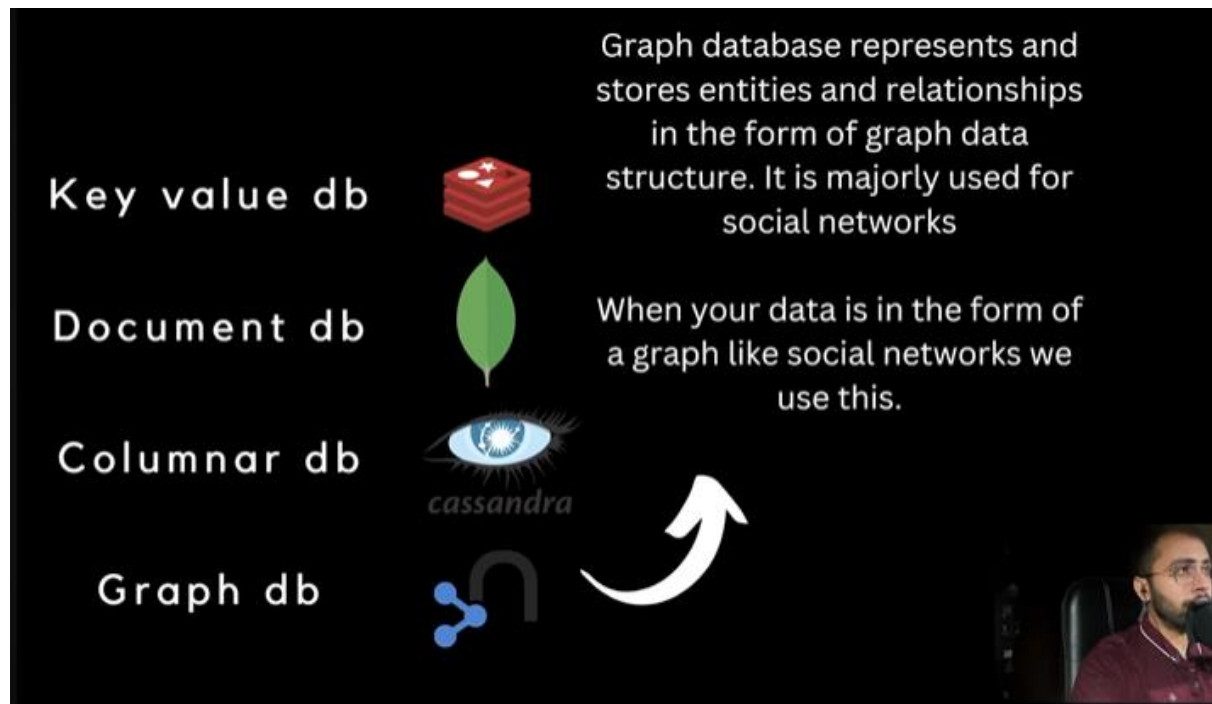




Because it just needs to read these specific columns, columnar DB conducts such queries quickly (while row-based DB would have to read the entire data).

Row oriented (Relational)		
Students		
ID	First name	Last name
1	Luna	Lovegood
2	Hermione	Granger
3	Ron	Weasley

