



Ashish Mishra

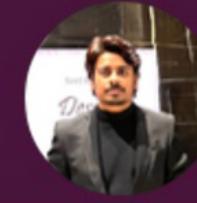
in @imashishmishra

# LET'S LEARN SCALABILITY

---

## SYSTEM DESIGN





Ashish Mishra

in @imashishmishra

# Introduction To Scalability

As the system grows, the performance starts to degrade unless we adapt it to deal with that growth. Scalability is the property of a system to handle a growing amount of load by adding resources to the system. A system that can continuously evolve to support a growing amount of work is scalable.

In this article, we will explore different ways a system can grow and common ways to make a system scalable.





Ashish Mishra

in @imashishmishra

# How Can a System Grow?

A system can grow in several dimensions:

## 1. Growth in User Base

More users start using the system, leading to an increased number of requests.

**Example:** A social media platform experiencing a surge in new users.

## 2. Growth in Features

Introducing new functionality to expand the system's capabilities.

**Example:** An e-commerce website adding support for a new payment method.

## 3. Growth in Data Volume

Growth in the amount of data the system stores and manages due to user activity or logging.

**Example:** A video streaming platform like YouTube storing more video content over time.

## 4. Growth in Complexity

The system's architecture evolves to accommodate new features, scale, or integrations, resulting in additional components and dependencies.

**Example:** A system that started as a simple application is broken into smaller, independent systems.



Ashish Mishra

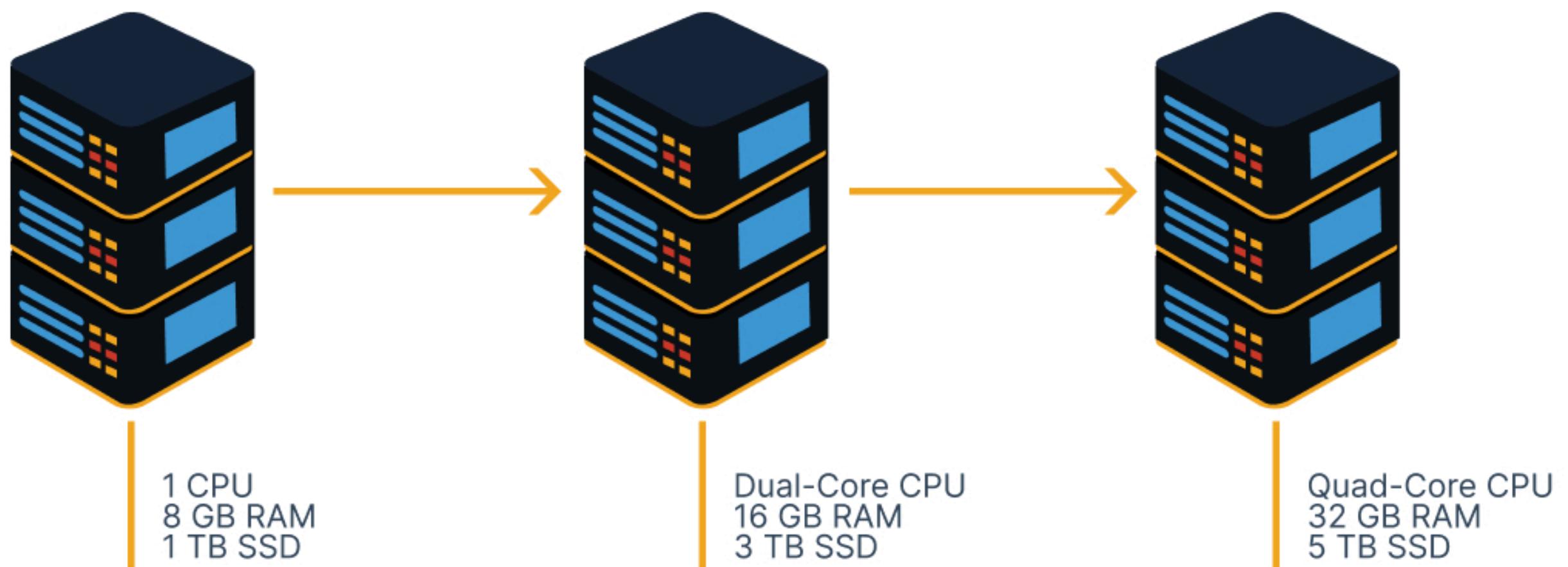
in @imashishmishra

# How to Scale a System?

## 1. Vertical Scaling (Scale up)

This means adding more power to your existing machines by upgrading the server with more RAM, faster CPUs, or additional storage. It's a good approach for simpler architectures but has limitations in how far you can go.

