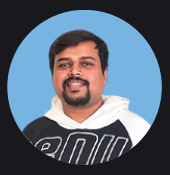


Omkar
@omsrivastava

Helps job aspirants to crack FAANG product companies

SYSTEM DESIGN

Important Concepts

Devops free webinar, Link in description


Omkar
@omsrivastava

Introduction

System design involves planning software development by understanding requirements, dividing the problem, selecting technologies, designing components, and ensuring scalability, performance, reliability, security, and iterative testing.

Why system Design ?

Meet user needs: Ensure the software satisfies user requirements.
Ensure efficiency: Optimize resource usage.
Enable scalability: Allow system growth without performance issues.
Maintain reliability: Ensure consistent and correct operation.
Facilitate maintenance: Simplify updates and troubleshooting.
Enhance security: Protect against unauthorized access.
Provide competitive advantage: Deliver superior performance and features.

Omkar
@omsrivastava

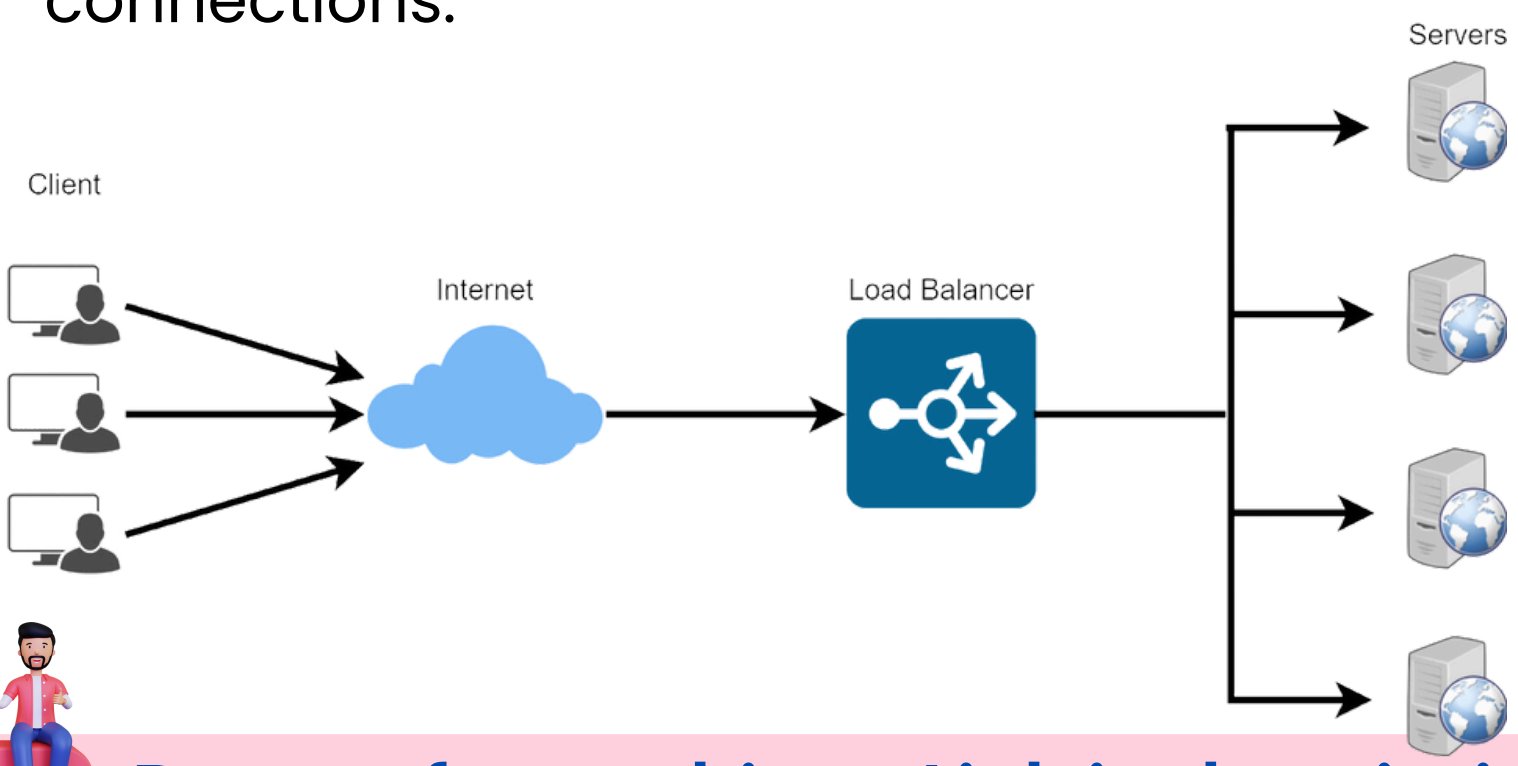
Load Balancer

Problem Statement :

- High-traffic websites and applications struggle to efficiently distribute incoming network traffic across multiple servers, leading to performance issues and potential server overload

Solution :

- **Load Balancer**:
  - Distributes incoming network traffic across multiple servers to prevent overload and optimize resource usage.
  - Monitors server health and routes traffic based on predefined algorithms like round-robin or least connections.




Omkar
@omsrivastava

- **Benefits:**
- Increases reliability by reducing downtime and managing server failures smoothly.
- Enhances scalability by adjusting server numbers based on traffic.
- Boosts performance by evenly spreading traffic and avoiding server overload.

Advantages Of Load Balancer:

- **Enhanced Performance:** Balances traffic to prevent server overload and speed up responses.
- **High Availability:** Redirects traffic from failed servers for continuous service.
- **Scalability:** Easily adds servers to handle more traffic without downtime.
- **Optimized Resource Use**: Efficiently allocates resources across servers.
- **Traffic Management**: Provides flexible routing and distribution strategies.
- **Security Enhancement**: Protects against common threats and attacks.
- **Simplified Maintenance**: Centralizes management for easier maintenance tasks.

