Gmail

Shanmugham Sundaram <shanmughams@gmail.com>

23 System Design Interview Tips

Saurabh Dashora <systemcodex@substack.com> Tue, Apr 16, 2024 at 2:14 PM
Reply-To: Saurabh Dashora
<reply+2dhkm0&edw13&&34fecb5f022960a2923dcaff4db0575e06b264ad7e3bc47894faaa3cc864a7bf@mg1.substack.com>
To: shanmughams@gmail.com

Forwarded this email? Subscribe here for more

23 System Design Interview Tips

Principles to consider while designing a system.

SAURABH DASHORA
APR 16

READ IN APP ↗

System Design interviews are tricky to navigate.

While the concepts may not be super-difficult to understand, their application in a particular context becomes important. These concepts are like a box of Lego blocks and the way you combine the blocks can have a huge impact on what you end up building.

Also, unlike real projects, you don't get multiple days to think and discuss about a system's design.

This is where some basic principles can be pretty useful to navigate a system design interview successfully.

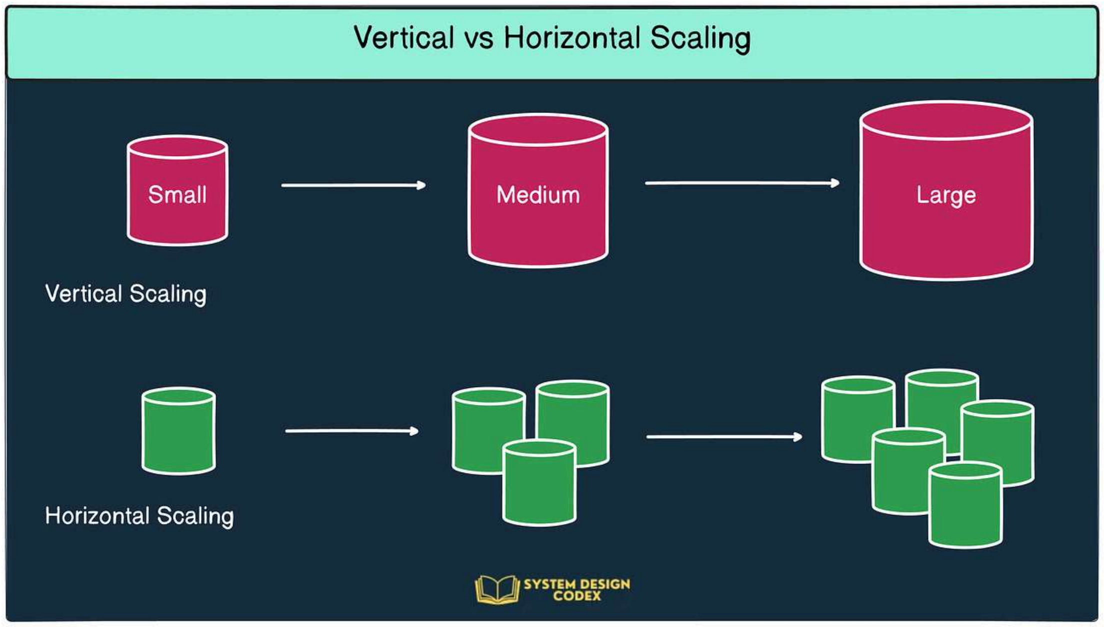
Here are 23 such principles that I've found to be useful in system design interviews as well as in building actual systems.

1 - Vertical Scaling followed by Horizontal

If you need to scale an individual component, it's mostly better to stick to vertical scaling first.

In most scenarios, vertical scaling can take you quite far in terms of demand.

Look at horizontal scaling after you reach a certain scale where allocating more resources to a single server is no longer cost-effective.



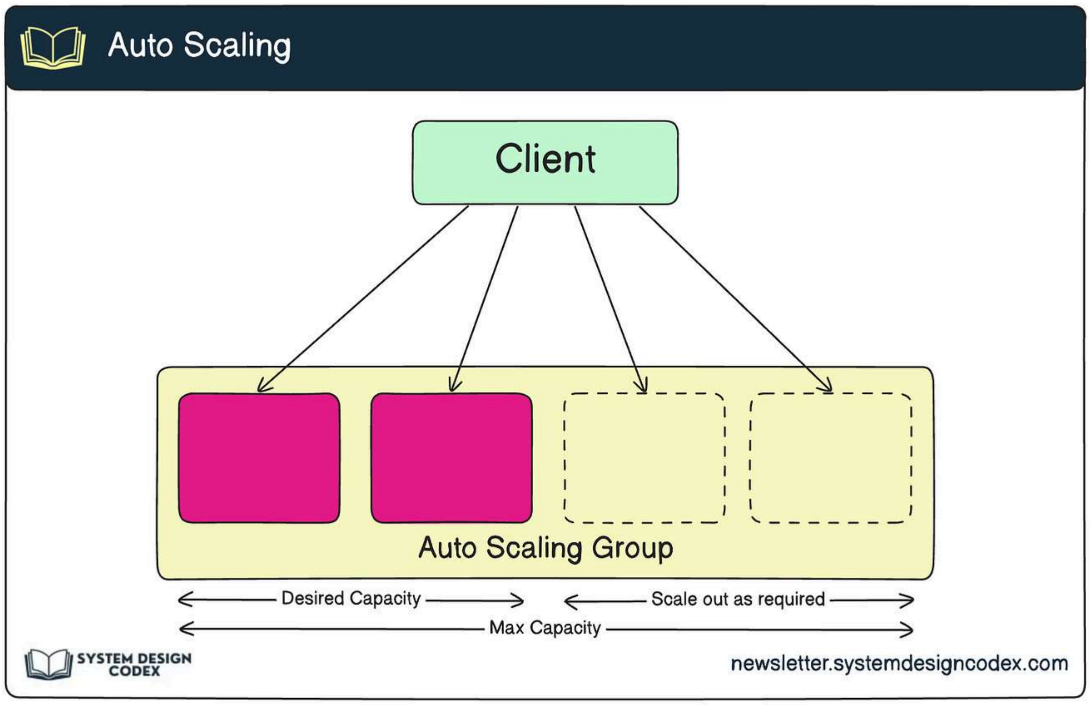
You can play around with the diagram on Eraser.io

