BOSSCODER ACADEMY X REVANTH MURIGIPUDI

FUNDAMENTALS OF SYSTEM DESIGN

Explained in **the simplest way**

BOSSCODER ACADEMY

DISCLAIMER

Let's understand system design with real-life examples.

The comparisons are just to make you understand the topics easily and can vary in real life

BOSSCODER ACADEMY

A **SCENARIO**

Let's say you are the floor manager for printers at a co-working space. Now let us understand system design using this

BOSSCODER ACADEMY

SCALING HAS :

- Horizontal Scaling

- Vertical Scaling

BOSSCODER ACADEMY

HORIZONTAL SCALING

Problem :

You have only one printer and, there are people lined up to use the printer & you don't want to make them wait long!

Solution :

Buy more printers, now people can use them without waiting for lot of time

Buying more servers to handle the request load is called Horizontal Scaling

BOSSCODER ACADEMY

VERTICAL SCALING

Problem :

You have a printer, but it's taking a very long time to print even a black and white sheet

Solution :

You buy an expensive printer with better & higher performance

Adding more power in terms of hardware or software to your existing server is called Vertical Scaling

BOSSCODER ACADEMY

LOAD BALANCING

Problem :

People lined up to use printers and even more people are coming, so you want to make sure that everyone gets to use the printer & people are not crowding a single printer

Solution :

You as a floor manager, observe which printer has less queue of people & start directing people to those printers

BOSSCODER ACADEMY

In a nutshell, a load balancer distributes traffic evenly onto your servers preventing server overloads

It uses a consistent hashing algorithm underneath to accomplish this evenly distribution

BOSSCODER ACADEMY

MICROSERVICE VS MONOLITH

Problem :

You have a big printer, performing all sorts of prints → printing color papers, A4 sheets, A3 sheets, posters, banners & lots of people are queuing up to use one printer

Solution :

Rather than making only one printer handle every single type of print, you can have multiple printers which perform one task alone, i.e one printer prints a4 sheet, one prints banners & now as demand rises, you can think of scaling individual type printers alone

But, that rises the cost & maintenance too!

If you're fine with a hike, then divide printers,
else the first printer alone is good enough

Monoliths are a single standalone application (a big printer) that performs all the tasks. Whereas microservices are a collection of small & independent services which are deployed and scaled individually

Both have their pros + cons & picking monolith or microservice architecture depends on the system and the use case

BOSSCODER ACADEMY

A = APPLICATION
P = PROGRAMMING
I = INTERFACE

Problem :

Now, you have to take a print

- How do you get it done?
- Do you also want to know how the printer applies ink & prints on paper?

Solution :

You can use a printer to get your print. And do you want to know how it does printing?

No right! You don't want details! All you care about is whether it is printed properly or not when given a print operation !

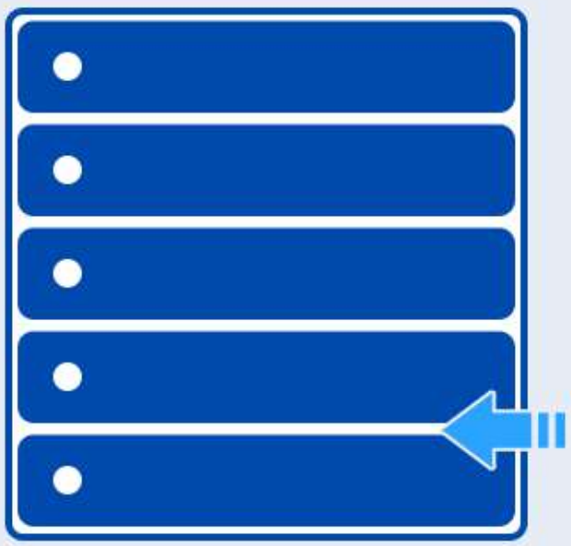
What did we observe here? You have a request (to get prints), and you're using the printer (API) to accomplish that task and get printed papers (response output)