**Example:** A small e-commerce site initially running on a basic server starts experiencing slower performance as the number of daily visitors increases. To handle this growth, they decide to scale up by upgrading their server:



Ashish Mishra

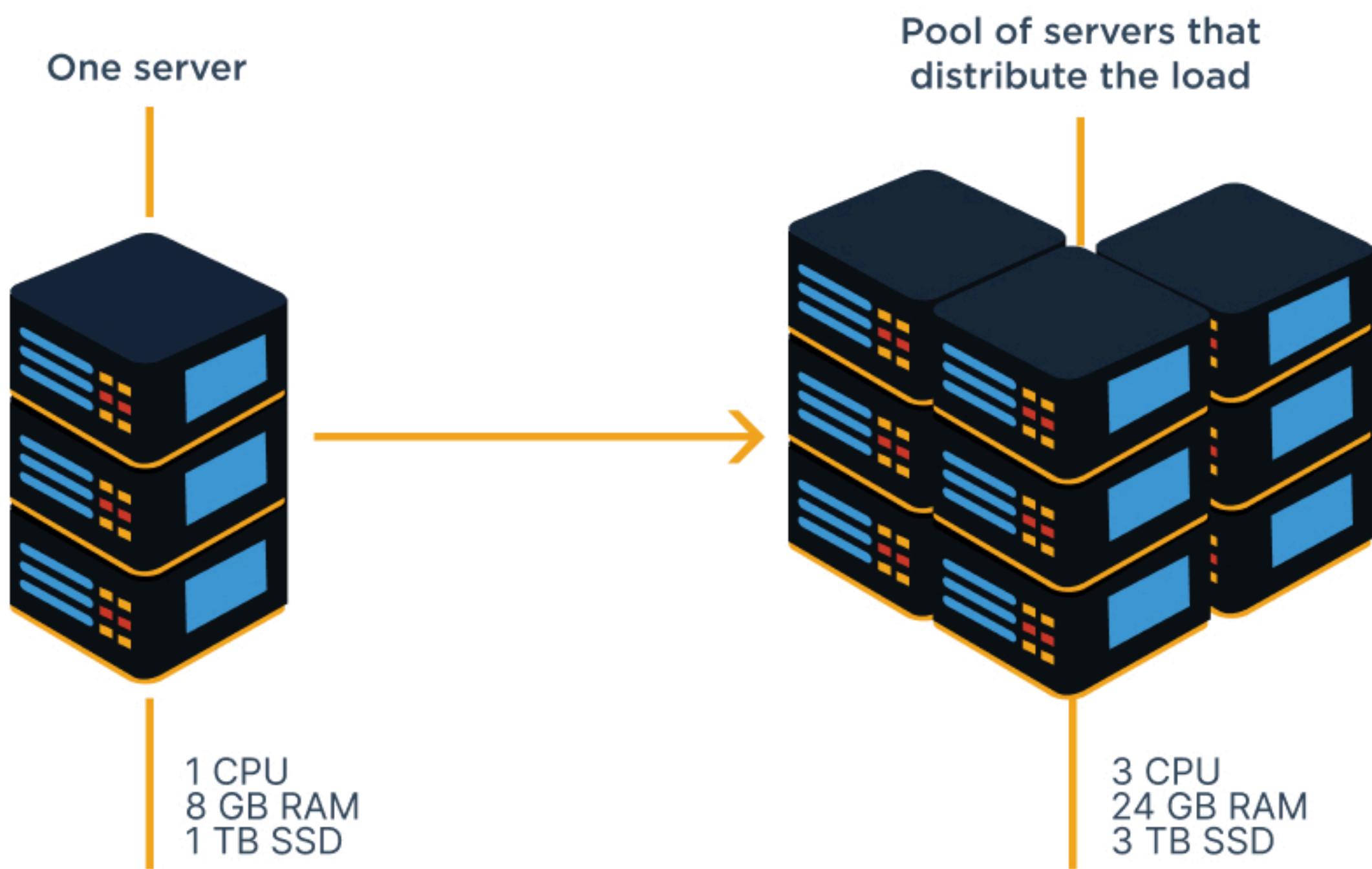
in @imashishmishra

# How to Scale a System?

## 2. Horizontal Scaling (Scale out)

This means adding more machines to your system to spread the workload across multiple servers. It's often considered the most effective way to scale for large systems.

**Example:** Netflix uses horizontal scaling for its streaming service, adding more servers to their clusters to handle the growing number of users and data traffic.



