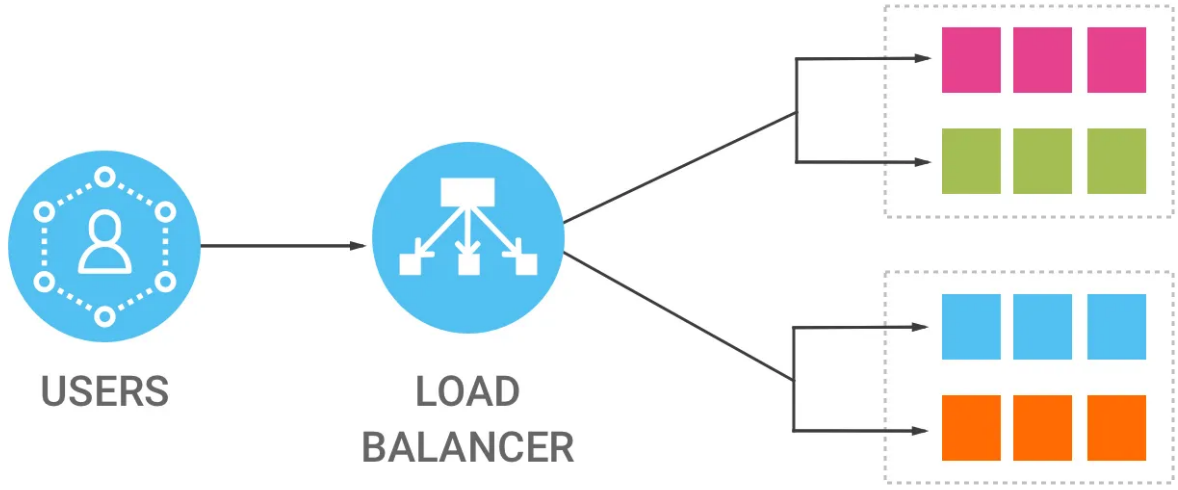
✦ Member-only story

System Design — Load Balancing

Concepts and considerations for Load Balancing in System Design

Larry | Peng Yang · Follow
Published in Computer Science Fundamentals
9 min read · Apr 6, 2020



❤️ Thank you for your interest in this article, if you like the content feel free to Subscribe, clap👏🏻 and share it.❤️

Table of Contents

4. Load Balancer locations
5. Algorithms
6. Implementation
7. Other

1. Concepts

A load balancer is a device that acts as a reverse proxy and distributes network or application traffic across a number of servers. It helps scale **horizontally** across an ever-increasing number of servers.

2. How does the load balancer work?

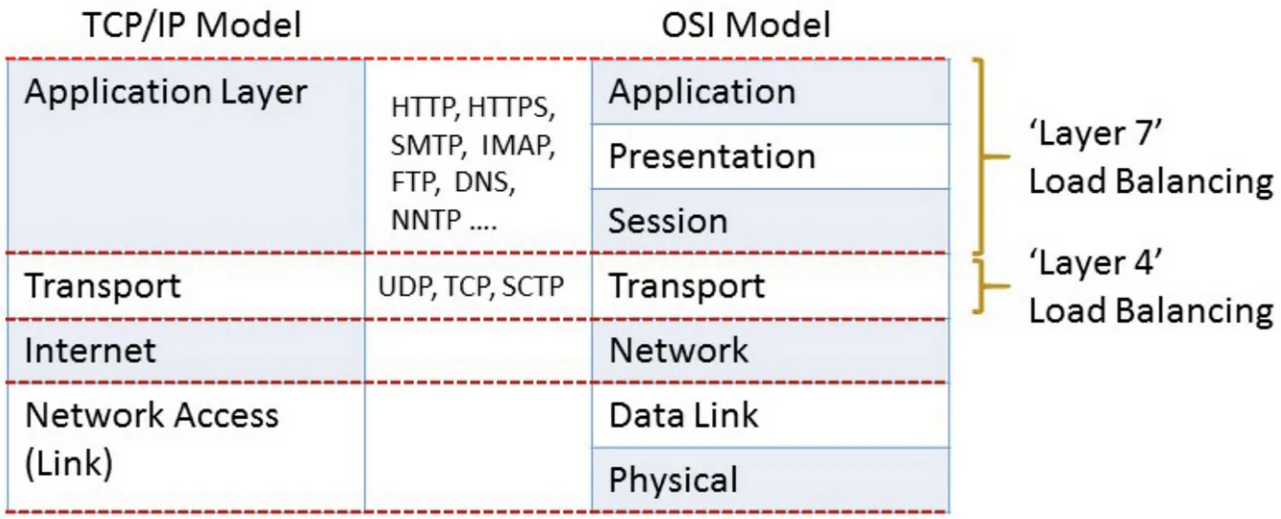
- **Define IP or DNS name for LB:** Administrators define one IP address and/or DNS name for a given application, task, or website, to which all requests will come. This IP address or DNS name is the load balancing server.
- **Add backend pool for LB:** The administrator will then enter into the load balancing server the IP addresses of all the actual servers that will be sharing the

which sits between your app servers and your users worldwide and accepts all traffic, or as a **gateway**, which assigns a user to a server once and leaves the interaction alone thereafter.