2 - Autoscaling for Traffic Spikes

Once you scale things horizontally, you still don’t want to over-allocate resources and lose money in the process (unless it’s a super-critical component).

You ideally want to scale out based on demand and this is where autoscaling helps.

Most cloud platforms provide autoscaling capabilities based on certain rules.

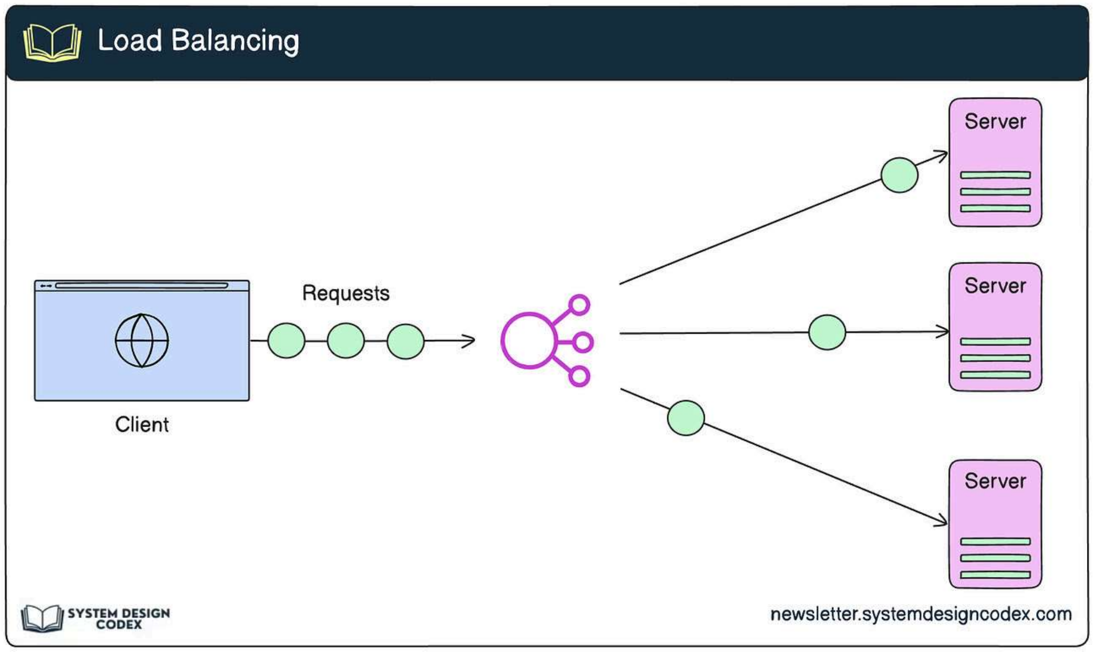


You can play around with the diagram on Eraser.io

3 - Load Balancer for High Availability

If your application requires high availability with a good performance, consider using a load balancer.

It will not only help you scale out a component to multiple instances but also make efficient use of those resources.