Ashish Mishra

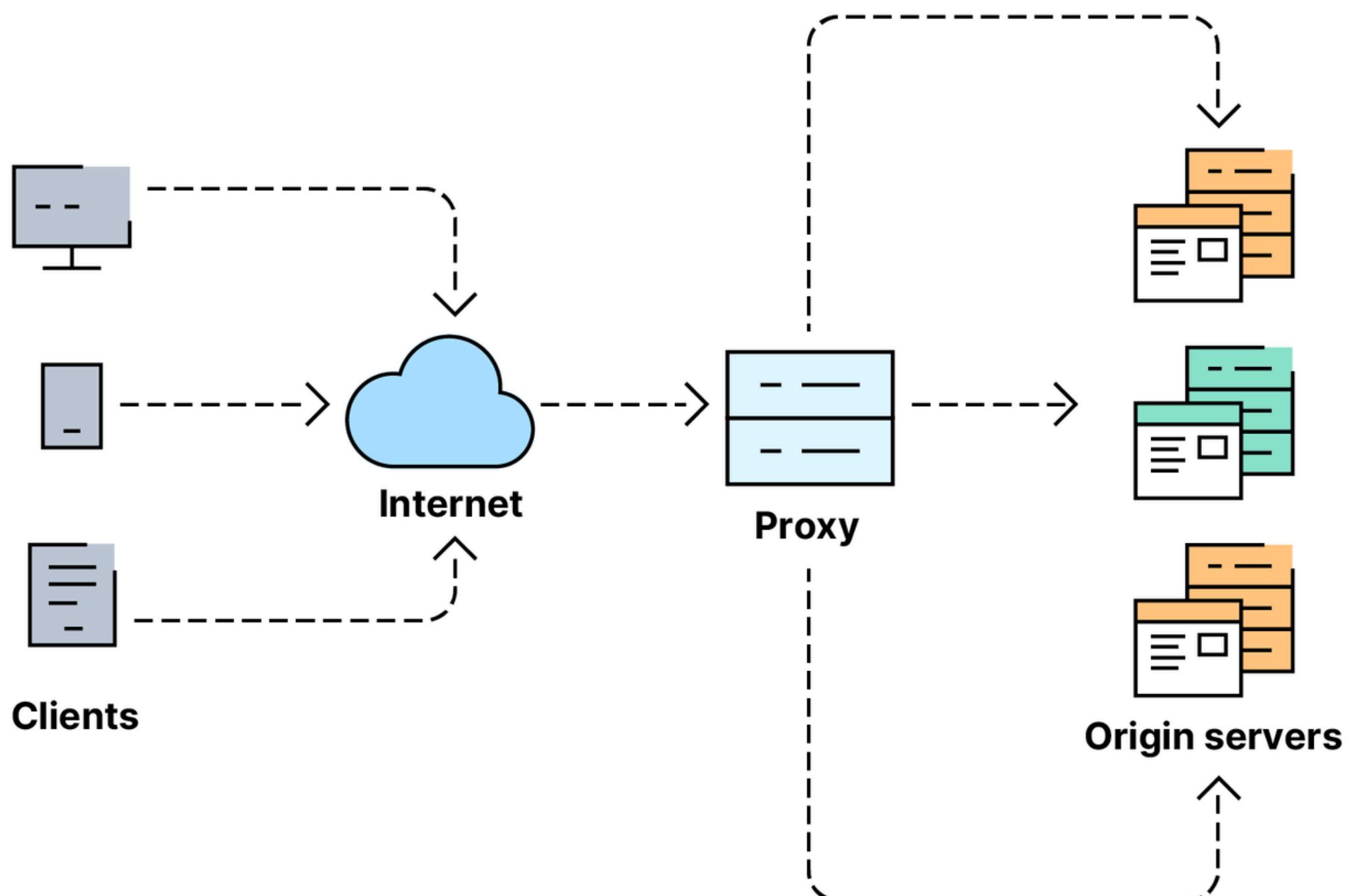
in @imashishmishra

# Best Practices For Scaling:

## 1. Load Balancing

Load balancing is the process of distributing traffic across multiple servers to ensure no single server becomes overwhelmed.

**Example:** Google employs load balancing extensively across its global infrastructure to distribute search queries and traffic evenly across its massive server farms.





