**Throughput in System Design**

**📌 What is Throughput?**

🔹 **Throughput** is the number of requests a system can process per second.  
🔹 It measures **system efficiency** and is typically expressed in **requests per second (RPS)** or **transactions per second (TPS)**.

**🛠️ Example: Amazon Order Processing**

1️⃣ A customer **places an order** on Amazon.  
2️⃣ The request is sent to **Amazon’s backend**, which processes payments, updates inventory, and generates an invoice.  
3️⃣ Amazon can handle **millions of orders per second**, thanks to high-throughput systems.

✅ **High Throughput** → Amazon processes **millions of orders per second** without delay.  
🚫 **Low Throughput** → If Amazon's servers slow down, orders take **minutes to process**, leading to bad user experience.

**🔴 Why Does Low Throughput Occur?**

🚫 **Single Point of Failure** → If a single database or service is overloaded, requests queue up.  
🚫 **Inefficient Database Queries** → Full-table scans and locks slow down request processing.  
🚫 **Network Bottlenecks** → If the bandwidth is low, requests cannot be handled efficiently.  
🚫 **Synchronous Processing** → If every step waits for the previous one to finish, the system gets slower.

**🟢 How to Increase Throughput?**

**1️⃣ Load Balancing (Distribute Requests)**

✅ Distributes traffic **across multiple servers** to avoid overload.  
✅ Example: **Google Search** distributes search queries to thousands of servers.

**2️⃣ Horizontal Scaling (More Servers)**

✅ Add **more servers** to process requests **in parallel**.  
✅ Example: **WhatsApp** adds more servers when user traffic spikes.

**3️⃣ Asynchronous Processing (Don’t Wait)**

✅ **Process background tasks separately** to avoid blocking the main request.  
✅ Example: **Uber** processes ride bookings asynchronously while updating driver locations in real time.

**4️⃣ Database Optimization**

✅ Use **Sharding** to split data across multiple databases.  
✅ Example: **Facebook** shards user data to different databases for high throughput.

**5️⃣ Caching (Reduce Load on Backend)**

✅ **Store frequently accessed data** in a cache (Redis, Memcached).  
✅ Example: **Twitter** caches user feeds to serve millions of requests quickly.

**📌 Throughput vs. Latency**

| **Metric** | **Definition** | **Goal** |
| --- | --- | --- |
| **Throughput** | Number of requests processed per second | Increase system capacity |
| **Latency** | Time taken to process a request | Reduce response time |

✅ **Low Latency ≠ High Throughput**  
🔹 Example: A system **responding in 1ms** but handling **only 10 requests per second** has low latency but low throughput.  
🔹 A system handling **100K requests per second** but taking **500ms per request** has high throughput but high latency.

**🛠️ Real-World Examples**

| **Company** | **Use Case** | **Throughput Optimization** |
| --- | --- | --- |
| **Amazon** | Order processing | Async processing, caching, load balancing |
| **YouTube** | Video streaming | CDN, caching, high-bandwidth networks |
| **WhatsApp** | Messaging | Distributed servers, database sharding |
| **Stock Trading Apps** | High-frequency trades | Low-latency, high-throughput systems |

**🛠️ When to Optimize for Throughput?**

✅ When handling **millions of concurrent users** (e.g., Twitter, Facebook).  
✅ When a **high volume of transactions** needs processing (e.g., banking, stock trading).  
✅ When **real-time systems** need to scale (e.g., ride-hailing apps like Uber).