Omkar
@omsrivastava

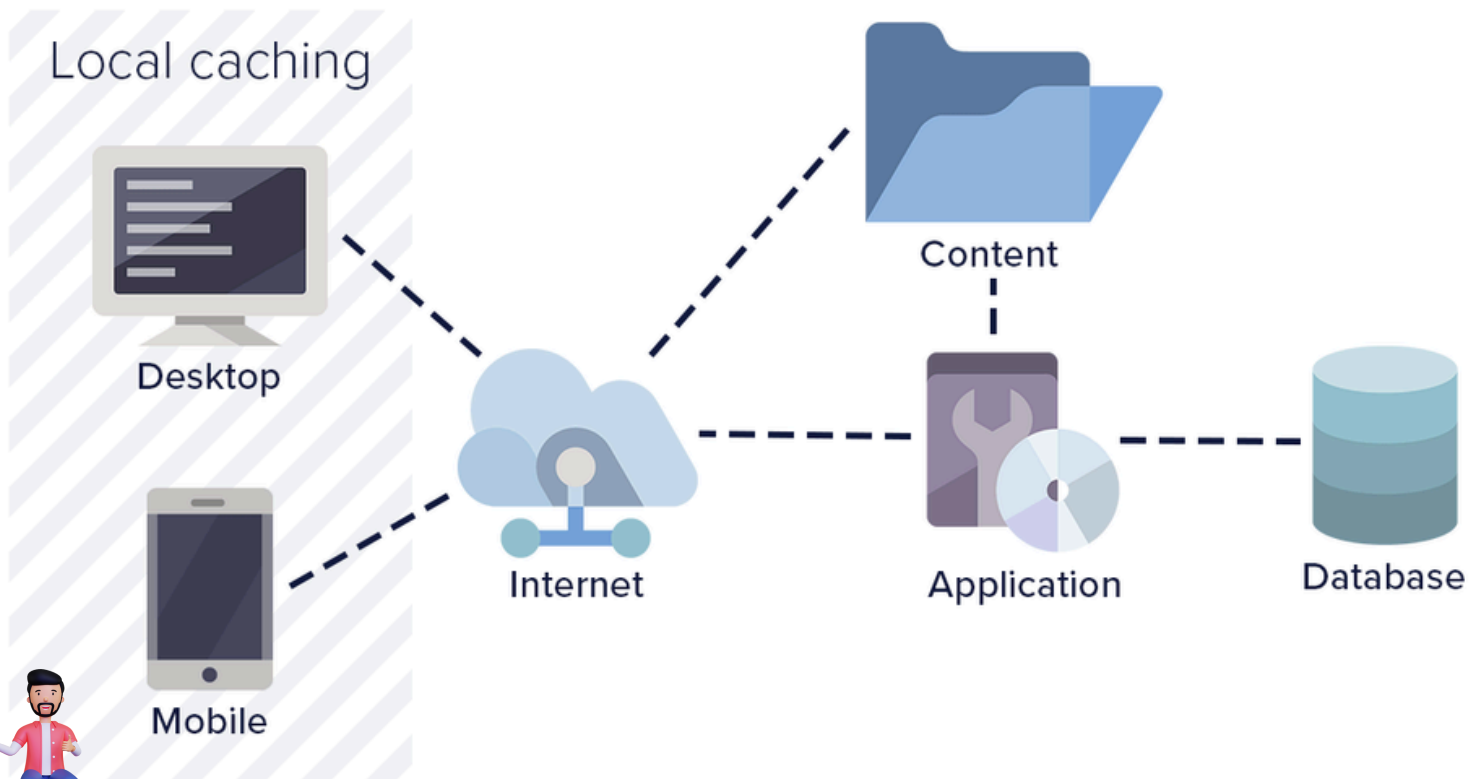
Caching

Problem Statement :

- High-traffic applications face latency and performance issues due to repeated access of data from slower sources like databases or external APIs.

Solution :

- **Caching**:
  - Store frequently accessed data in a high-speed cache closer to the application or users.
  - When data is requested, check the cache first to reduce reliance on slower sources.



- **Benefits:**
    - Improves application performance, reduces latency, and minimizes resource consumption.
    - Enhances scalability by reducing load on backend systems.
    - Ensures data consistency with expiration policies.

- **Note:**

Caching plays a pivotal role in optimizing web performance by storing frequently accessed data closer to users, reducing latency and server load. It enhances user experience, speeds up content delivery, and improves overall website responsiveness, making it indispensable for modern web applications.