Ashish Mishra

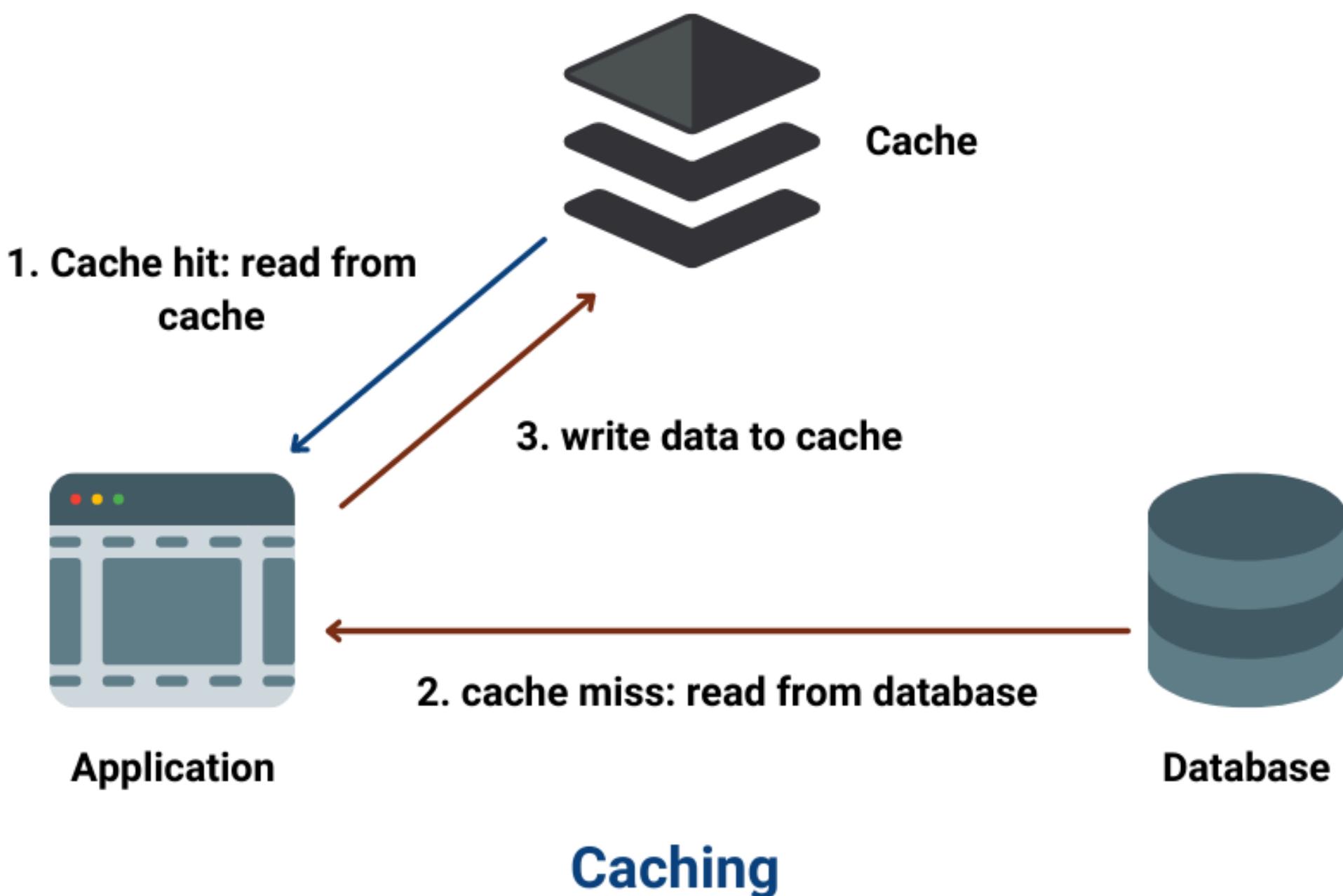
in @imashishmishra

# How to Scale a System?

## 2. Caching

Store frequently accessed data in-memory (like RAM) to reduce the load on the server or database. Implementing caching can dramatically improve response times.

**Example:** Reddit uses caching to store frequently accessed content like hot posts and comments so that they can be served quickly without querying the database each time