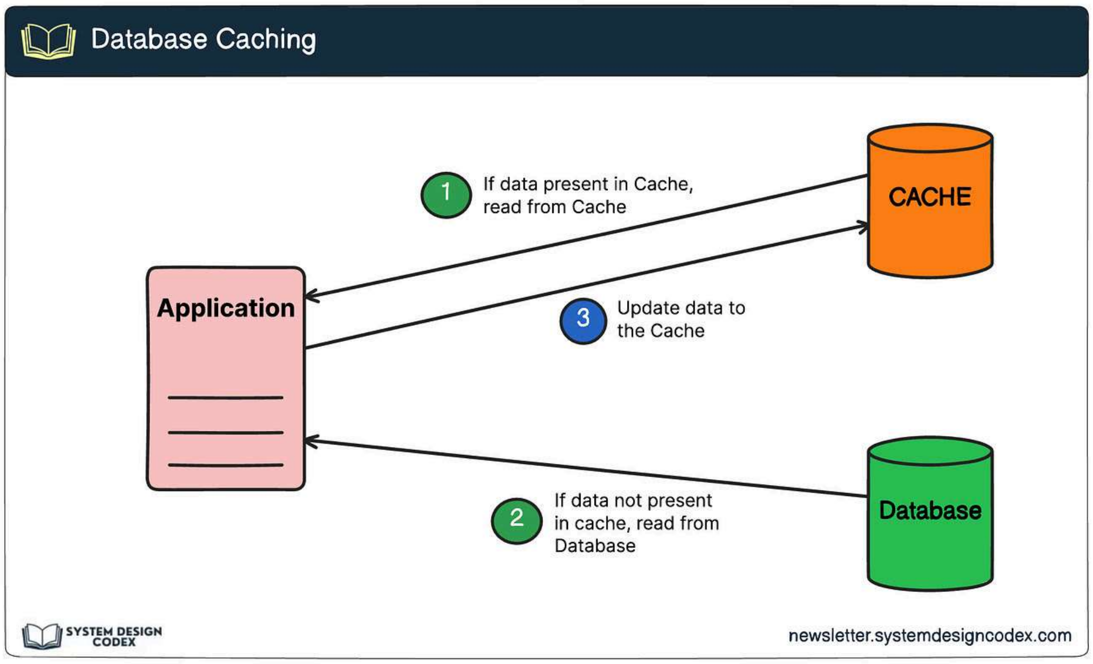
You can play around with the diagram on Eraser.io

4 - Caching for read-heavy system

Unlike popular belief, Caching isn't a silver bullet.

But if you've got a read-heavy system that is struggling with performance issues, using a cache can be a good idea.

It will give you time to fix the real issues and also can work as a long-term solution.


Database Caching
1 If data present in Cache, read from Cache
CACHE
Application
3 Update data to the Cache
Database
2 If data not present in cache, read from Database
SYSTEM DESIGN CODEX
newsletter.systemdesigncodex.com

[You can play around with the diagram on Eraser.io]

5 - Listen and Record

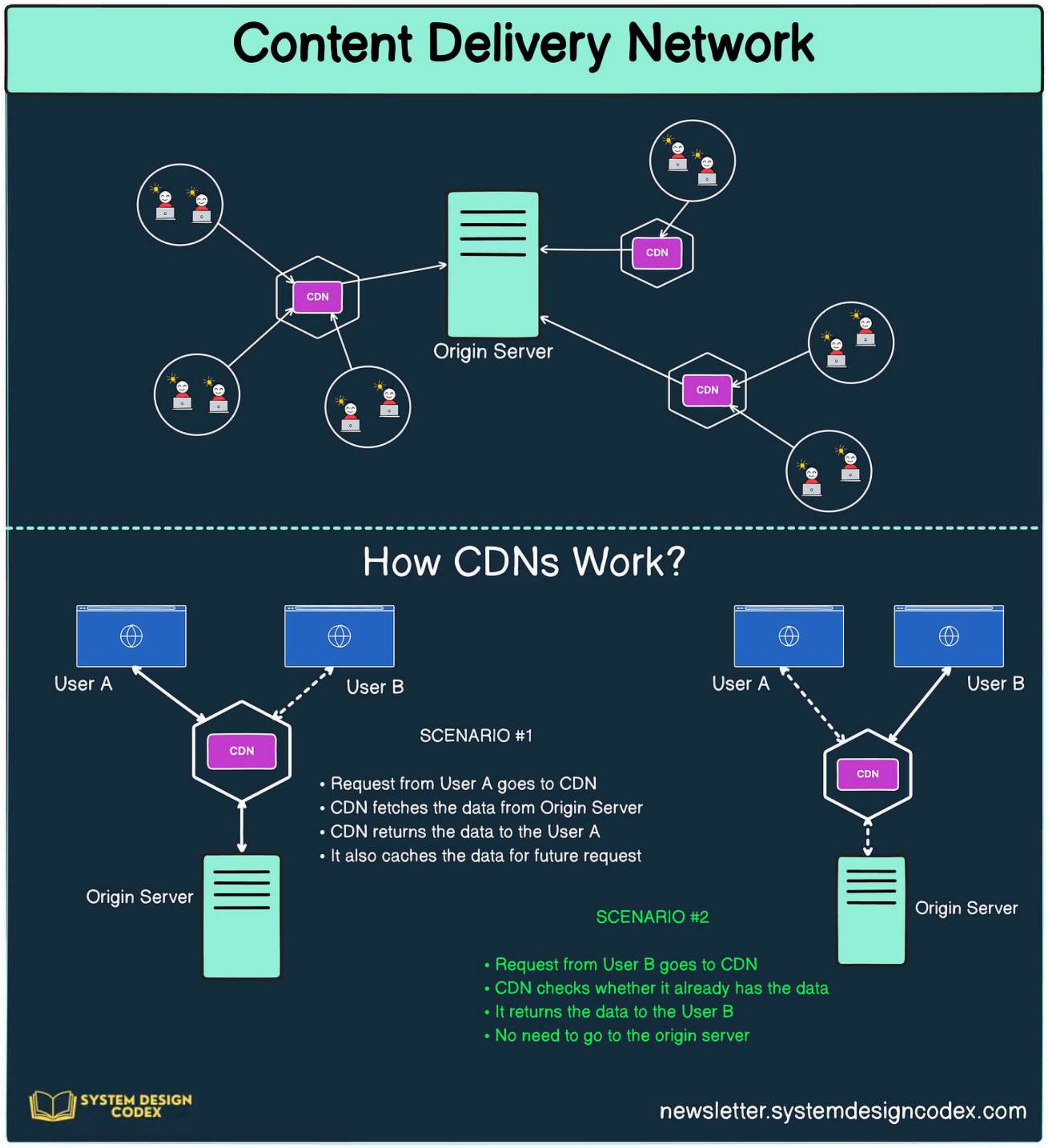
Your biggest superpower in an interview is listening.