BOSSCODER ACADEMY

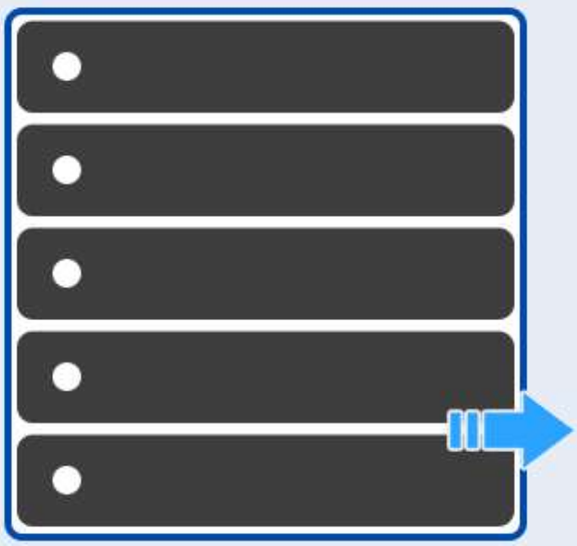
API allows communication between two services etc. They just let the applications talk to each other in form of requests & responses. The person who sends a request doesn't care about how it's being handled, but only about receiving a response

BOSSCODER ACADEMY

PROXIES HAS :



Reverse proxy



Forward proxy

BOSSCODER ACADEMY

REVERSE PROXY

Problem :

People are damaging the printers & for security reasons, you don't want anyone to physically interact with printers anymore

Solution :

You as the manager, will take their print request & ask them to wait and you will perform an operation & give back the copy to them

BOSSCODER ACADEMY

What happened here ?

People didn't use the printer directly, they are unknown which printer is printing their copy. The manager acted as the middleman for getting their prints

Similarly, a reverse proxy sits before the application servers and receives client requests, validates them & does a lot more things like load balancing, caching, traffic control, and forwards the request to the server, and returns the response back to the client

BOSSCODER ACADEMY

FORWARD PROXY

Problem :

So, let's say you've taken a thousand prints and the people are angry with you. But you need another print

So what do you do now?

Solution :

You send your friend instead of you going directly and taking heat from everyone

BOSSCODER ACADEMY

What happened here?

You basically asked your friend to do your job. Also, the people don't know that the actual print is for you

In the same way, A forward proxy also sits between client and server but disguises the client's IP address blocking a lot of unwanted incoming traffic as well

BOSSCODER ACADEMY

CACHING

Problem :

It’s a festival, & lots of people are now coming to print posters for the festive event. Everyone wants to use the printer, and potentially wait a lot of time & overload the printers as well

Solution :

Instead of everyone using & overloading the printer for the same print, you as a floor manager can print lots of templates beforehand so people can directly take from you

BOSSCODER ACADEMY

Caching is a temporary and fast access storage layer where information can be stored

Instead of hitting the database for information, you hit the cache first, & if it's not present (cache miss), only then do you hit the database !

BOSSCODER ACADEMY

MESSAGE QUEUE

Problem :

There’s a bulk-hiring event happening on the floor and lots of candidates are coming to print their resumes

So what can happen? They'll flock to the printing room and there will be chaos even though there are lots of printers !

Solution :

To avoid that, you as a floor manager will inform that at maximum, only 10 people are allowed into the printing room at any point in time

BOSSCODER ACADEMY

Or, you take the print request from candidates and assure them you'll give them a print as soon as it's ready. This won't stop the demand + makes things more structured & you know there won't be any huge chaos

A message queue (MQ) is an asynchronous communication between services

There's a producer that creates a message and a consumer listening in the queue

The producer just puts the message on the MQ and doesn't expect an immediate response either

BOSSCODER ACADEMY

CONTENT DELIVERY NETWORK

Problem :

Now your printing machines are a big hit. Everyone is happy using the printers, but suddenly there's a power outage on the floor and nobody is able to get any prints

What now?