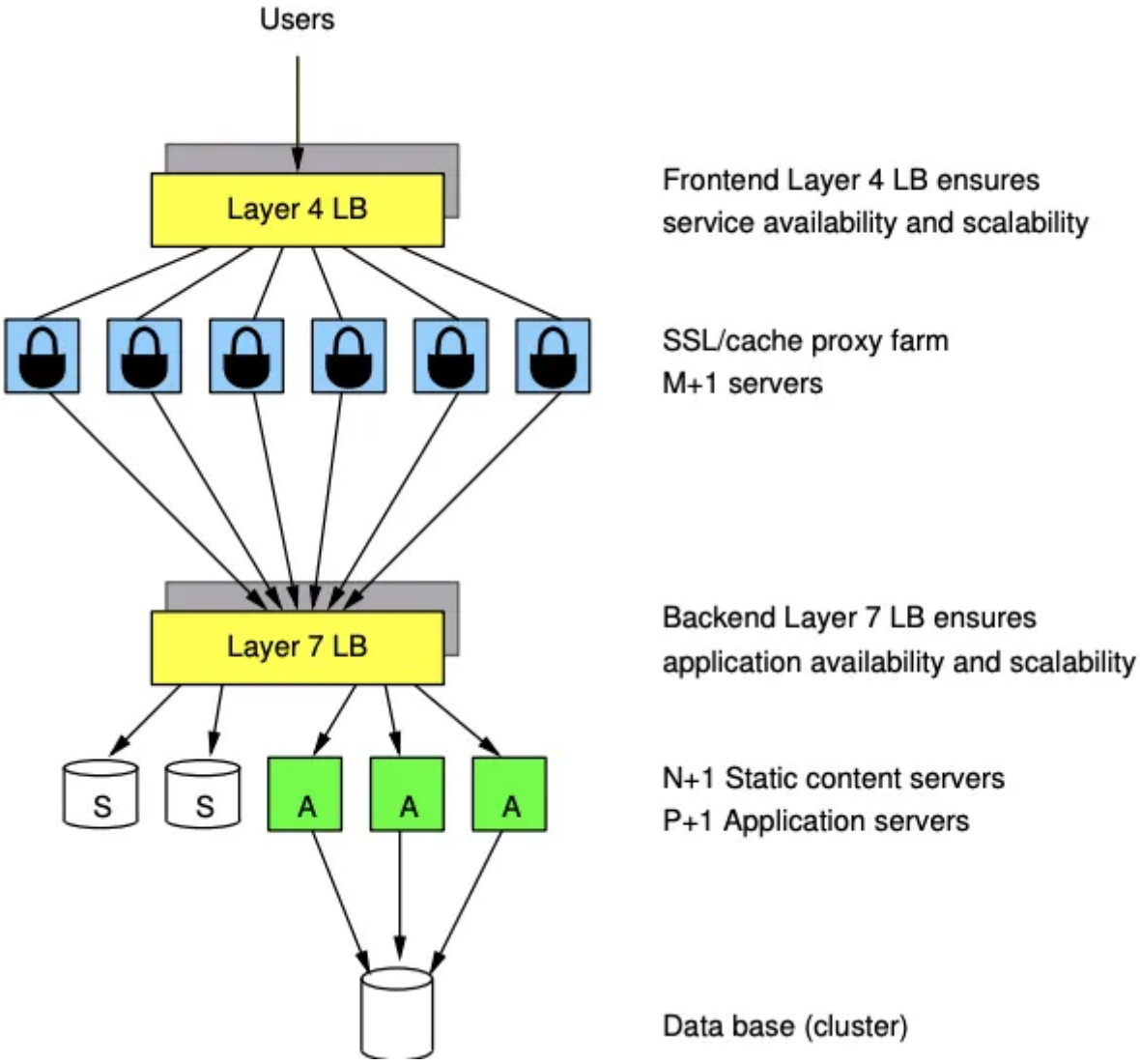
- **Redirect requests**: Once the load balancing system is in place, all requests to the application come to the load balancer and are redirected according to the administrator's preferred algorithm.

3. Types of LBs

- Load balancers are generally grouped into two categories: Layer 4 and Layer 7.