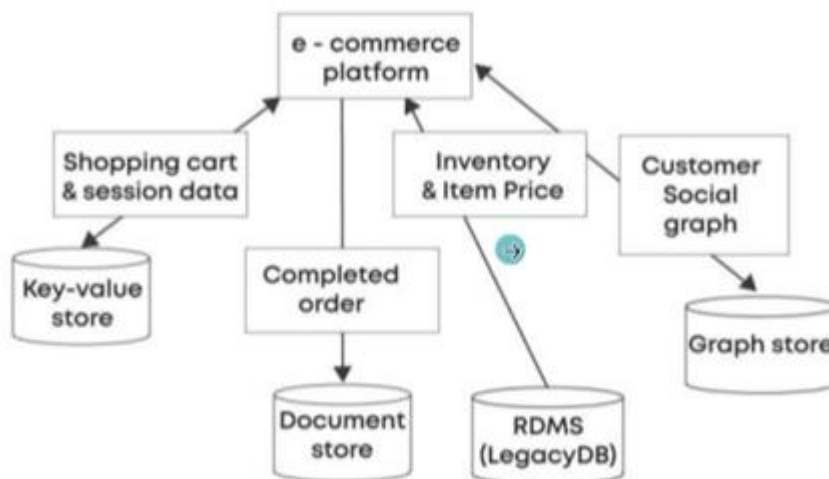
Column oriented		
Students		
ID	First name	Last name
1	Luna	Lovegood
2	Hermione	Granger
3	Ron	Weasley



Polyglot Persistence

when one data base is not enough for our application and we implement multiple database in one application then it is known as Polyglot Persistence.

Polyglot persistence



What is denormalization in RDBMS

Normalization



Putting data in multiple tables to avoid redundancy

Denormalization

It combines the data and organizes it in a single table. Denormalization is the process of adding redundant data to the normalized relational database to optimize its performance.

e_id	e_name	dept_id

dept_id	dept_name	dept_desc

e_id	e_name	dept_id	dept_name	dept_desc

Benefits of denormalization

1. Faster data read operations



2. Management convenience



3. High data availability



4. Reduces the number of network calls to fetch data from multiple places.



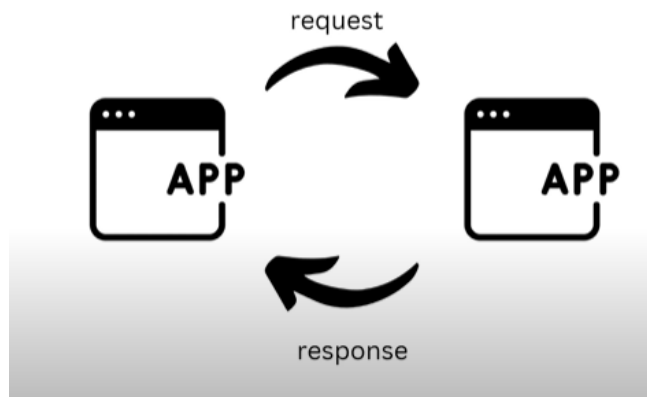
indexing

Indexing creates a lookup table with the column and the pointer to the memory location of the row, containing this column.

B-trees data structure is used to store the indexing as it is a multilevel format of tree-based indexing, which has balanced binary search trees.

Synchronous communication

Synchronous
communication
(Blocking call)



- 1.To achieve consistency
- 2.Transaction

Industrial use cases

- 1.Stock Market 
- 2.Bank payments   
- 3.Ticket Booking 
- 4.Real time decision making

asynchronous



Asynchronous communication (Non Blocking call)



synchronous

Application waits for this response before allowing the user to add the product into the cart.



synchronous

Application waits till the confirmation of payment has been successfully received from the bank.



asynchronous

It can take some time for the notification to arrive, but in the meantime a user can do anything on the application without waiting for the notification.

Where is it necessary ?

1. Computation takes a lot of time



2. Scalability of application



3. Avoid cascading failure



message based communication

- Client sends request in the form of message
- receives response in form of message
- It is async, so client is not required to halt the process and wait for the process

Producer

Consumer

Agent

P2P Model -> Peer to Peer Model

Public Subscribe Model

web server



Tools or programs that help keep the web application always up and running.

The term *web server* can refer to hardware or software, or both of them working together.

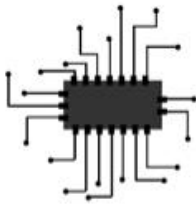
1. On the hardware side, a web server is a computer that stores web server software and a website's component files (for example, HTML documents, images, CSS stylesheets, and JavaScript files). A web server connects to the Internet and supports physical data interchange with other devices connected to the web.
2. On the software side, a web server includes several parts that control how web users access hosted files. At a minimum, this is an *HTTP server*. An HTTP server is software that understands URLs (web addresses) and HTTP (the protocol your browser uses to view webpages). An HTTP server can be accessed through the domain names of the websites it stores, and it delivers the content of these hosted websites to the end user's device.

