System Design

cheat sheet

Follow @rajivranjan00

01

Distributed transaction has to be handled

Consider 2-phase commit or saga patterns

02

High Availability and Consistency Trade-Off

Consider CAP theorem

03

Decouple read & write for high performant system

Consider using CQRS

04

To build reliable & robust system

Consider logging, monitoring & observability

05

Concurrency in distributed system

Consider distributed (optimistic) lock

06

Need coordination & synchronization in distributed system

Use Zookeeper

07

Detection failure in Distributed Systems

Implement a Heartbeat

08

Need ACID Compliant Database

Go for Relational / SQL Database

09

Have unstructured data & doesn't require ACID properties

Go for NoSQL Database

10

If database need to be scaled

Implement Database Sharding & Partition

11

High-Performing Database Queries Optimization

Use Database Indexes

12

If the system has a single point of failure

Implement Redundancy

13

For Fault-Tolerance and Durability

Implement Data Replication

14

To distribute network traffic to get high availability, performance, & throughput

Use Load Balancer

15

Over fetching & under fetching of data

Consider Graph QL

16

For a Low Latency or Read-Heavy System

Consider using a Cache

17

If we are dealing with a write-heavy system

Use Message Queues for async processing

18

High performant search and query

Consider search index, tries, or elastic search

19

Static content delivery with enhanced Performance, Scalability & Security

Consider using a CDN

20

Better scalability and fault tolerance system

Implement Horizontal Scaling

21

If need to have bulk job processing

Consider Batch Processing and Message Queues

22

Preventing from DOS Attacks

Use a Rate Limiter

23

Centralized entry point for managing and securing API requests

Use an API Gateway

24

Need real-time or streaming system

Consider Server Sent Events

25

Bi-directional and real-time communication between a client and server

Use WebSocket

26

If data integrity need to be ensured

Use Checksum Algorithm

27

Peer-peer communication, Decentralized Data Transfer and eventual consistency

Consider Gossip Protocol

28

Efficient and scalable approach to data distribution in distributed systems

Implement Consistent Hashing

29

Handling Large Data in Network Requests

Implement Pagination

30

To handle traffic spikes on-demand

Implement Autoscaling