Listen to the system requirements properly and note them down for reference as you build your solution.

6 - CDN to Reduce Latency

CDNs or Content Delivery Networks are your friend if your goal is to reduce latency for the end users especially when it comes to loading static assets.

CDNs also help with the global distribution of data and mitigating the impact of DDoS attacks.


Content Delivery Network
CDN
CDN
CDN
CDN
Origin Server
How CDNs Work?
User A
User B
CDN
Origin Server
SCENARIO #1
• Request from User A goes to CDN
• CDN fetches the data from Origin Server
• CDN returns the data to the User A
• It also caches the data for future request
User A
User B
CDN
Origin Server
SCENARIO #2
• Request from User B goes to CDN
• CDN checks whether it already has the data
• It returns the data to the User B
• No need to go to the origin server
SYSTEM DESIGN CODEX
newsletter.systemdesigncodex.com

[You can play around with the diagram on Eraser.io]

7 - Create the Right Indexes

To improve the performance of your database queries, create the right indexes.

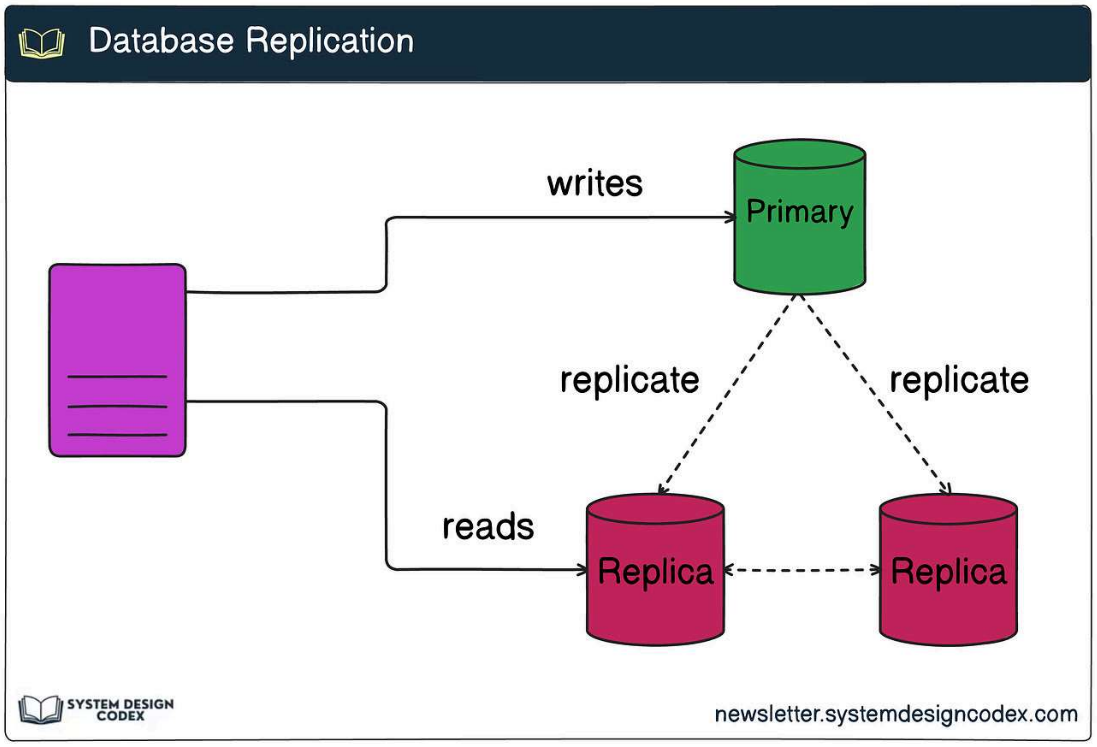
A good indexing strategy can also mean not having the pressure to invest in other strategies like caching.

8 - Scale the reads with Database Replication

A majority of the systems that we encounter are read-heavy.

To scale the reads, a common strategy is to use the primary for all the write operations and route the read requests to the replicas. Of course, critical reads still go to the primary but you get the point.

The trade-off here is between high availability and the required consistency level.