Omkar
@omsrivastava

cache eviction policies

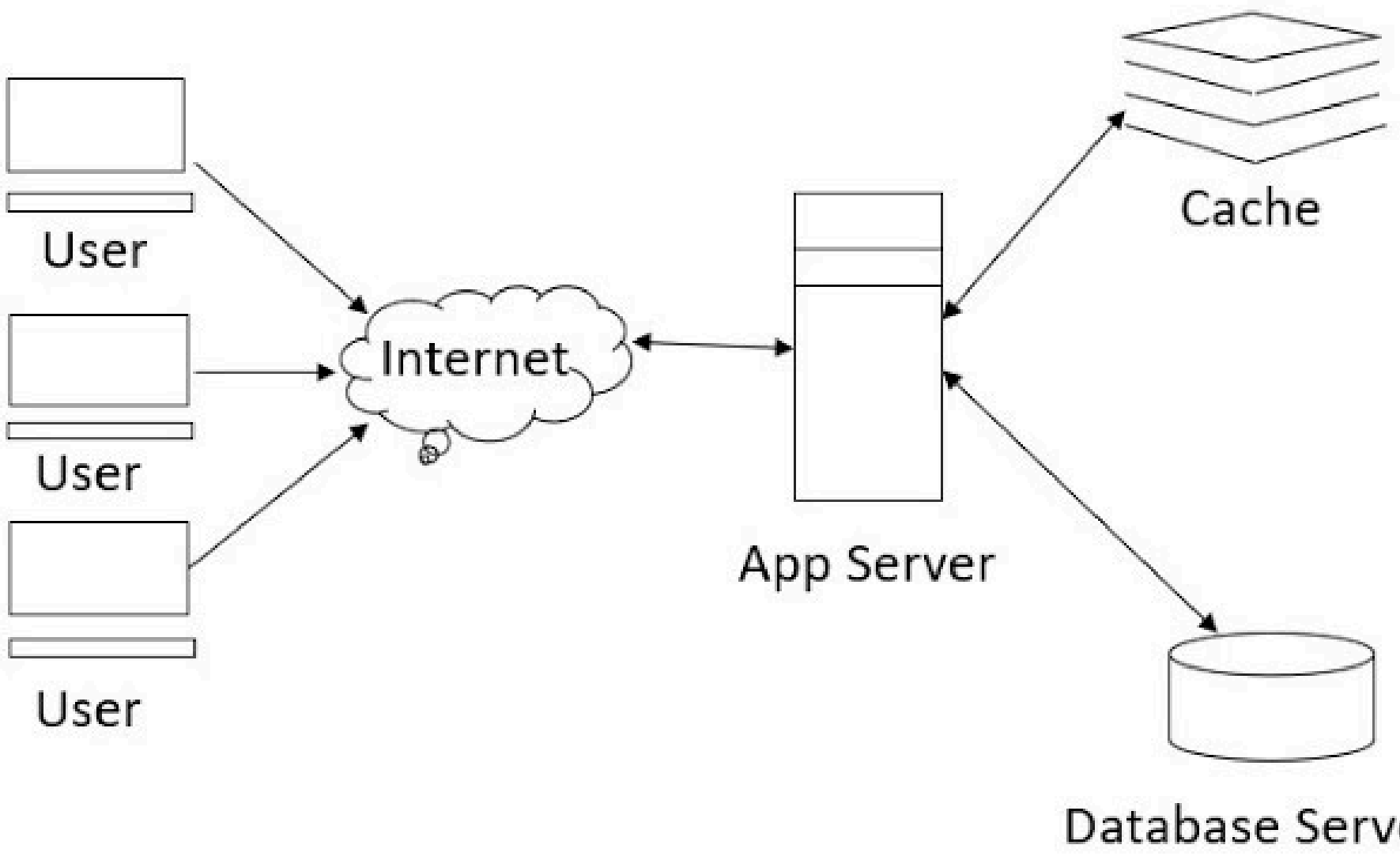
Problem Statement : 

- Caching systems need efficient strategies for managing the eviction of cached items to maintain optimal performance and resource utilization.
- Without effective eviction policies, caches may become inefficient, leading to increased memory usage and reduced cache hit rates.

Solution : 

- **Eviction Policies**:
  - Define rules for removing or replacing cached items when cache reaches capacity.
  - Common policies: Least Recently Used (LRU), Least Frequently Used (LFU), First-In-First-Out (FIFO).
- **Benefits**:
  - Optimizes cache utilization and maintains consistent performance.
  - Helps in balancing between retaining frequently accessed data and making room for new items.
  - Ensures efficient resource allocation and improves cache hit rates.


Omkar
@omsrivastava



Note:

- Cache eviction policies decide which items are removed when full.
- Common methods are LRU, FIFO, LFU, and Random Replacement.
- Choosing a policy depends on application needs and behavior.


Omkar
@omsrivastava

Data Partitioning

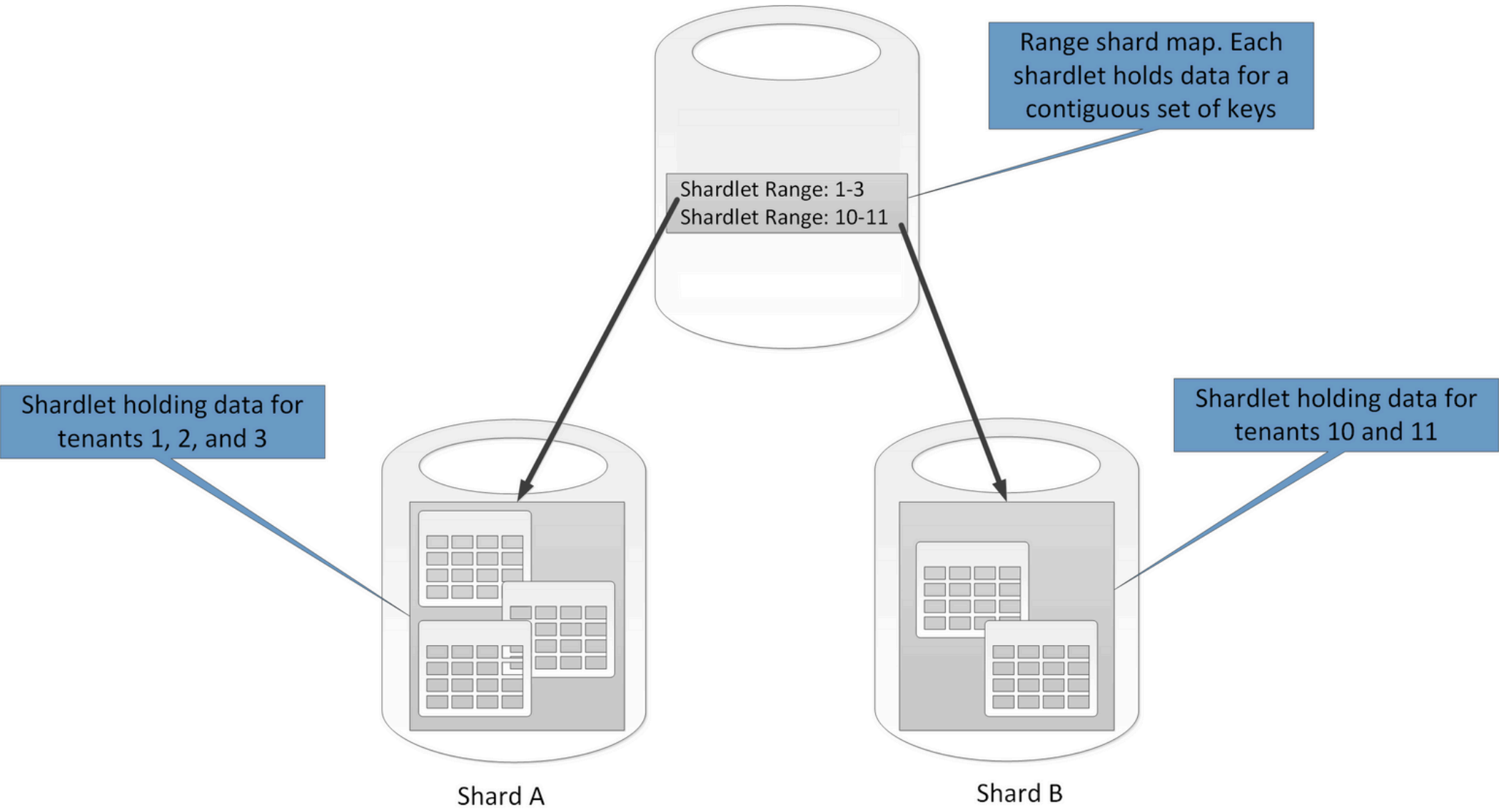
Problem Statement : 