At the most basic level, whenever a browser needs a file that is hosted on a web server, the browser requests the file via HTTP. When the request reaches the correct (hardware) web server, the (software) *HTTP server* accepts the request, finds the requested document, and sends it back to the browser, also through HTTP. (If the server doesn't find the requested document, it returns a [404](#) response instead.)



communication protocol

The rules, syntax, semantics and timing of communication and possible error recovery methods.



These protocols can be implemented using hardware, software, or a combination of both.

Models

1. Push
2. Pull / Polling
3. Long Polling
4. Socket
5. Server sent events

Models

Client requests, server respond

1. Push

2. Pull / Polling

3. Long Polling

4. Socket

5. Server sent events



what if client sends 100 request but get reponse to just 5 requests

Models

Client requests, server keep it open until it gives response

1. Push

2. Pull / Polling

3. Long Polling

4. Socket

5. Server sent events



Models

Disadvantage:

1. Ordering issue
2. Server will be always busy

1. Push

2. Pull / Polling

3. Long Polling

4. Socket

5. Server sent events



Models

1. Push

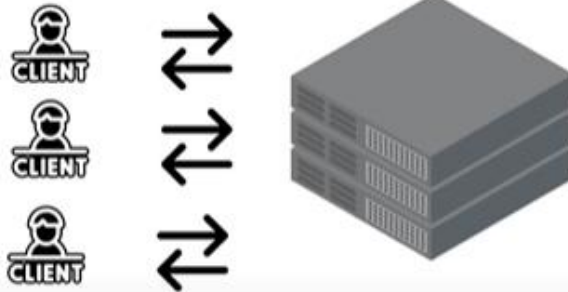
2. Pull / Polling

3. Long Polling

4. Socket

5. Server sent events

The client opens the connection with the server and keeps it always active



Models

1. Push

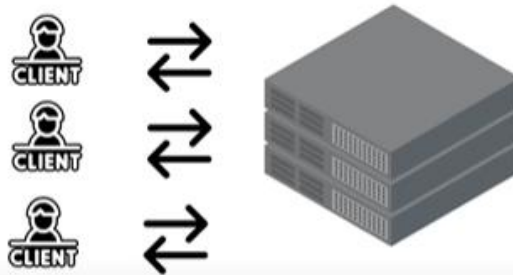
2. Pull / Polling

3. Long Polling

4. Socket

5. Server sent events

The server pushes new events to the client.
This method reduces the server load.



Models

1. Push

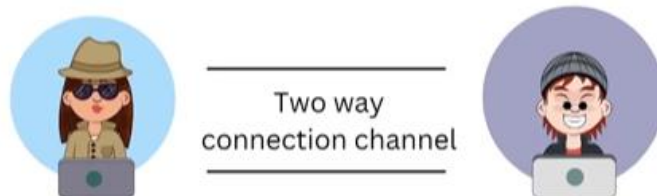
2. Pull / Polling

3. Long Polling

4. Socket

5. Server sent events

A socket is an endpoint of a two-way connection link between two servers/nodes over the network



Models

1. Push
2. Pull / Polling
3. Long Polling
4. Socket
5. Server sent events \

Ex - Cricbuzz

Client subscribes to the server "stream", and the server will send a message ("stream of events") to the client until the server or client closes the stream.

One way connection

Long-Lived Connection



Microservices architecture

Web App : Intractive'

Web Site : Static

1. Client: System that initiates the communication over the Network.
2. Server: System that receives the request over the Network.

Therefore, examples of clients can be -

mobile apps, 
web-based consoles, 
laptops, etc. 