Ashish Mishra

in @imashishmishra

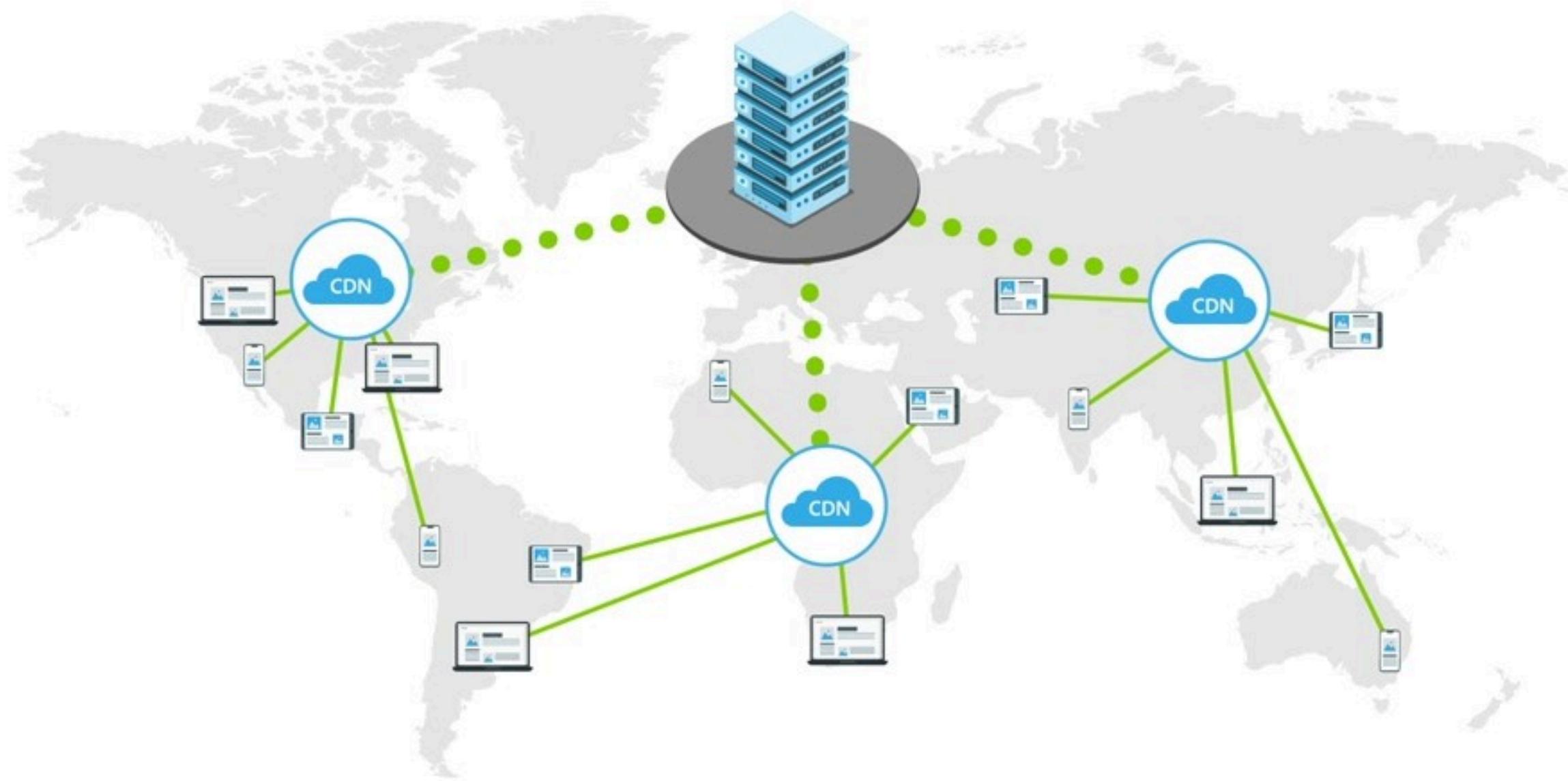
# How to Scale a System?

## 3. Content Delivery Networks (CDNs)

Distribute static assets (images, videos, etc.) closer to users.

This can reduce latency and result in faster load times.

**Example:** Cloudflare provides CDN services, speeding up website access for users worldwide by caching content in servers located close to users.