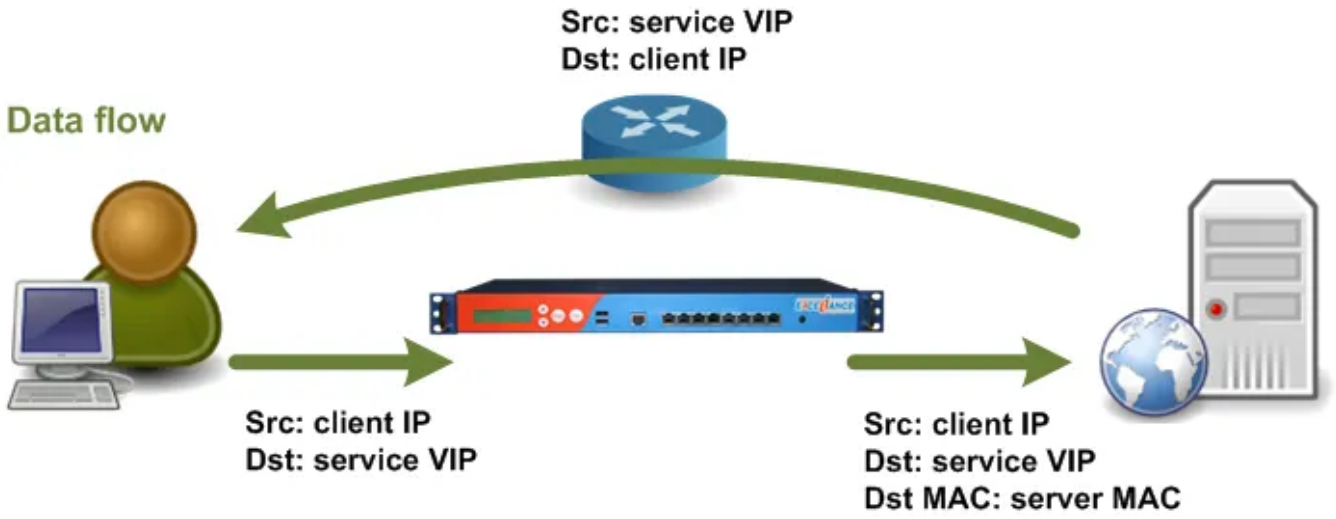
L4 vs L7



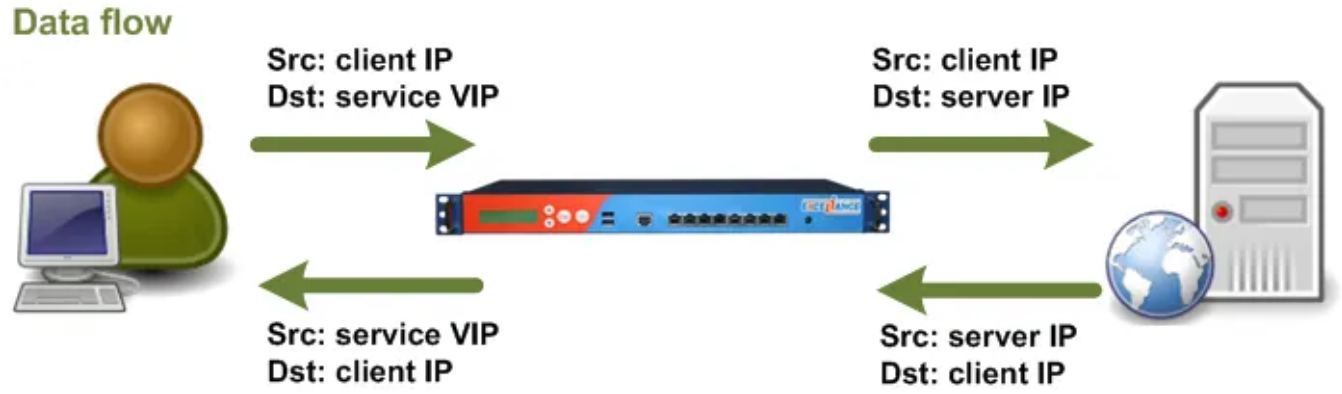
L4 vs L7

Layer 4 load balancers

- It acts upon data found in **network and transport layer** protocols (IP, TCP, FTP, UDP). They are mostly the **network address translators** (NATs) which share the load to the different servers getting translated to by these load balancers.
- Session persistence can be achieved at the IP address level
- No termination for TCP connections
- Two modes of L4 LB:
  1. Direct Server Return (DSR): The TCP connection is established directly between the client and the backend. The load-balancer sees only the requests and **just changes the destination MAC address** of the packets. The backends answer directly to the client using the **service IP (VIP)** configured on a loopback interface (this is why the src is VIP for response). **Note that the requests pass through the load balancer while the responses not.** LB won't be the bandwidth bottleneck.
  2. NAT Mode: The clients get connected to the **service VIP**. The load balancer chooses a server in the pool then forwards packets to it by **changing the destination IP address (DNAT)**, the LB becomes the default gateway for the real servers, and the source IP is the client's IP. **All traffic will pass through the load balancer**, and output bandwidth is limited by load balancer output capacity. There is only one connection established.