- Large-scale distributed systems face challenges in efficiently managing vast amounts of data across multiple nodes.
- Issues include uneven data distribution, hotspots, increased network traffic, and reduced performance.

Solution : 

- **Data Partitioning**:
  - Divide dataset into smaller partitions and distribute across nodes.
- **Partitioning Strategies**:
  - Range-based, hash-based, and key-based partitioning.
- **Benefits**:
  - Improves performance, scalability, and fault tolerance.
  - Enables parallel processing, reduces hotspots, and optimizes network usage.

Omkar
@omsrivastava



Note:

Data partitioning involves splitting a database into smaller segments to distribute data across multiple nodes or servers. Common partitioning strategies include range-based, hash-based, and list-based partitioning. The choice of partitioning method depends on factors like data distribution, access patterns, and scalability requirements.

Omkar
@omsrivastava

Data Redundancy

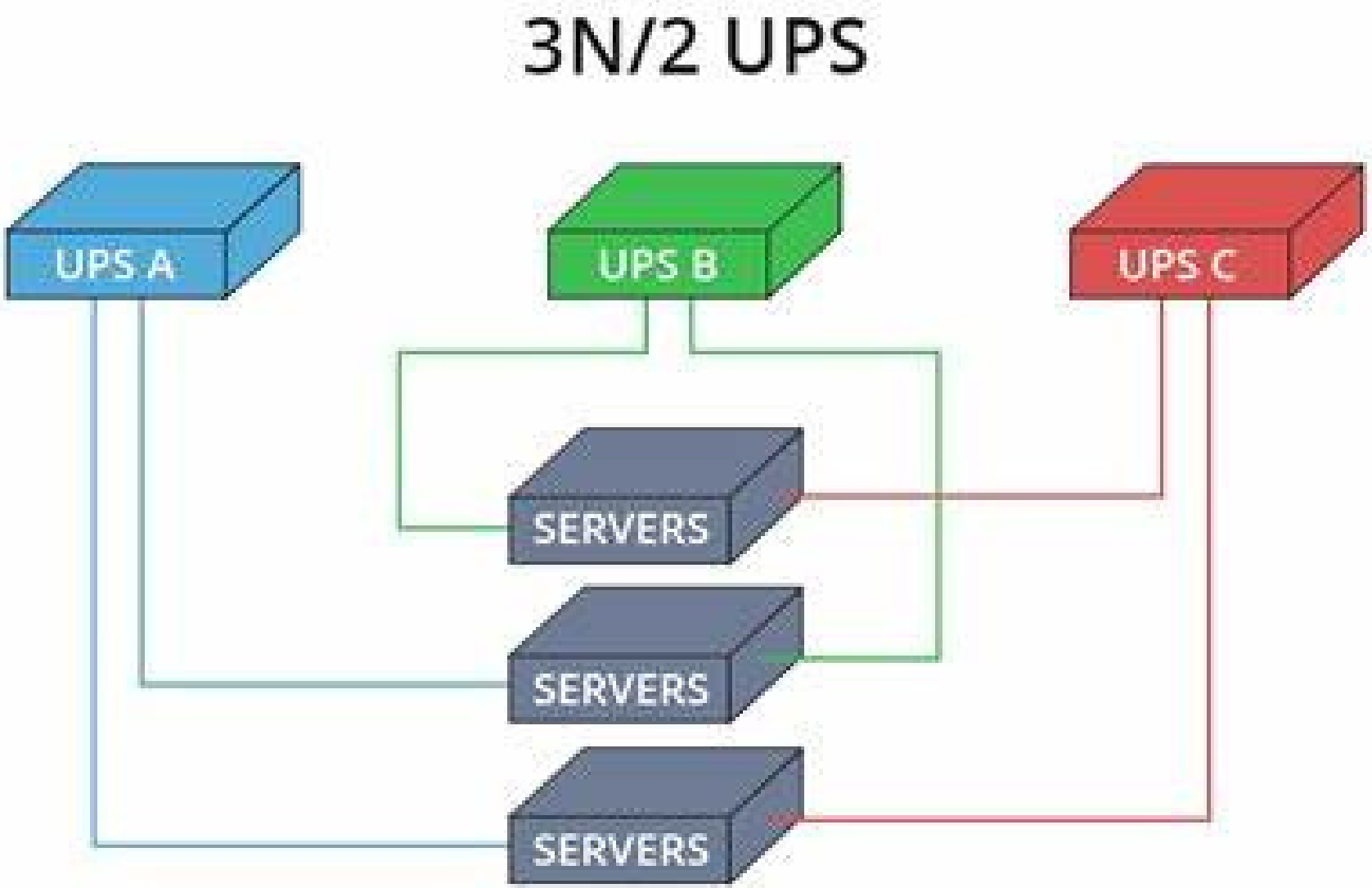
Problem Statement :

- Data loss or corruption can occur due to hardware failures, human errors, or malicious attacks, leading to loss of critical information and system downtime.