Ashish Mishra

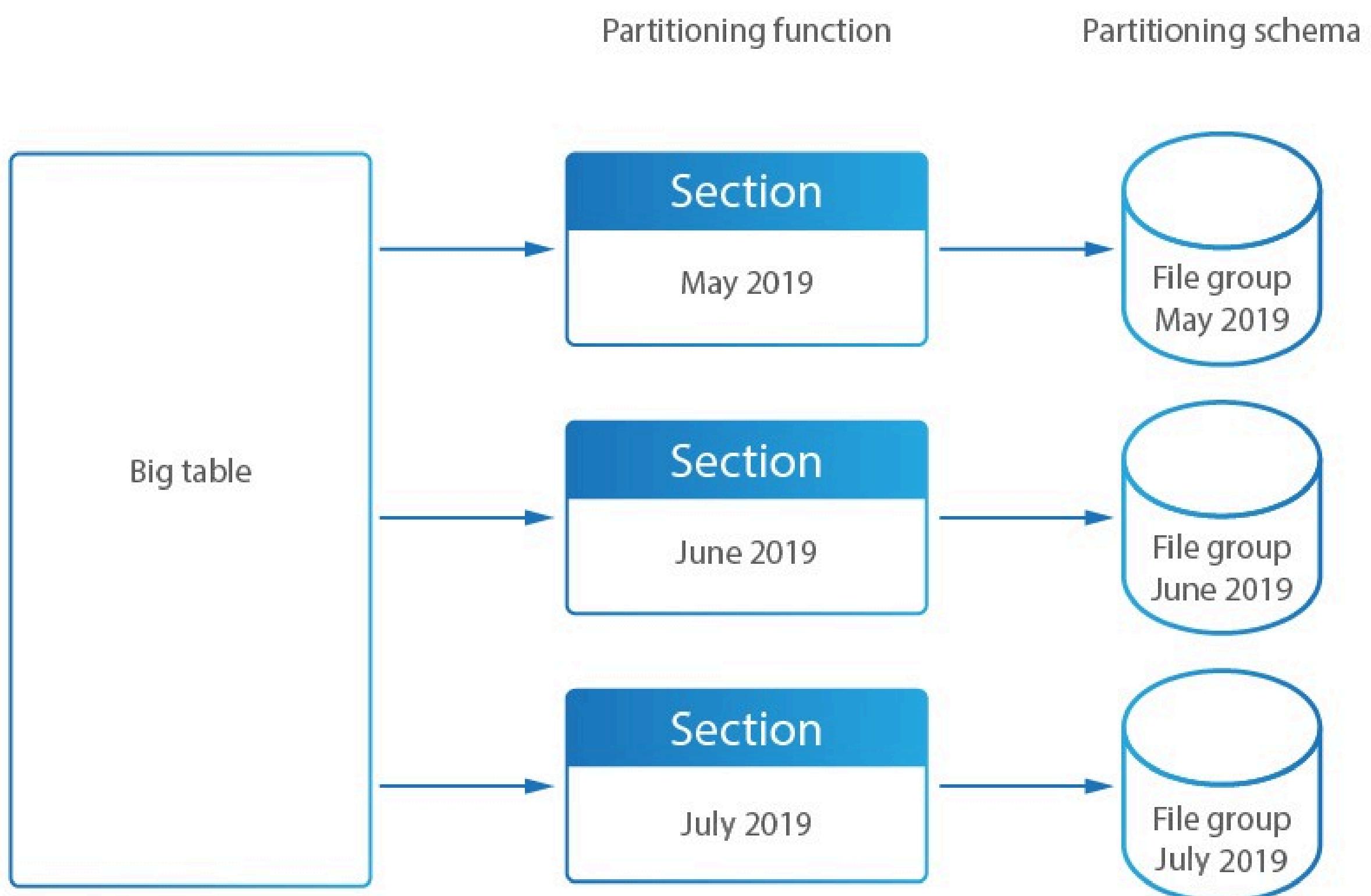
in @imashishmishra

# How to Scale a System?

## 4. Partitioning

Split data or functionality across multiple nodes/servers to distribute the workload and avoid bottlenecks.

**Example:** Amazon DynamoDB uses partitioning to distribute data and traffic for its NoSQL database service across many servers, ensuring fast performance and scalability.



