SHARDING:

Distributing data across multiple machines for scalability and efficiency.

VERTICAL SCALING:

Increasing the resources of a single machine to handle more load.

HORIZONTAL SCALING:

Adding more machines to distribute the load and increase capacity.

LOAD BALANCER:

A System component that distributes incoming network traffic across multiple servers.

ROUND ROBIN:

A simple and fair scheduling algorithms that cycles through the servers for load distribution.

DISTRIBUTED SYSTEMS:

System where the components are located on different networked computers, which communicated and coordinate their actions by passing message.

CONSISTENT HASHING:

A technique used to evenly distribute data across machines to handle dynamic nodes without large-scale rehashing.

LATENCY BASED LOAD BALANCING:

Distributing load to servers based on their response time to minimize latency.

MACHINE LEARNING LOAD BALANCERS:

Load balancer that uses machine learning to predict server performance and manage loads efficiently.

SCALABILITY:

The capacity of a system to handle a growing amount of work, or its potential to accommodate growth.

ICANN:

Non-profit organization for coordinating the global domain name system.

CAP THEROM:

States that in the presence of a network partition, one can choose at most two out of three data system: Consistency, Availability and Partition tolerance.

**Revision Notes: Building a Scaling Mindset**

**Introduction to Scaling**

* **Scaling Mindset**: The ability to build systems thinking about growth and demand from the outset. A scalable system should handle large traffic, be highly available, and maintain performance as the load increases【8:5†transcript.txt】.

**Importance of Scalability**

* **Examples of Scalable Applications**: Amazon, Facebook, Google Pay, Hotstar, Netflix, YouTube, which are designed to handle billions of users【8:5†transcript.txt】.
* **Coldplay Tickets Analogy**: Highlighted issues experienced by platforms like BookMyShow in handling a large volume of transactions simultaneously, emphasizing the need for scalability【8:5†transcript.txt】.

**High-Level Concepts**

**Load Balancing**

* **Definition**: Distributes incoming network traffic across multiple servers to ensure no single server becomes overwhelmed.
* **Types of Load Balancing**:
  + **Round Robin**: Distributes requests evenly across all servers.
  + **Weighted Round Robin**: Gives preference to more powerful servers.
  + **Latency-Based**: Directs requests to the server that can respond the quickest【8:10†transcript.txt】.

**Consistent Hashing**

* **Concept**: A technique to distribute data across servers so that re-allocating data due to changes in the number of servers (scaling up or down) requires minimal data movement.
* **Impact**: Enables efficient distribution and retrieval, ensuring even load across servers【8:1†transcript.txt】【8:0†transcript.txt】.

**Caching**

* **Role**: Improves performance by storing frequently accessed data closer to the application.
* **Types**:
  + **Local Cache**: Stored within a single server.
  + **Distributed Cache**: Spreads across multiple servers.
* **Eviction Policies**: Strategies to manage cache content, such as Least Recently Used (LRU)【8:3†transcript.txt】【8:2†transcript.txt】.

**Sharding**

* **Concept**: Partitioning the database into smaller, more manageable pieces that can be distributed across multiple machines.
* **Comparison to Partitioning**: While partitioning typically refers to dividing data within a single database, sharding distributes data across multiple databases【8:11†transcript.txt】【8:12†transcript.txt】.

**System Design Patterns**

* **Architectural Patterns**: Include microservices, which involve breaking down applications into smaller, independent services that can be deployed and scaled individually【8:9†transcript.txt】.

**Vertical vs Horizontal Scaling**

* **Vertical Scaling**: Enhancing existing hardware (e.g., adding more capacity to a single machine).
* **Horizontal Scaling**: Adding more machines to share the load, often seen as more cost-effective and sustainable【8:8†transcript.txt】.

**Practical Considerations in Scaling**

* **Database Management**: Considers strategies like using NoSQL databases (e.g., Redis, Cassandra) for flexibility and scalability【8:2†transcript.txt】.
* **Distributed Systems**: Deals with challenges like data consistency, machine failures, and network partitions【8:17†transcript.txt】【8:18†transcript.txt】.

**Additional Insights**

* **Cost Efficiency**: Emphasizing the balance between scaling and budget constraints, avoiding unnecessary expenses like excessive server resources【8:14†transcript.txt】.
* **Design Thinking**: Encouraged strategic design and planning to avoid scalability issues later【8:14†transcript.txt】.

These notes aim to encapsulate key points from the class, structured to provide a comprehensive understanding of scalability concepts.

Scale: High traffic

Scalable: Capable of handling huge traffic

[**https://howdns.works/**](https://howdns.works/)

- Story of Joshua for bookmark website in del.icio.us (The Website)

- ICANN is no profit organization. It maintains entire world’s Domain Name Servers.

- When we search any URL in browser 1st time then 1st we check Domain name associated IP in browser cache if not present then check to laptop’s cache if not present then check in ISP’s cache if not present then check to DNS cache(There are total 13 Domain Name servers) if not present then check to name server cache if not present then check to detail server , once ip got it add in all cache browser to Name server

- IP address is not fixed for laptop in general term ip addresses are not static, only mac address fix.

- Check SUBNET?????

**Vertical Scaling:**

Easy to manage

Single point of failure

**Horizontal Scaling:**

Cheaper

Easily available

Can add as many m/c as we can

No Single point of failure

Management becomes very difficult

Some Load balancer Algorithms:

- Round Robin

- Weighted Round Robin

- Latency [Fastest Request received by server takes 1st]

- Smart Load Balancer [Comes with machine Learning algo which predict and decide server]

Sharding:

Distributing data on multiple machines.

HLD Curriculum (Tentative):

1) Load balancing & Consistent hashing

2) Caching:

- local vs global/distributed

- eviction policies

- algorithms

- invalidation policies

3) CAP | PACELC Theorem

4) SQL vs NoSQL

5) Case Studies

- Build a typeahead

- messaging (WA/Slack)

- Elastic Search

- Ride Booking (UBER- Quad tree)

- File Storage System (S3)

- Video Streaming (Hotstar- Live +recoreded)

6) Messaging Queue

7) Popular interview questions.