A web server can also act as a client when requesting information

What is required for communication to happen ?

1. Language independent
2. Fast
3. Enable communication over the n/w
4. light weighted

REST



REpresentational State Transfer

A technique / style /
standard to provide those
requirements

To use that standard we
expect from the server that
it should **expose REST API**

this request can be of 4
types



Client



Ex - 173.76.310.45:7001/ios/nflx/plan-listing

URI

SOA (Service Oriented Architecture)

Service-Oriented Architecture(SOA) is a style of
architecture that promotes loose coupling and granular
applications to make the components of the software
reusable.

Advantages of Service-Oriented Architecture

1. Selective scaling

2. Different tech stacks



3. loose coupling

4. Agile



Disadvantages of Service-Oriented Architecture

1. Higher latency



2. Complex to secure



3. Cascading failures



4. Complex understanding



Microservices Architecture

Microservices architecture is an evolved version of SOA that promotes software components to be loosely coupled.

It is the most granulated type of architecture design and every service is completely independent of the other

SOA

Services can share data storage

Less scalable architecture

Deployment is time-consuming

Focused on maximizes application



Microservices

Each microservice has separate and independent data storage

Highly scalable architecture

Deployment is easy and less time-consuming

More focused on decoupling

Tier architecture

A web application can be designed according to the n-tier architecture where tiers are different layers of architecture.

Tier architecture helps make modifications and updation of different components easy. It helps in assigning dedicated tasks and roles to each component.

1 tier



2 tier



3 tier



Authentication and Authorization

Who are you ?



Authentication

VS

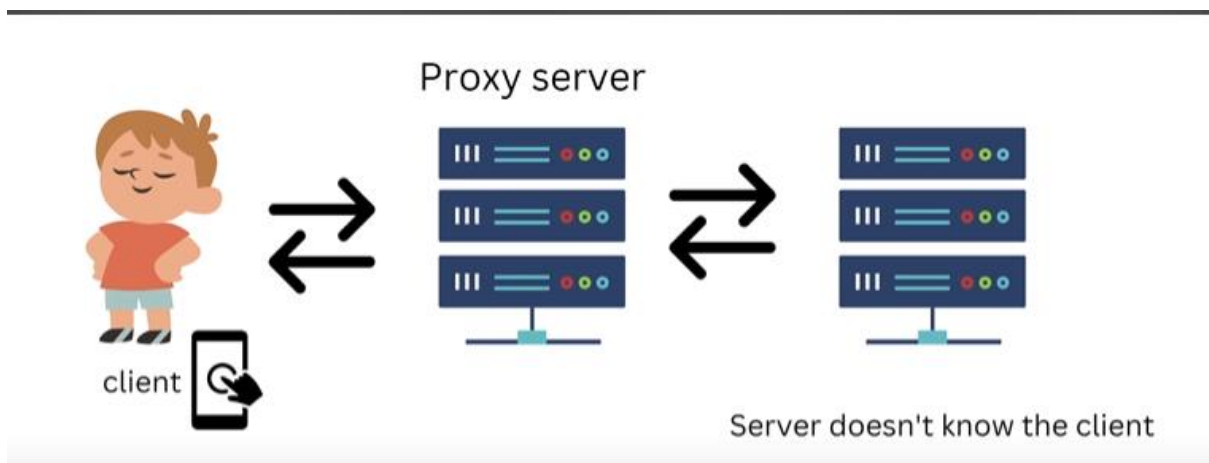


What can you do ?

Authorization

Forward proxy and reverse proxy

A proxy server is a hardware or a piece of software that is placed between a client and an application to provide intermediary services in the communication.



Forward proxy **hides the client**

Reverse proxy **hides the server**

When to use reverse proxy ?

1. When I don't want to expose the server
2. To make user feel there is only one server
3. Provide load balancing for the server
4. Allows administrators to swap the backend servers without disturbing the traffic
5. Filter out some requests