Ashish Mishra

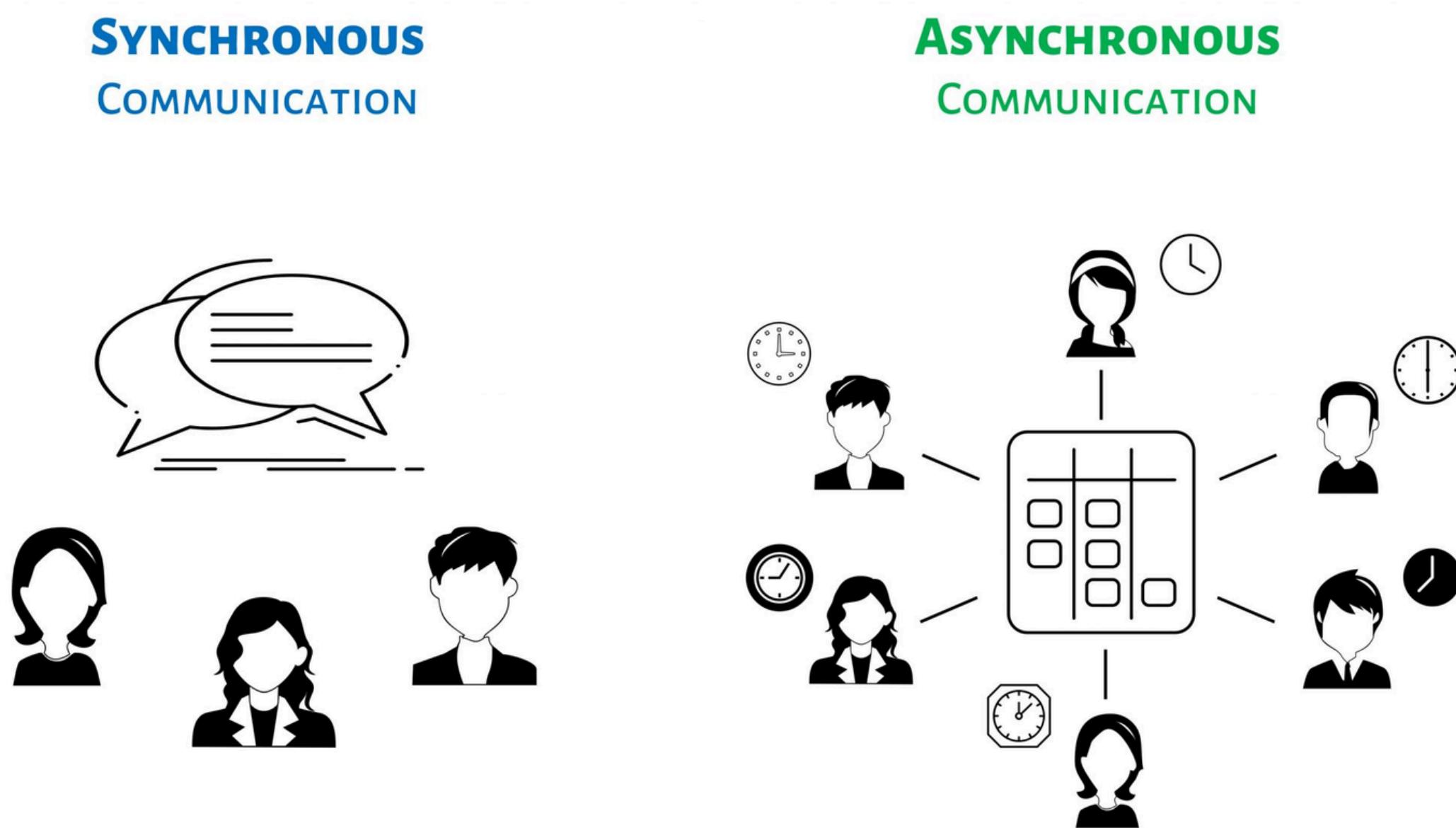
in @imashishmishra

# How to Scale a System?

## 5. Asynchronous Communication

Defer long-running or non-critical tasks to background queues or message brokers. This ensures your main application remains responsive to users.

**Example:** Slack uses asynchronous communication for messaging. When a message is sent, the sender's interface doesn't freeze; it continues to be responsive while the message is processed and delivered in the background.



Ashish Mishra

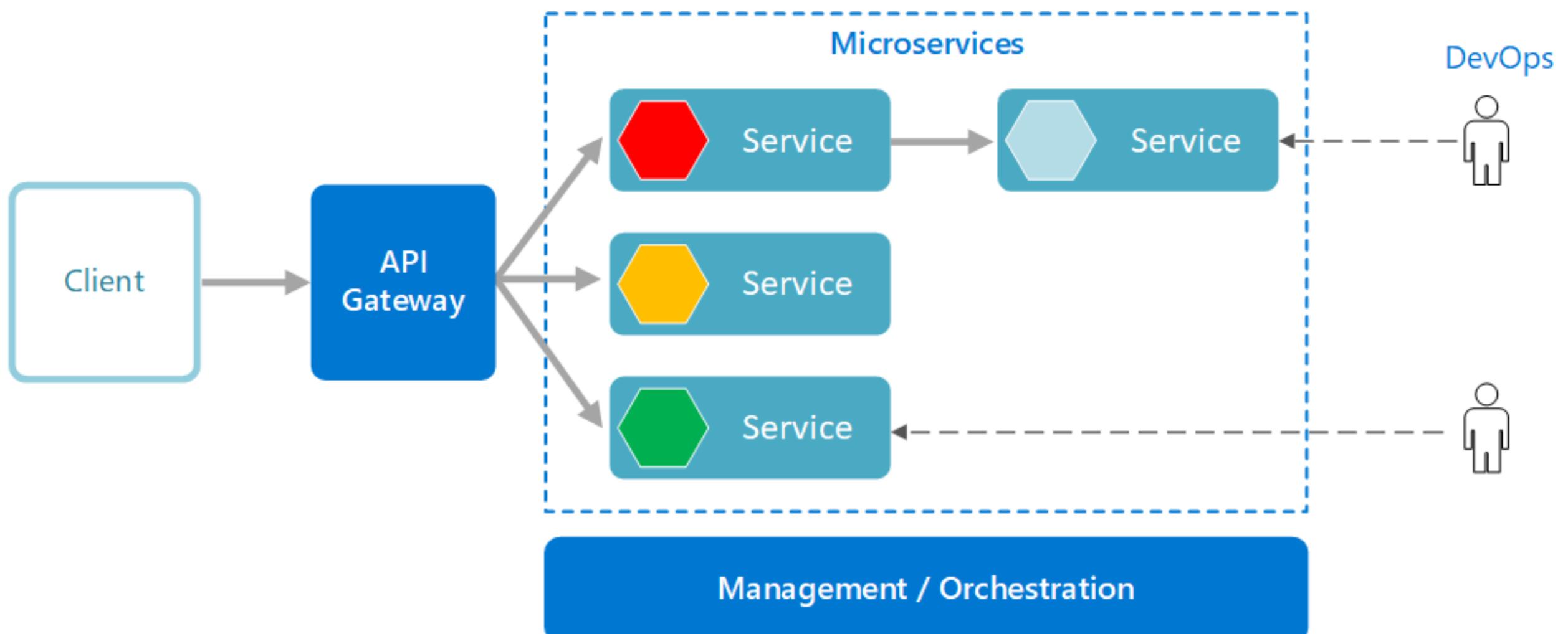
in @imashishmishra

# How to Scale a System?

## 6. Microservices Architecture

Break down your application into smaller, independent services that can be scaled independently. This improves resilience and allows teams to work on specific components in parallel.