DSR Mode



NAT Mode

- Example: Azure Load Balancer, Nginx as TCP and UDP load balancer

Layer 7 load balancers

- It distributes requests based upon data found in **application layer** protocols such as HTTP. They can further distribute requests based on application-specific data such as HTTP headers, cookies, or data within the application message itself, such as the value of a specific parameter.
- For the client, the destination IP will be the IP of the load balancer, for the backend server, the source IP will be the IP of the load balancer.
- The cookie can be used to achieve a sticky session.
- IP of the client will be kept with the **X-Forwarded-For** header.
- To get the HTTP information, the connection is terminated at the load balancer, thus, there will be 2 TCP connections: Client-LB, LB-Backend.
- Example: Azure Application Gateway, Nginx as HTTP load balancer

4. LB Locations

- Between user and web servers (User => Web Servers)
- Between web servers and an internal platform layer (application servers, cache servers) (Webservers => App or Cache servers)
- Between internal platform layer and database (App or Cache servers => Database servers)

5. Algorithms

- Least connection

- Least response time
- Least bandwidth
- Round robin
- Weighted round-robin
- IP hash

6. Implementations

Reference: https://www.thegeekstuff.com/2016/01/load-balancer-intro/

Smart clients

- *Developers lean towards smart clients because they are developers, and so they are used to writing software to solve their problems, and smart clients are software.*
- The smart client is a client that takes a pool of service hosts and balances load across them, detects downed hosts, and avoids sending requests their way (they also have to detect recovered hosts, deal with adding new hosts, etc, making them fun to get working decently and a terror to setup).

Hardware load balancers

- Hardware load balancers are specialized hardware deployed in-between the server and the client. It can be a **switching/routing hardware or even a dedicated system** running a load balancing software with specialized capabilities. The hardware load balancers are implemented on Layer4 and Layer7 of the OSI model so prominent among this hardware are L4-L7 routers.
- Hardware Load Balancer Examples:
  1. **F5 BIG-IP load balancer** (Setup HTTPS load balance on F5)
  2. CISCO system catalyst
  3. Barracuda load balancer
  4. Coytepoint load balancer
  5. Citrix **NetScaler**

Software load balancers

- Software load balancers generally implement a combination of one or more scheduling algorithms. They are often installed on the servers and consume the processor and memory of the servers.

- The following are a few examples of software load balancers:
  1. HAProxy — A TCP load balancer.
  2. NGINX — An HTTP load balancer with SSL termination support.
  3. mod_athena — Apache-based HTTP load balancer.
  4. Varnish — A reverse proxy-based load balancer.
  5. Balance — Open-source TCP load balancer.
  6. LVS — Linux virtual server offering layer 4 load balancing.

7. Other Topics

7.1 Load balancer vs Reverse proxy

- They are components in a client-server computing architecture. Both act as intermediaries in the communication between the clients and servers, performing functions that improve efficiency. **Most load balancer programs are also reverse proxy servers**, which simplifies web application server architecture.

- **Load balancers** are most commonly deployed when a site needs ***multiple servers***. Some load balancers also provide ***session persistence***. It refers to direct a client's requests to the same backend web or application server for the duration of a "session" or the time it takes to complete a task or transaction (e.g. shopping).

- It often makes sense to deploy a **reverse proxy** even with just *one web server or application server.*

7.2 Local vs Global Load Balancing

- Originally, load balancing referred to the distribution of traffic across servers in one locality — a single data center.

- As more and more computing is done online, load balancing has taken on a broader meaning. **Global Server Load Balancing** is the same in principle but the load is distributed planet-wide instead of just across a data center. Early generation GSLB solutions relied on **DNS load balancing**, which limited both

their performance and efficacy os it is now considered acceptable mostly for simple applications or websites.

- Another implement of GSLB is through a Content Delivery Network (CDN), such as the Cloudflare CDN. A global CDN service will take data from their customers' origin servers and cache it on a geographically distributed network of servers, providing fast and reliable delivery of Internet content to users around the world.

7.3 Load Balancer and Certificates

- https://serverfault.com/questions/68753/does-each-server-behind-a-load-balancer-need-their-own-ssl-certificate
- If we do load balancing on the TCP or IP layer (OSI layer 4/3, a.k.a L4, L3), then all HTTP servers will need to have the SSL certificate installed.
- If we do load balance on the HTTPS layer (L7), then you'd commonly install the certificate on the load balancer alone, and use plain unencrypted HTTP over the local network between the load balancer and the webservers (for best performance on the web servers). The certificate installed on the load balancer can be used to decrypt the data (cookie, etc). This is called L7 TSL termination.
- How to Setup F5 HTTPS SSL Load Balancing in Big-IP

7.4 L4 Load Balancer Configuration with Nginx (TCP and UDP)

The `Hash` load-balancing method is also used to configure *session persistence*. As the hash function is based on the client's IP address, connections from a given client are always passed to the same server unless the server is down or otherwise unavailable. Specify an optional `consistent` parameter to apply the ketama consistent hashing method: **`hash`** `$remote_addr consistent;`

```
stream {
    upstream stream_backend {
        hash $remote_addr;
        server backend1.example.com:12345 weight=5;
        server backend2.example.com:12345 max_fails=2
fail_timeout=30s;
        server backend3.example.com:12345 max_conns=3;
    }
```

```
    upstream dns_servers {
        least_conn;
        server 192.168.136.130:53;
        server 192.168.136.131:53;
        server 192.168.136.132:53;
    }

    server {
        listen        12345;
        proxy_pass    stream_backend;
        proxy_timeout 3s;
        proxy_connect_timeout 1s;
    }

    server {
        listen     53 udp;
        proxy_pass dns_servers;
    }
}
```