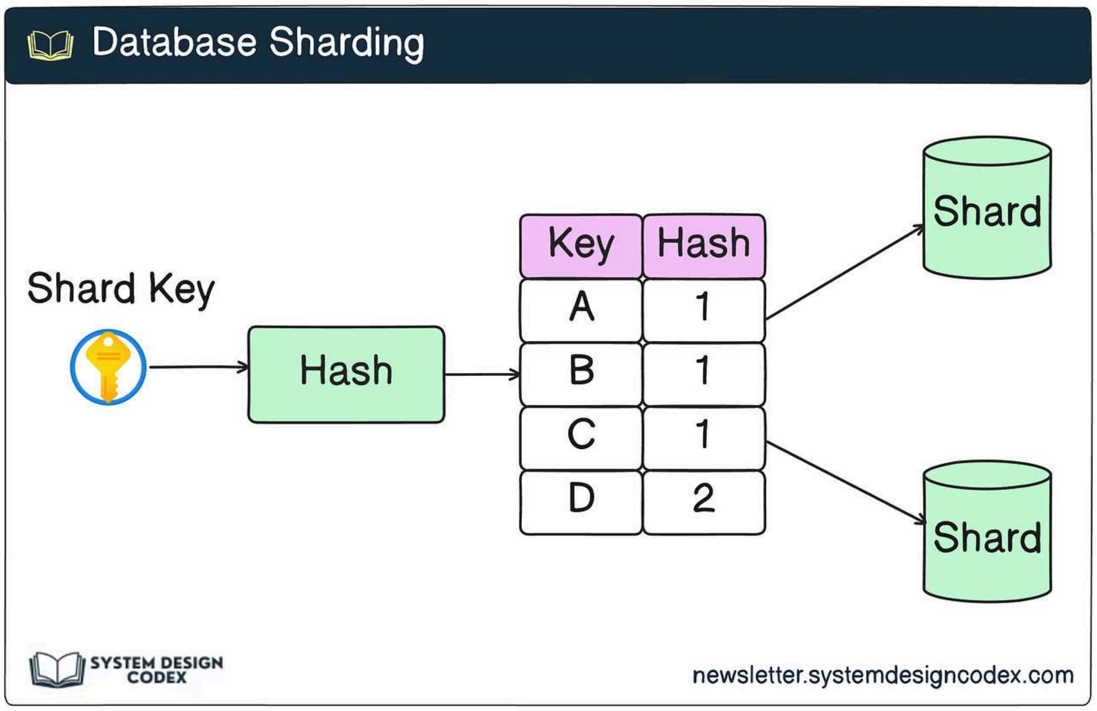


You can play around with the diagram on Eraser.io

9 - Scale the writes with Database Sharding

Though read-heavy systems are more common, sometimes you also need to scale the writes.

In such a case, you can consider database sharding where you partition the table across multiple nodes so that the load could be distributed efficiently.



You can play around with the diagram on Eraser.io

10 - Clarify assumptions

In a system design interview, don’t jump into solution mode straight away.

Always spend some time clarifying any assumptions you might have about the potential solutions.

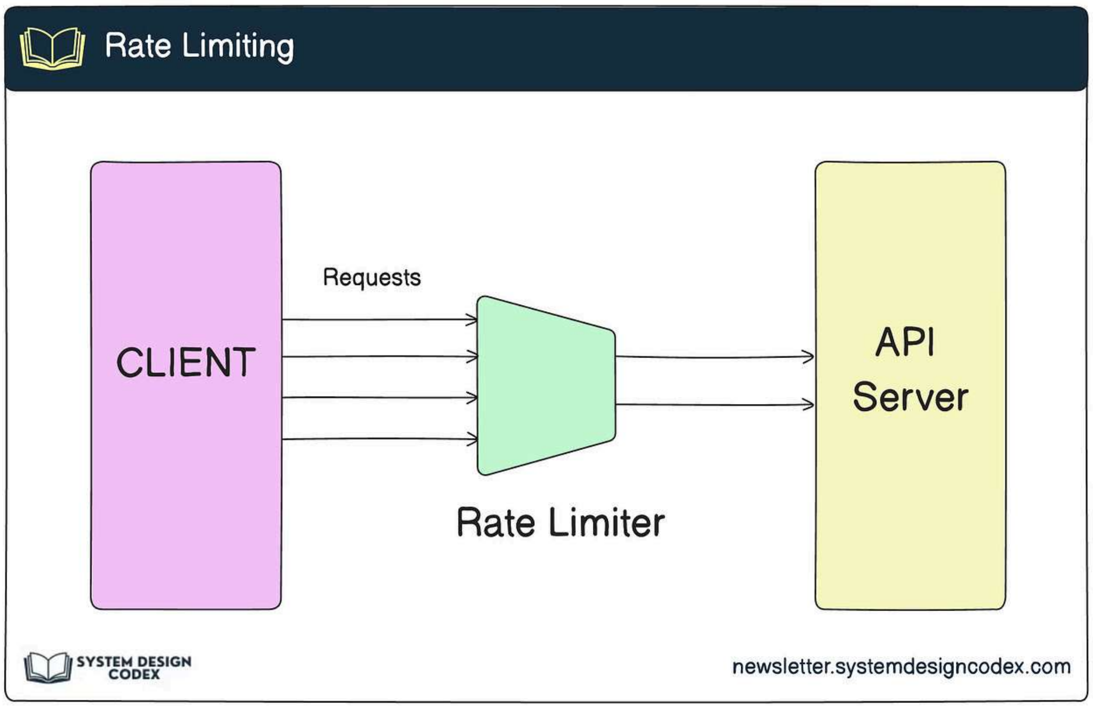
11 - Store complex data in Object Storage

If the requirement is to store complex data such as videos, images, and files, use an Object storage.