Solution :

How about you create another printing space on some other floor or building? This solves a couple of things

BOSSCODER ACADEMY

1. If printers on a floor or building are down, people can use printers on other floors

2. All printers are working, and let's say you have printers on floors 10 and 3

   A person on floor 11 can just go to floor 10 and get his work done fast instead of going to floor 3 unless floor 10 is absolutely occupied

BOSSCODER ACADEMY

This is what CDN does too, Instead of having servers in only one location, you have them distributed geographically across multiple data centers

Now you can serve requests super quickly & even be tolerant to failures if a data center goes down (since you have other data center working fine)

BOSSCODER ACADEMY

NETWORK PROTOCOLS

Problem :

You want to talk to the floor manager about some issue related to the printers

How do you do that?

Solution :

You speak in a language understood by the manager, if he understands English, you speak English, if he understands Hindi, you speak Hindi

BOSSCODER ACADEMY

Similarly, network protocols are nothing but a set of rules used for communication between anything over the internet

Remember we talked previously that APIs allow communication?

Well, they use network protocols to communicate properly!

Ex- IP, TCP, HTTP, etc

BOSSCODER ACADEMY

CONTAINERIZATION

You've developed an application & it runs perfectly on your workspace, but it doesn't work on another operating system

And, that's a problem, right?

That's where containers come into play. A container is essentially an isolated space, like a lightweight virtual machine and it bundles everything needed for your application to run properly, like the binaries, libraries, packages, dependencies, etc

BOSSCODER ACADEMY

In another concept called “orchestration” you can manage these containers → provisioning, deployment, scaling, networking, etc

BOSSCODER ACADEMY

CAP THEOREM

CAP stands for :

C = Consistency
A = Availability
P = Partition Tolerance

- Consistency - All the clients see identical data at the same time no matter what
- Availability - A system is up 100% of the time and responds to every request regardless of the state of the server/node

BOSSCODER ACADEMY

- **Partition Tolerance** - When there's a break in communication between the servers (maybe a network outage), even then the servers should work regardless of this partition in the system

In an ideal world, it's not possible to have all three of them coexist together

There has to be a trade-off made and we pick one to suit our scenario, i.e CA (or) CP (or) AP

BOSSCODER ACADEMY

SQL VS NOSQL

Picking the right database for your system is probably one of the important choices to make

The right one would mean less latency, fewer costs, etc, whereas the wrong one would become expensive and unscalable with constant schema updates frequently

BOSSCODER ACADEMY

Picking a relational or a non-relational database entirely depends on the use case & it's often opinionated. Let's say your data is structured and would require ACID properties, then going with a relational database makes sense

Even in a non-relational databases, there are types → Document DB, Columnar DB, Search DB, etc. Also, there can be several instances where you will use a combination of multiple databases for a system as well!

BOSSCODER ACADEMY

INDEXING

An index is a data structure that helps in improving the performance of the database when a query is hit to retrieve records

In an index, you'll have a couple of columns. The first one is "search key", typically a primary or candidate key of the table & the second column has the pointers containing the address to where exact that key value can be found

There are multiple indexes → primary, clustering, secondary, etc

BOSSCODER ACADEMY

SHARDING

When you want to query **huge data** in your database, it's an expensive. So, you can either optimize the query put indexes or shard the data

So, sharding is nothing but, the division of the the data into different database servers. The partitioning is done on a key (typically an attribute in the data we store)

It helps in managing data since it's distributed across multiple database servers in small and easily manageable parts!

BOSSCODER ACADEMY

Bosscoderacademy.com

Recruiters in Meta Apple Amazon Netflix Google seek engineers
who can design scalable systems

System design is one of the important ingredients
along with **DSA**, **CS Fundamentals**

BOSSCODER ACADEMY HAS GOT YOU COVERED ENTIRELY !

Within a span of 7 months, you will
develop skills + confidence + hands-on experience
to grab your dream job

BOSSCODER ACADEMY

Bosscoderacademy.com

200+ alumni placed at Top product-based companies

More than 120% hike for every 2 out of 3 working professional

Avg package of 22 LPA

I would recommend **Bosscoder Academy** to anyone who wants to either become an expert in DSA or crack tech giant interviews

Rahul Verma
Google

Bosscoder gives you everything starting from **DSA**, **LLD** and **HLD** followed **CS Fundamentals**

Dheeraj Barik
amazon