7.5 L7 Load Balancer Configuration with Nginx (HTTP)

Load Balance the traffic based on requesting URI (Nginx reverse proxy). By default the round-robin algorithm will be used, we can also use least_conn, configure sticky cookie, etc.

```
http {
    server {
        listen 80;

        location / {
            proxy_pass http://backend;
        }
        location /images {
            proxy_pass http://backend-static;
        }
    }

    upstream backend {
        server web-server1:80;
        server web-server2:80;
        sticky cookie srv_id expires=1h domain=.example.com path=/;
    }

    upstream backend-static {
        least_conn;
        server web-server3:80;
        server web-server4:80;
    }
}
```

We can also do TSL termination if we `listen 443` and configure below.

```
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }
}
```

7.6 Setup an HA load balancer with Nginx

In 7.4 and 7.5, we demonstrated how to set up L4 and L7 load balancers with Nginx, but how can we make sure the load balancer itself is highly available? The basic idea for an active-passive high-availability (HA) setup is as below:

1. Setup an Nginx cluster with 2+ nodes with **keepalived** daemon.
2. An implementation of the Virtual Router Redundancy Protocol (VRRP) to manage virtual routers (virtual IP addresses, or *VIPs*). VRRP ensures that there is a master node at all times. The backup node listens for VRRP advertisement packets from the master node. If it does not receive an advertisement packet for a period longer than three times the configured advertisement interval, the backup node takes over as master and assigns the configured VIPs to itself.
3. A health-check facility to determine whether a service (for example, a web server, PHP backend, or database server) is up and operational.

If a service on a node fails the configured number of health checks, `keepalived` reassigns the virtual IP address from the master (active) node to the backup

(passive) node. On each node, the configuration file will contain the following information:

- The IP address of the local and remote nodes (one of which will be configured as a master, the other as a backup)
- One free IP address to be used as the cluster endpoint's (floating) VIP

To see which node is currently the master for a given VIP, run the `ip addr show` command for the interface on which the VRRP instance is defined (in the following commands, interface `eth0` on nodes `centos7-1` and `centos7-2`):

```
centos7-1 # ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state
     UP qlen 1000
    link/ether 52:54:00:33:a5:a5 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.100/24 brd 192.168.122.255 scope global dynamic
eth0
       valid_lft 3071sec preferred_lft 3071sec
    inet 192.168.100.150/32 scope global eth0
       valid_lft forever preferred_lft forever

centos7-2 # ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state
     UP qlen 1000
    link/ether 52:54:00:33:a5:87 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.101/24 brd 192.168.122.255 scope global eth0
       valid_lft forever preferred_lft forever
```

In this output, the second `inet` line for `centos7-1` indicates that it is `master` – the defined VIP (`192.168.100.150`) is assigned to it. The other `inet` lines in the output show the master node's real IP address (`192.168.100.100`) and the backup node's IP address (`192.168.100.101`).

References

- https://www.imperva.com/learn/availability/load-balancing-services/
- https://www.nginx.com/resources/glossary
- https://www.f5.com/services/resources/glossary
- https://avinetworks.com/glossary/

- Making applications scalable with Load Balancing
- Introduction to modern network load balancing and proxying
- Load balancing methods
- L7 LB and Session persistence, SSL, reverse proxy (Japanese)
- L4 and L7 Balancers

Other Topics for System Design

- System Design — Load Balancing
- System Design — Caching
- System Design — Sharding / Data Partitioning
- System Design — Indexes
- System Design — Proxies
- System Design — Message Queues
- System Design — Redundancy and Replication
- System Design — SQL vs. NoSQL
- System Design — CAP Problem
- System Design — Consistent Hashing
- System Design — Client-Server Communication
- System Design — Storage
- System Design — Other Topics
- Object-Oriented Programming — Basic Design Patterns in C++

❤️ Thank you for reading my article, if you like the content feel free to **Subscribe**, clap👏🏻 and share it.❤️