Solution :

- **Data Redundancy**:
- Duplicate or replicate data across multiple storage locations or servers.
- Ensure that if one copy of the data is lost or corrupted, another copy is available for retrieval.
- **Benefits**:
- Improves data availability and reliability by reducing the risk of data loss.
- Enhances fault tolerance and disaster recovery capabilities.
- Provides resilience against hardware failures and ensures continuity of operations.

Omkar
@omsrivastava



Note:

Data redundancy refers to the duplication of data within a system, which can enhance fault tolerance and data availability. Redundancy can be achieved through techniques like replication, where copies of data are stored across multiple nodes or servers. While increasing resilience, managing redundancy requires balancing storage costs and synchronization complexities.


Omkar
@omsrivastava

Helps job aspirants to crack FAANG product companies

SQL Or Non-SQL

Problem Statement : 

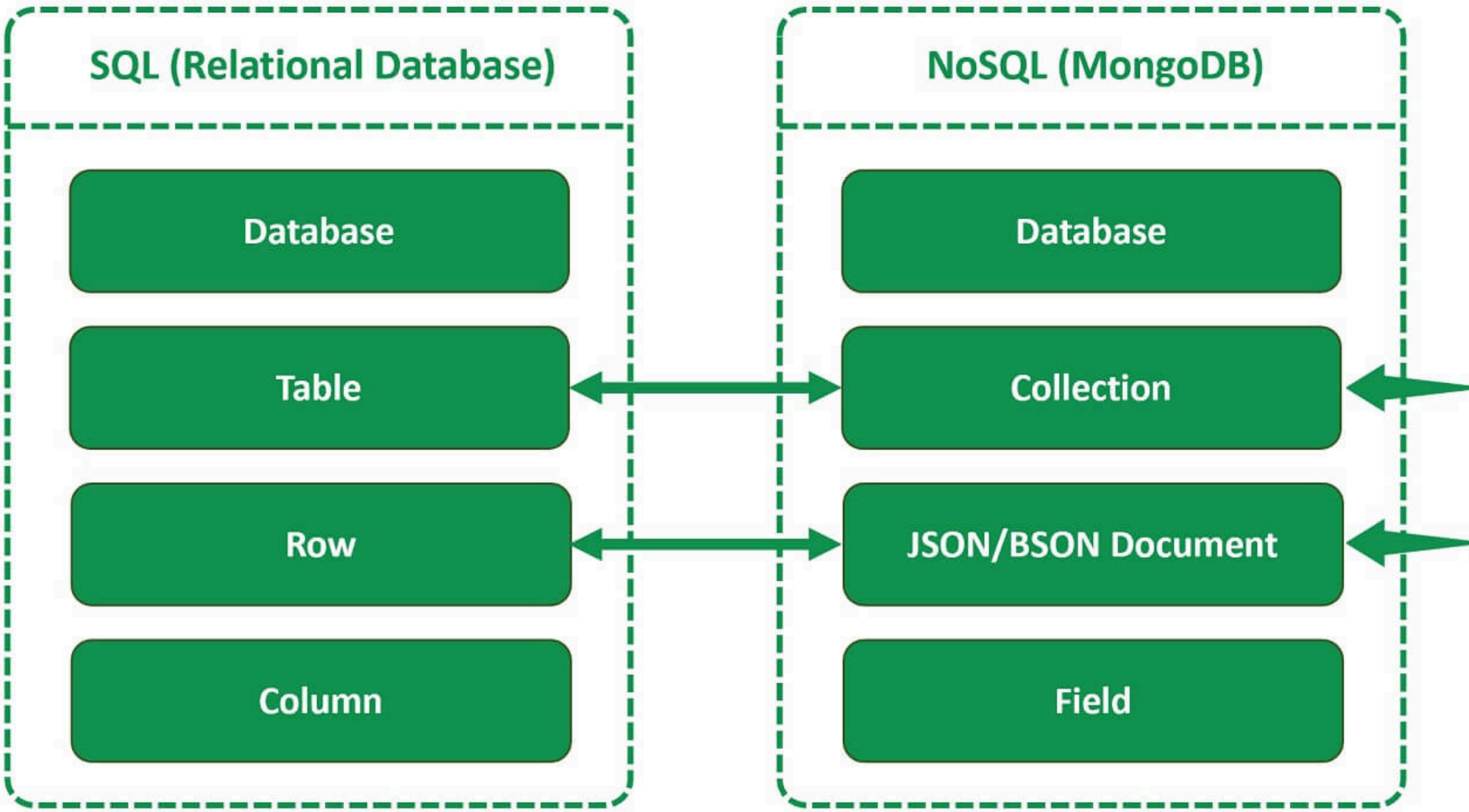
- Developers face a choice between SQL (relational databases) and NoSQL (non-relational databases) when designing a database system, each with its own benefits and drawbacks.

Solution : 

- **SQL (Relational Databases)**:
- Structured Query Language (SQL) databases use tabular schemas and are suitable for applications with structured data and complex queries.
- Examples: MySQL, PostgreSQL, Oracle.
- **NoSQL (Non-Relational Databases)**:
- NoSQL databases offer flexible schemas and are ideal for applications with unstructured data and scalability requirements.
- Examples: MongoDB, Cassandra, Redis.