All major cloud providers have specific services to support this.

12 - Rate Limiting to Regulate Usage

If you want to manage the load on your service, control the usage of a particular resource, and mitigate the impact of DoS attacks, use Rate Limiting.

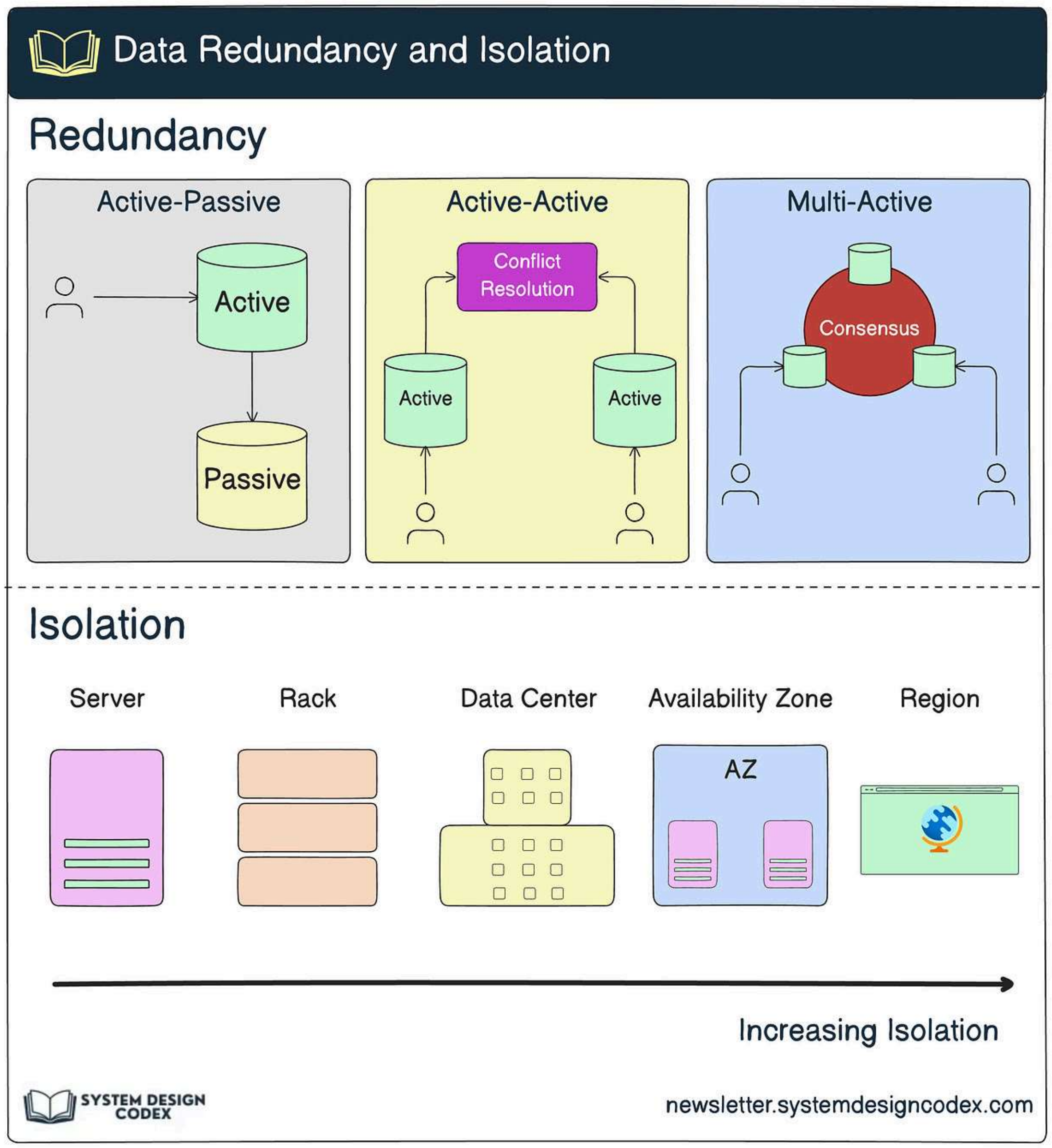


You can play around with the diagram on Eraser.io

13 - Remove Single Points of Failure

Single points of failure are like the Achilles heel of your system design.

Build data redundancy and isolation in your design to remove any potential single points of failure.