System Design Interview | Load Balancing

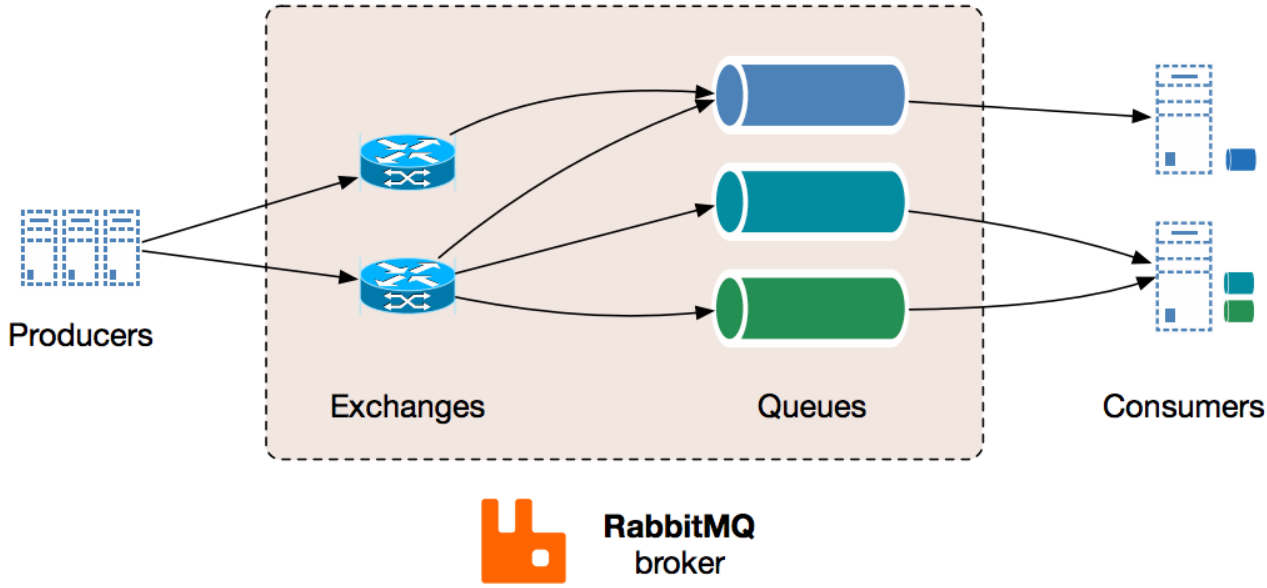
Follow

Written by Larry | Peng Yang

1.6K Followers · Editor for Computer Science Fundamentals

Software Engineer in Tokyo. Aim to understand computer science very well.
www.buymeacoffee.com/yangpeng. www.linkedin.com/in/yangpeng-tech.

More from Larry | Peng Yang and Computer Science Fundamentals



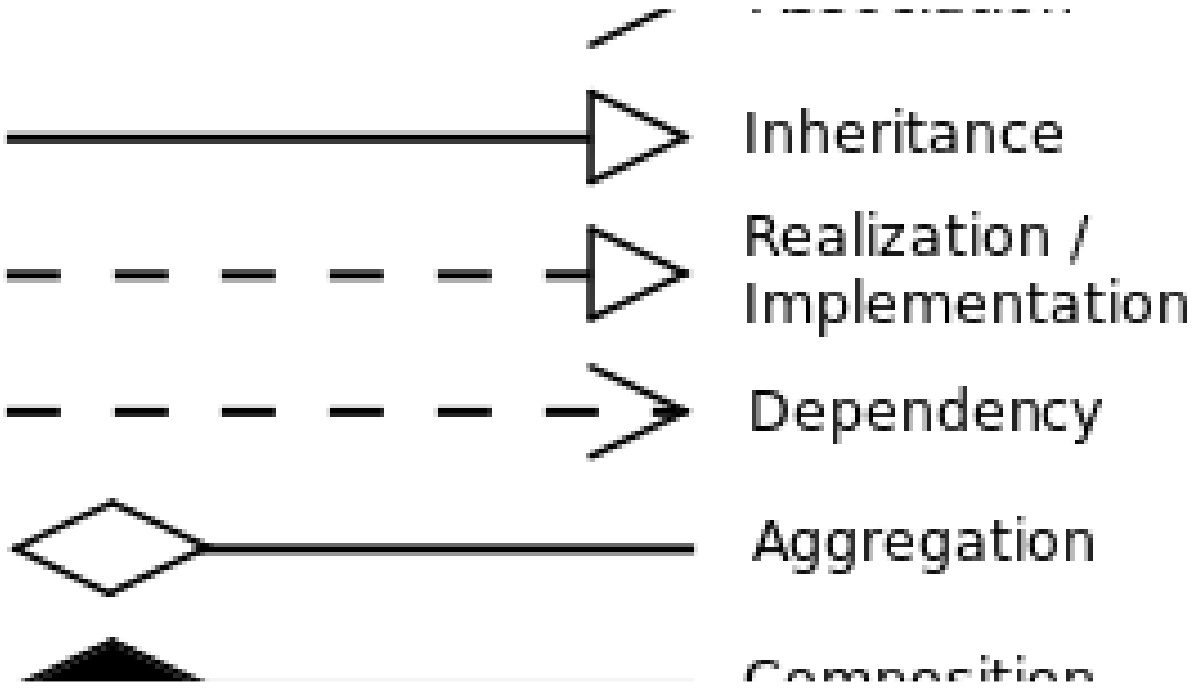
Larry | Peng Yang in Computer Science Fundamentals