Omkar
@omsrivastava

Helps job aspirants to crack FAANG product companies



SQL databases follow a structured, relational model, suitable for complex queries and structured data, ensuring ACID compliance. NoSQL databases offer flexibility and scalability for unstructured or semi-structured data, prioritizing performance and horizontal scalability over strict consistency. Choosing between them depends on data structure, scalability needs, and development preferences.

Devops free webinar, Link in description

Omkar
@omsrivastava

CAP Theorem

Problem Statement :

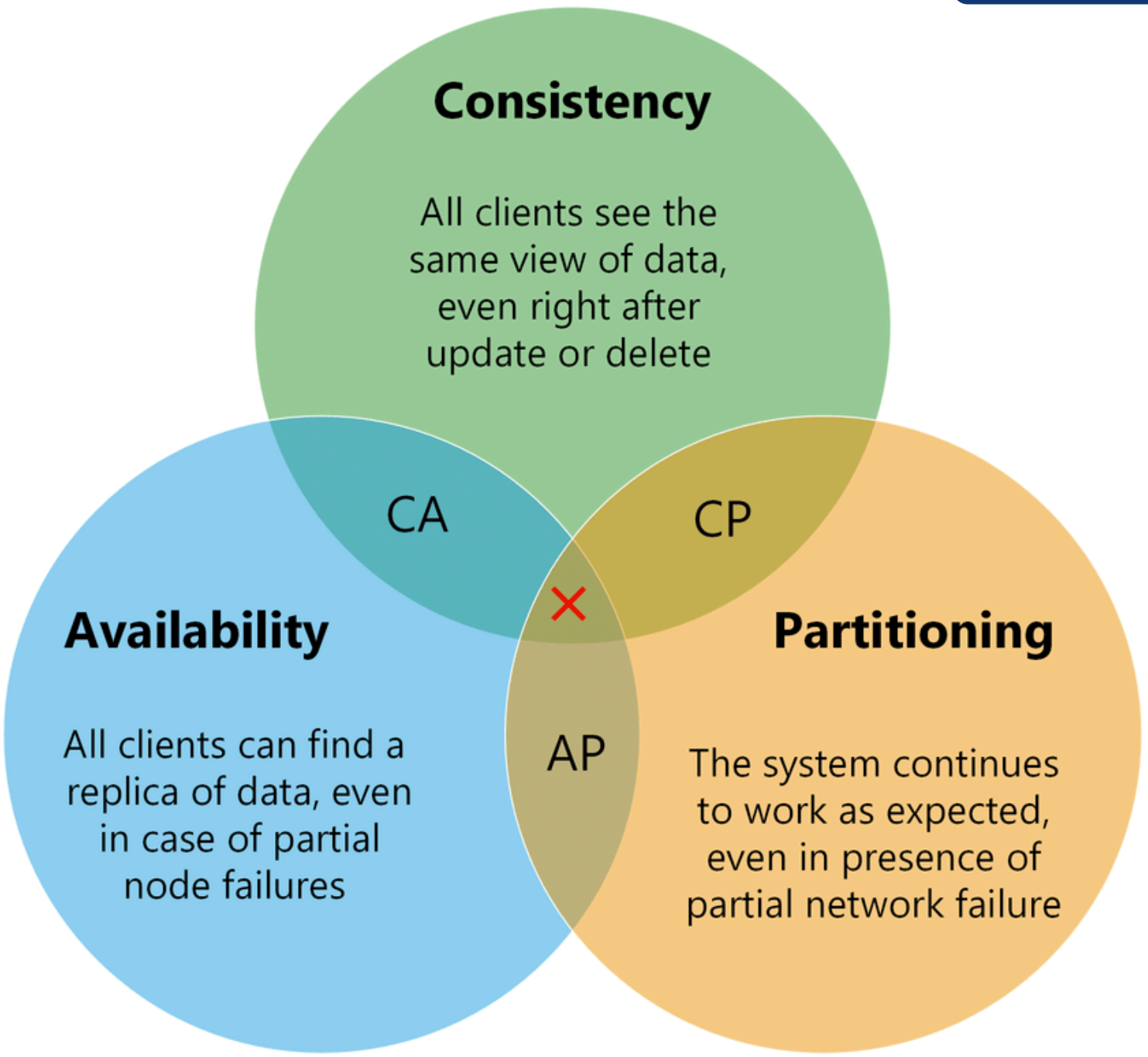
- Developers encounter the CAP theorem when designing distributed systems, which highlights the trade-offs between consistency, availability, and partition tolerance.

Solution :

- **CAP Theorem**:
- Consistency: All nodes have the same data at the same time.
- Availability: Every request receives a response, even with node failures.
- Partition Tolerance: The system operates despite network partitions.
- **Trade-offs**:
- Systems must prioritize between consistency, availability, and partition tolerance as achieving all three simultaneously is impossible.
- Choose the appropriate balance based on application requirements.

Omkar
@omsrivastava

Helps job aspirants to crack FAANG product companies



The CAP theorem asserts that in a distributed system, it's impossible to simultaneously guarantee all three of the following: Consistency (all nodes see the same data), Availability (every request receives a response), and Partition tolerance (system continues to operate despite network failures). System architects must prioritize between consistency and availability during network partitions, influencing database design and trade-offs in distributed systems.