You can play around with the diagram on Eraser.io

14 - Consider Non-Functional Requirements

Don't ignore the non-functional requirements in an interview setup. For example, consider things like:

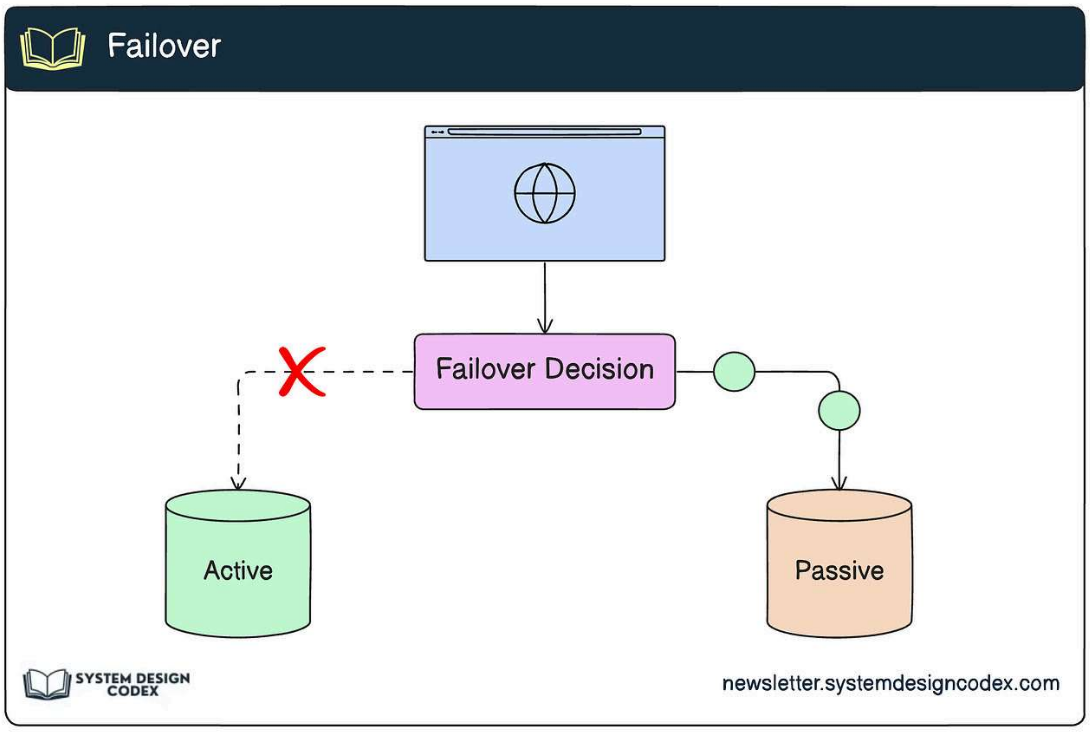
"The website pages should load in 3 seconds with the total number of simultaneous users <5 thousand"

These requirements can have a big impact on your design choices. When in doubt, clarify them with the interviewer and build the solution accordingly.

15 - Improve Fault Tolerance with Failover

Failures are going to occur in a system.

To improve the fault tolerance of your system, invest in failover strategies.

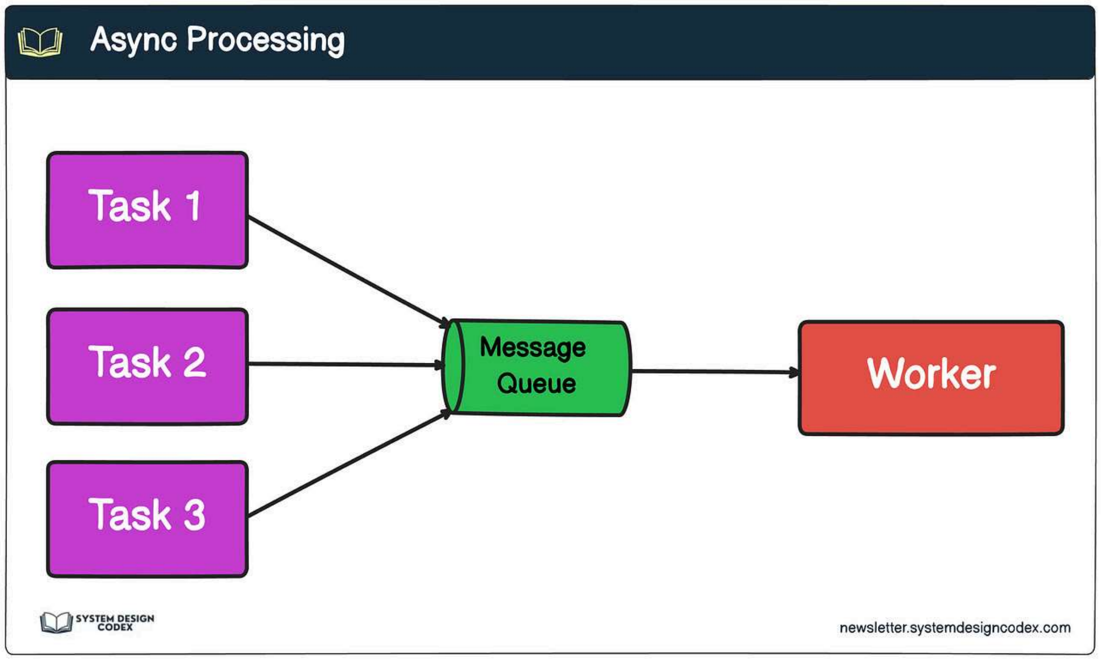


[You can play around with the diagram on Eraser.io]

16 - Strategy for Long-Running Tasks

Many times, you've to perform some heavy computation and execute long-running tasks. These can degrade the user experience if not handled properly.

Use async processing to execute long-running tasks.



You can play around with the diagram on Eraser.io

17 - Event-Driven Design for Loose Coupling

Tight coupling between components can reduce the velocity of the team while adding new features and making changes.