System Design—Message Queues

Concepts and considerations for Message Queues in System Design

4 min read · Apr 6, 2020

355



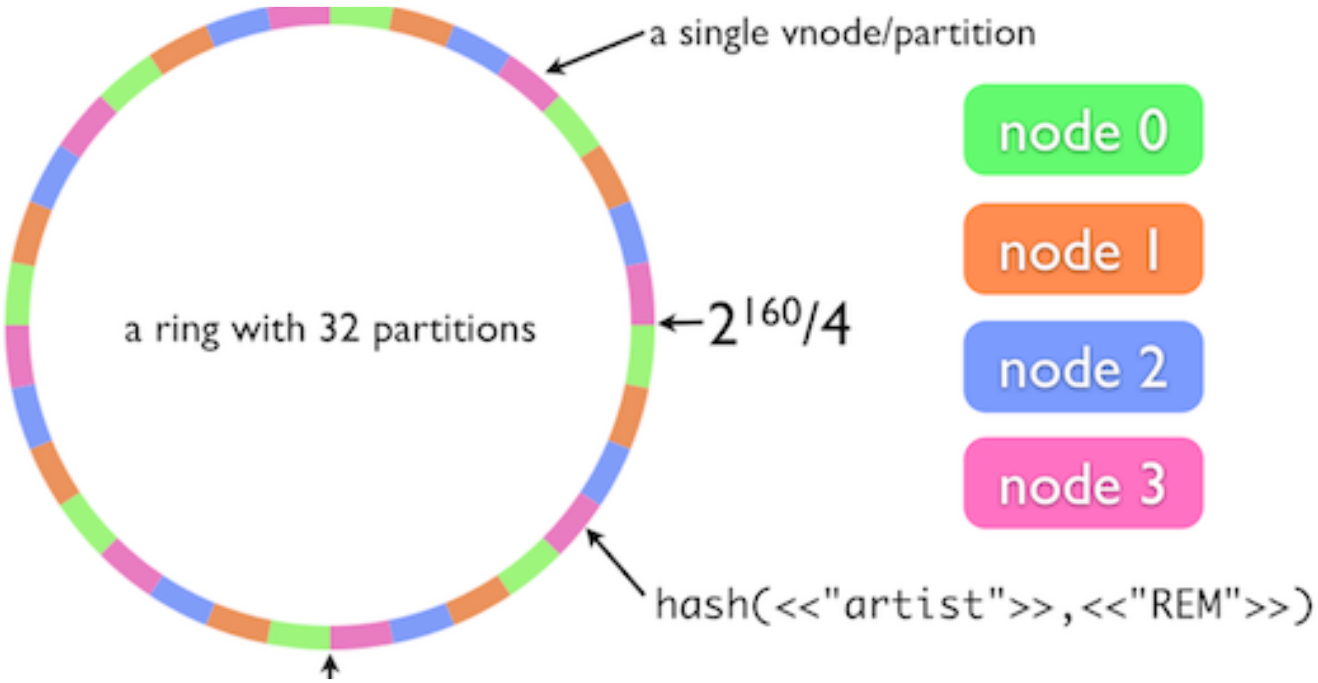
Larry | Peng Yang in Computer Science Fundamentals

Basic Design Patterns in C++

Most important design patterns in C++

23 min read · Sep 23, 2019

182



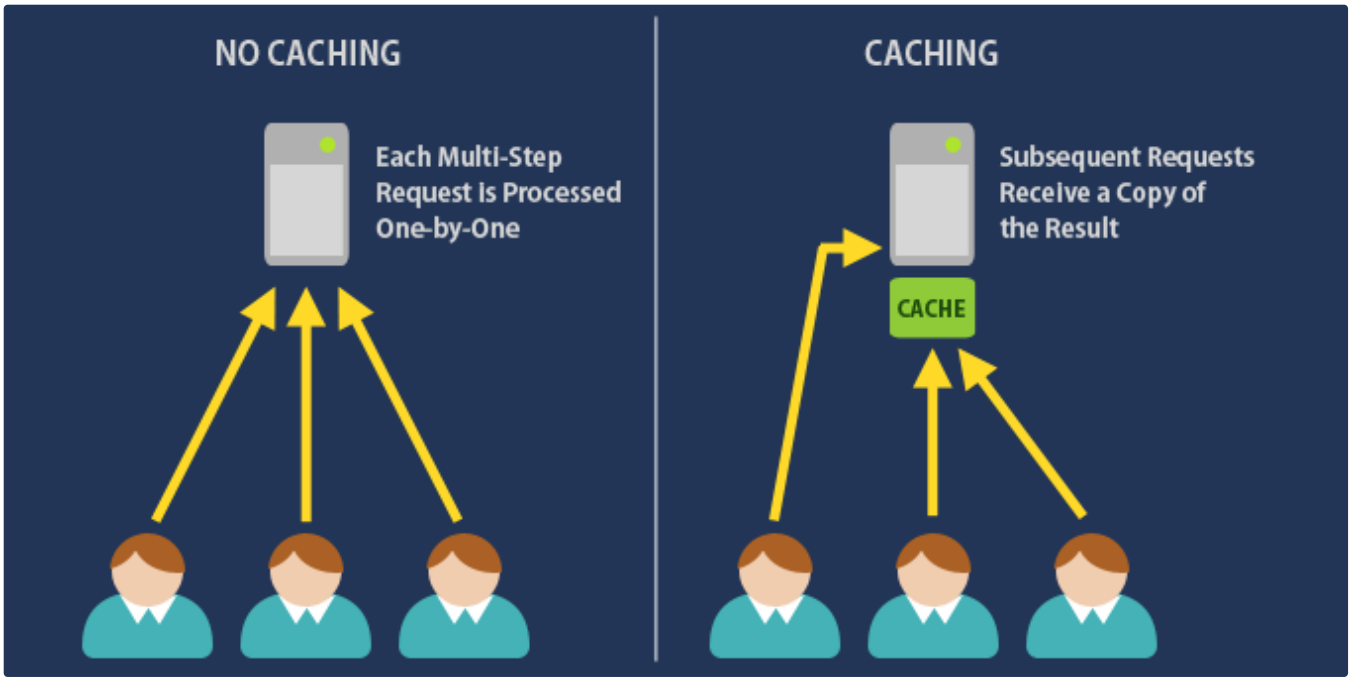
Larry | Peng Yang in Computer Science Fundamentals