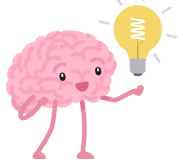
Omkar
@omsrivastava

Consistent Hashing

Problem Statement :

- In distributed systems, efficient data distribution and load balancing across multiple nodes are crucial. Traditional hashing methods can lead to uneven distribution and unnecessary data relocation when nodes are added or removed from the system

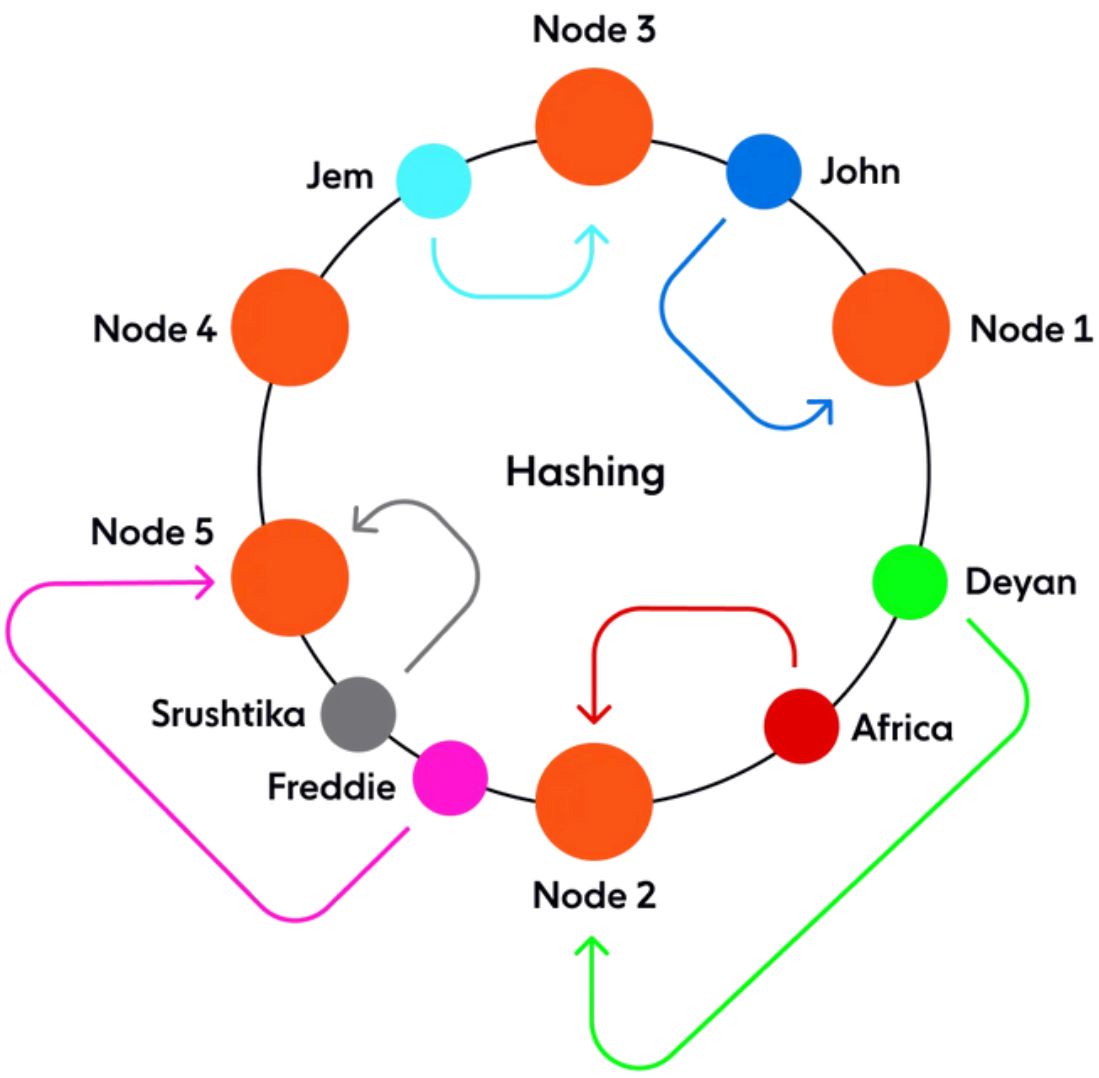
Solution :



- Consistent hashing efficiently distributes data in distributed systems, minimizing data relocation when nodes are added or removed.
- It maps keys and nodes to a common hash space, ensuring balanced load distribution.
- Each node is responsible for a range of hash values, reducing overhead and ensuring fault tolerance.
- Widely used in distributed caching, databases, and content delivery networks (CDNs) for optimized data distribution and load balancing.

Omkar
@omsrivastava

MAPPING IN THE HASHING



- **Note:**

- The CAP theorem states that a distributed system can't guarantee Consistency, Availability, and Partition tolerance all at once.
- During network issues, architects must choose between consistency and availability.
- This choice impacts database design and trade-offs in distributed systems.

Omkar
@omsrivastava

Message Queues

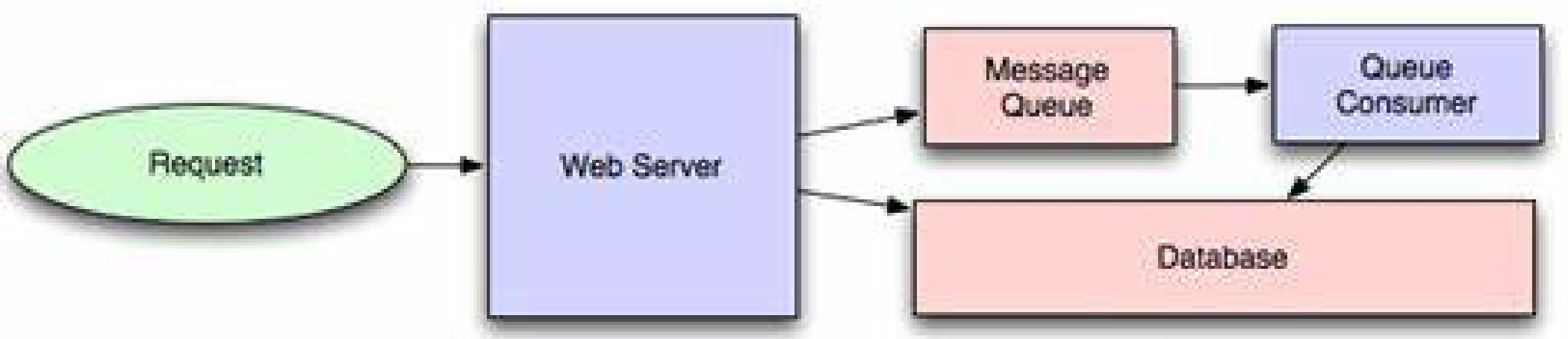
Problem Statement :

- In distributed systems, components often need to communicate asynchronously, decoupling producers and consumers to enhance scalability, reliability, and resilience.