Consider using an event-driven approach when you want to design for agility and reduce the blast radius of changes.

18 - NoSQL Databases for Unstructured Data

The lines between SQL and NoSQL databases have blurred over the years and you can use them quite interchangeably in most cases.

However, typically, for unstructured data and flexible schema, consider going for NoSQL databases.

19 - Constant Feedback

Think of the system design interview as a discussion rather than one-sided storytelling.

Keep getting feedback along the way as you walk through the design choices rather than waiting till the end and getting bombarded with tough questions.

20 - Data Compression & Pagination

To handle large amounts of data flowing through the network without creating bandwidth issues, consider using data compression and pagination techniques.

21 - CI/CD for Automated Builds

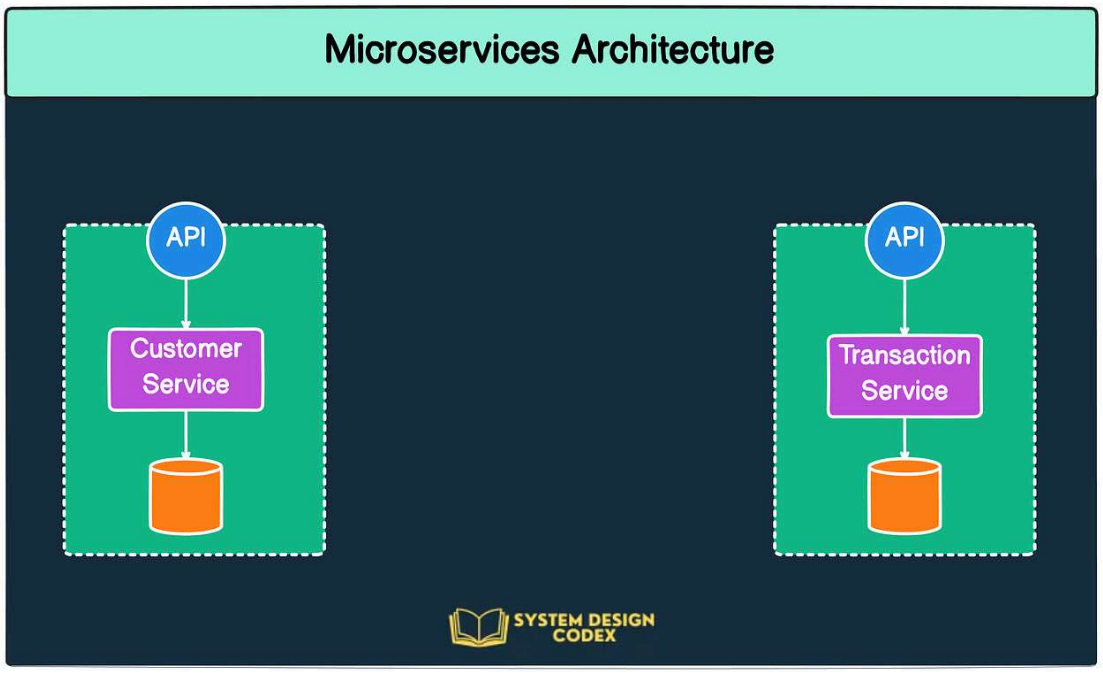
To improve delivery velocity and have reliable software releases, you need automatic builds and deployments.

Consider investing in a robust CI/CD pipeline to handle this requirement.

22 - Microservices for Independent Deployments

A monolith or modular monolith can take your application quite far. Billion-dollar companies have been built using traditional monoliths.

However, if you want to achieve independent deployments and scaling of various parts of the system, you can opt for microservices architecture. At the same time, you also need to be prepared to handle the complexities that come with microservices.



You can play around with the diagram on Eraser.io

23 - There are no Perfect Answers

Lastly, remember that there are no perfect answers in system design.

Most system design is about considering a bunch of trade-offs and making the least worst design choices.

These interviews are meant to test your analytical skills and adaptability.

Be confident about your answers and maintain good eye contact during the conversation.

👉 **So - what other tips do you think can help in a System Design interview?**

Leave a comment

Eraser Professional Plan Free Trial (Affiliate)

As you all know, I use the Eraser for drawing all the diagrams in this newsletter.

Eraser is a fantastic tool that you can use as an all-in-one markdown editor, collaborative canvas, and diagram-as-code builder.

And now you can get **one month free on their Professional Plan** or a **$12 discount** if you go for the annual plan. The Professional Plan contains some amazing features like unlimited AI diagrams, unlimited files, PDF exports, and many more.

Head over to Eraser and at the time of checkout, use the promo code “**CODEX”** to get this offer now.

Shoutout

- Refactor like a Pro by **Daniel Moka**

- Dealing with Impostor Syndrome in the Engineering World by **Nicola Ballotta**
- How To Get More Done – HR hates me for this weird trick by **Akos**
- How a 17th-century shipbuilding technique revolutionized modern software by **Amrut Patil**

That's it for today! 🌞

Enjoyed this issue of the newsletter?

Share with your friends and colleagues.

Share

See you later with another value-packed edition — Saurabh.

LIKE COMMENT RESTACK