



Arif Alam • 3rd+

Making Data Science and AI Accessible to All | Educator | Storyteller | Building Data Science Re...
3d • 🌐

[Follow](#)

Top 20 System Design Concepts You MUST Know

You want to build something that handles real-world scale?

Not just code that runs -- but systems that survive.

Here's what you need in your toolbox:

► 1. Load Balancing

Manage traffic smartly across multiple servers. Prevent overload. Ensure uptime.

► 2. Caching

Because speed matters. Whether it's Redis or a CDN — reduce latency, save cost.

► 3. Database Sharding

Break the DB into pieces. Handle large datasets efficiently.

► 4. Replication

Make your data fault-tolerant. Think failover, think backups.

► 5. CAP Theorem

Understand why you can't have Consistency, Availability, and Partition Tolerance all at once.

► 6. Message Queues

Asynchronous processing. Decouple services. Think Kafka, RabbitMQ.

► 7. Rate Limiting

Protect your APIs. Control abuse. Prioritize key users.

► 8. CDN (Content Delivery Network)

Push content closer to the user. Lightning-fast experience worldwide.

► 9. Heartbeat and Health Checks

Know if your services are alive before your users do.

► 10. Eventual Consistency

Perfect consistency is a myth. Learn to design for “eventual” correctness.

► 11. Data Partitioning

Split large datasets for performance. Use horizontal scaling wisely.

► 12. Leader Election

One leader, many followers. Used in distributed coordination.

► 13. Circuit Breakers

Fail gracefully. Don't let one broken service crash the system.

► 14. Observability

Metrics, logs, traces. Monitor everything.

► 15. API Gateway

Entry point for your microservices. Handle routing, auth, throttling.

► 16. Authentication vs Authorization

They are not the same. Know who the user is vs what they can do.

► 17. Scalability

Vertical vs horizontal scaling. Know when to use each.

► 18. Statelessness

Design services that don't rely on memory of previous calls.

► 19. Database Indexing


Faster lookups. Smarter queries. But index only what's needed.

► 20. Backpressure

When consumers are slower than producers apply resistance or buffering.

 **400+ Data Science Resources:** <https://lnkd.in/gv9yvfd>

 **Premium Data Science Interview Resources :** <https://lnkd.in/gPrWQ8is>

 **Python Data Science Library:** <https://lnkd.in/gHSDtsmA>

 **45+ Mathematics Books Every Data Scientist Needs:** <https://lnkd.in/ghBXQfPc>

Join What's app channel for jobs updates: https://lnkd.in/gu8_ERtK

 / [ByteByteGo](#)

Top 20 System Design Concepts Every Developer Should Know

1 Load Balancing



Distributes traffic across multiple servers for reliability

2 Caching



Stores frequent data in memory for faster access

3 Database Sharding



Splits databases to handle large-scale data growth

4 Replication



Copies data across replicas for availability and fault-tolerance

5 CAP Theorem



Trade-off between consistency, availability, and partition tolerance

6 Consistent Hashing



Distributes traffic across multiple servers for reliability

7 Message Queues



Decouples services using async event-driven architecture

8 Rate Limiting



Controls request frequency to prevent system overload

9 API Gateway



Centralized entry point for routing API requests

10 Microservices



Breaks systems into independent, loosely coupled services

11 Service Discovery



Locates services dynamically in distributed systems

12 CDNS



Delivers content from edge servers for speed

13 DB Indexing



Speeds up queries by indexing important fields

14 Partitioning



Divides data across nodes for scalability and performance

15 Eventual Consistency



Guarantees consistency over time in distributed databases

16 WebSockets



Ensure bidirectional communication for live updates

17 Scalability



Increases capacity by upgrading or adding machines

18 Fault Tolerance



Ensures system availability during hardware/software failures

19 Monitoring



Tracks metrics and logs to understand system health

20 AuthN & AuthZ



Controls user access and verifies identity securely