### ****Scalability in System Design****

Scalability is the system’s ability to handle increasing workloads efficiently by adding resources. It ensures that performance remains stable as demand grows.

### ****Types of Scalability****

1. **Vertical Scaling (Scaling Up)**
   * Increasing the power (CPU, RAM, SSD) of a single server.
   * Example: Upgrading a database server from 16GB RAM to 64GB RAM.
   * **When to Use:**
     + When your system has a single database or application server.
     + When you need a quick performance boost without architectural changes.
     + When the workload is CPU/memory-bound (e.g., analytics, AI workloads).
2. **Horizontal Scaling (Scaling Out)**
   * Adding more servers to distribute the load.
   * Example: Instead of upgrading one server, adding multiple servers behind a load balancer.
   * **When to Use:**
     + When the system has high traffic and needs to support millions of users.
     + When redundancy and fault tolerance are required.
     + When workloads can be distributed across multiple machines.

### ****Scalability Example: Netflix****

#### ****1️⃣ Initial Phase (Small Scale)****

* Netflix started as a DVD rental service with a small monolithic application.
* A single relational database (SQL) stored user accounts and movie data.
* A single server handled user requests.

#### ****2️⃣ Growth Phase (Performance Bottlenecks)****

* As streaming gained popularity, millions of users started watching movies.
* The database became slow, leading to high response times.
* A single server couldn’t handle traffic spikes, leading to frequent downtime.

#### ****3️⃣ Scaling the System****

✅ **Horizontal Scaling:**

* Instead of using one large SQL database, Netflix sharded data across multiple databases.
* They replicated services across multiple AWS regions to improve performance.

✅ **Microservices Architecture:**

* They broke their monolithic app into smaller services:
  + **User Authentication Service**
  + **Recommendation Engine**
  + **Streaming Service**
* This allowed independent scaling of each service.

✅ **Content Delivery Network (CDN):**

* To reduce latency, Netflix used CDNs like Open Connect.
* This allowed users to stream videos from nearby locations instead of a central server.

#### ****4️⃣ Final Scalable System****

* Uses **AWS Auto-Scaling** to handle dynamic traffic.
* **Load Balancers** distribute traffic across multiple instances.
* **Redis and Memcached** handle frequent database queries efficiently.
* **Event-driven architecture** processes user requests asynchronously.

### ****Key Takeaways****

* **Use Vertical Scaling** when you need an immediate performance boost without changing the architecture.
* **Use Horizontal Scaling** when handling large-scale users, requiring fault tolerance and better load distribution.
* **Combine caching, microservices, and CDNs** for a highly scalable and efficient system.