**Example:** Uber has evolved its architecture into microservices to handle different functions like billing, notifications, and ride matching independently, allowing for efficient scaling and rapid development.



Ashish Mishra

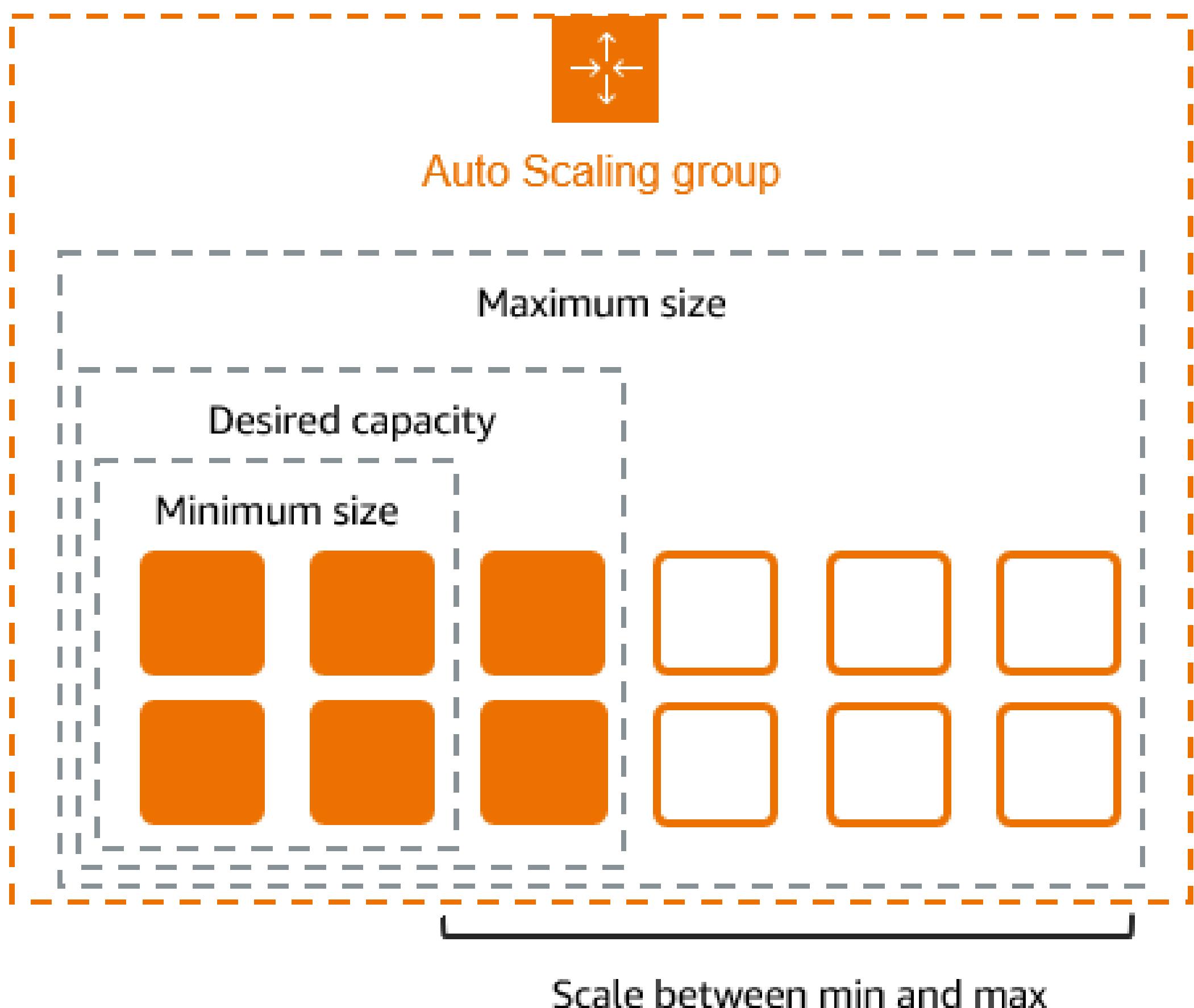
in @imashishmishra

# How to Scale a System?

## 7. Auto-Scaling

Automatically adjust the number of active servers based on the current load. This ensures that the system can handle spikes in traffic without manual intervention.

**Example:** AWS Auto Scaling monitors applications and automatically adjusts capacity to maintain steady, predictable performance at the lowest possible cost.



Ashish Mishra

in @imashishmishra

# How to Scale a System?

## 8. Multi-region Deployment

Deploy the application in multiple data centers or cloud regions to reduce latency and improve redundancy.

Example: Spotify uses multi-region deployments to ensure their music streaming service remains highly available and responsive to users all over the world, regardless of where they are located.

