System Design Interview Cheatsheet

A comprehensive guide to help you prepare for system design interviews. Covers key concepts, common architectures, and practical tips.

Core Concepts

Scalability

Vertical Scaling (Scale Up)	Increasing the resources of a single server (e.g., CPU, RAM). **Pros:** Simple **Cons:** Hardware limits, single point of failure.
Horizontal Scaling (Scale Out)	Adding more servers to the system. **Pros:** High availability, fault tolerance **Cons:** Complexity, data consistency challenges.
Tradeoffs	Consider which type of scaling is appropriate for your application's needs and constraints.

Availability

Definition	The percentage of time a system is operational and accessible.
High Availability (HA)	Designing systems to minimize downtime and ensure continuous operation.
Strategies	Replication, redundancy, failover mechanisms.

Consistency

Definition	Ensuring that all clients see the same data at the same time.
CAP Theorem	In a distributed system, it's impossible to simultaneously guarantee Consistency, Availability, and Partition Tolerance. You must choose two.
Eventual Consistency	A weaker guarantee where data will be consistent eventually, but not necessarily immediately.

Key Components & Technologies

Load Balancers

Purpose	Distribute incoming network traffic across multiple servers.
Types	Layer 4 (TCP) and Layer 7 (HTTP/HTTPS).
Algorithms	Round Robin, Least Connections, IP Hash.

Databases

SQL (Relational)	MySQL, PostgreSQL. **Pros:** ACID properties, structured data. **Cons:** Scalability challenges.
NoSQL (Non-Relational)	MongoDB, Cassandra, Redis. **Pros:** Scalability, flexibility. **Cons:** Eventual consistency, unstructured data.
Considerations	Choose the database type based on data model, consistency requirements, and scalability needs.

Caching

Purpose	Store frequently accessed data to reduce latency and improve performance.
Types	Client-side, CDN, server-side (e.g., Redis, Memcached).
Strategies	Write-through, write-back, cache invalidation.

Common System Architectures

Microservices Architecture

Description	An architectural style that structures an application as a collection of small, autonomous services, modeled around a business domain.
Benefits	Independent deployment, scalability, technology diversity.
Challenges	Complexity, distributed tracing, inter-service communication.

Message Queues

Purpose	Asynchronous communication between services. Decouples services and provides buffering.
Examples	RabbitMQ, Kafka.
Use Cases	Background processing, task queues, event-driven architectures.

Content Delivery Network (CDN)

Description	A distributed network of servers that delivers content to users based on their geographic location.
Benefits	Reduced latency, improved performance, decreased load on origin servers.
Use Cases	Serving static assets (images, videos, CSS, JS).

Interview Tips

Clarify Requirements

Always start by clarifying the requirements and constraints of the system. Ask questions to understand the scope and goals.

Examples:

- How many users?
- What are the read/write ratios?
- What are the latency requirements?
- What are the storage requirements?

Think Out Loud

Explain your thought process and reasoning as you design the system. This allows the interviewer to understand your approach and provide feedback.

Discuss the trade-offs of different design choices and justify your decisions.

Focus on Bottlenecks

Identify potential bottlenecks in the system (e.g., database, network) and discuss how to address them.

Propose solutions such as caching, load balancing, and database sharding to mitigate bottlenecks.