Solution :

- Message queues facilitate asynchronous communication between components by storing messages until they are processed by consumers.
- Producers enqueue messages into the queue, and consumers dequeue and process them at their own pace.
- Message queues ensure reliable message delivery, even in the event of component failures or network issues.

Omkar
@omsrivastava



Note:

- Message queues enable asynchronous communication, storing messages until processed.
- They decouple sender and receiver, improving scalability, fault tolerance, and reliability.
- Common message queues include RabbitMQ, Kafka, and Amazon SQS, each with unique features.
- Using message queues is crucial for creating robust, loosely coupled distributed systems.

Omkar
@omsrivastava

Content Delivery Network

Problem Statement :

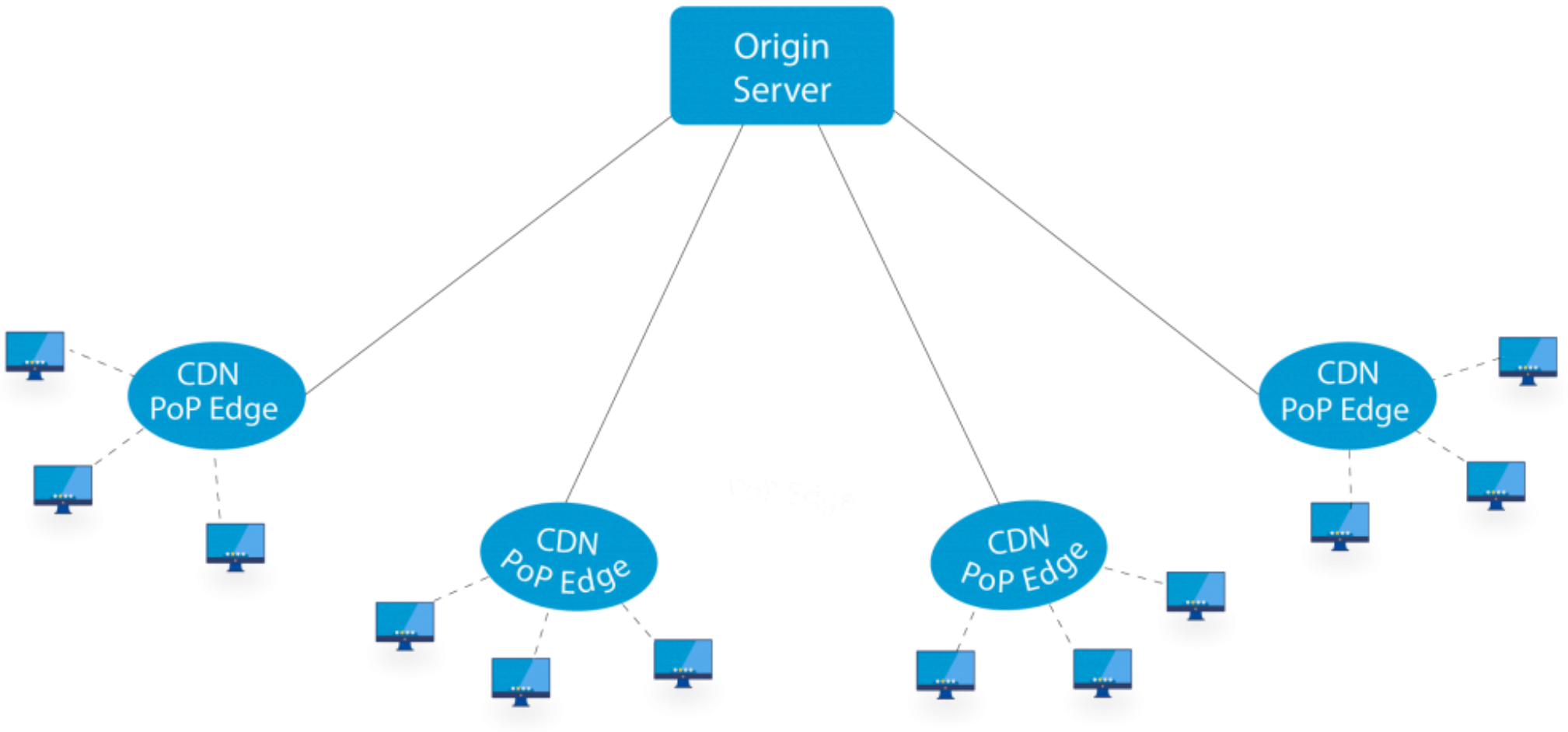
- Delivering content efficiently to users across the globe poses challenges such as latency, network congestion, and server overload.

Solution :

- CDNs distribute content across a network of servers strategically located in various geographic regions.
- When a user requests content, the CDN delivers it from the server closest to the user, minimizing latency and network congestion.
- CDNs cache content at edge locations, reducing the load on origin servers and improving scalability and reliability.

Omkar
@omsrivastava

Helps job aspirants to crack FAANG product companies



Note:

- CDNs are server networks worldwide that deliver web content efficiently.
- They cache content near users to reduce latency and improve website performance.
- CDNs decrease server load and speed up content delivery.
- They enhance user experience and protect against traffic spikes and DDoS attacks.
- Using CDNs is essential for optimized content distribution and smooth browsing globally.

Devops free webinar, Link in description