System Design — Consistent Hashing

Concepts and considerations for Consistent Hashing in System Design

· 3 min read · Apr 6, 2020

22



Larry | Peng Yang in Computer Science Fundamentals

System Design—Caching

Concepts and considerations for Caching in System Design

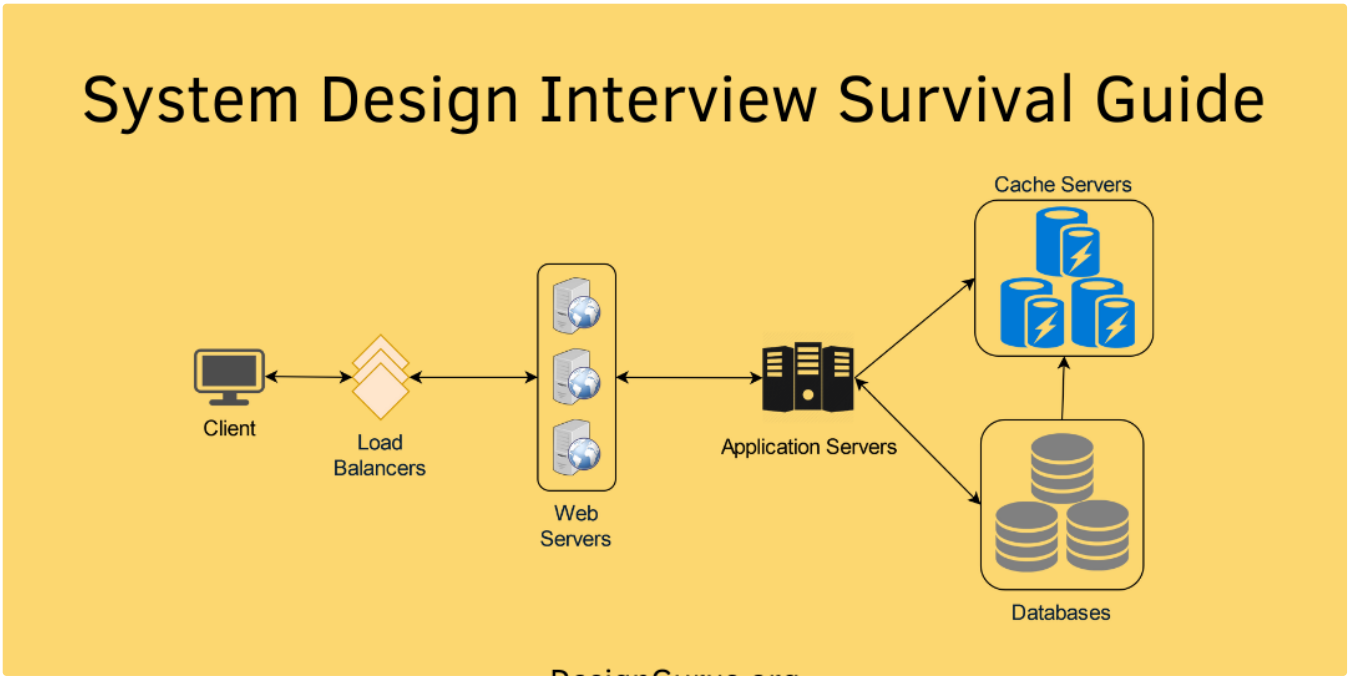
6 min read · Apr 6, 2020

79

See all from Larry | Peng Yang

See all from Computer Science Fundamentals

Recommended from Medium



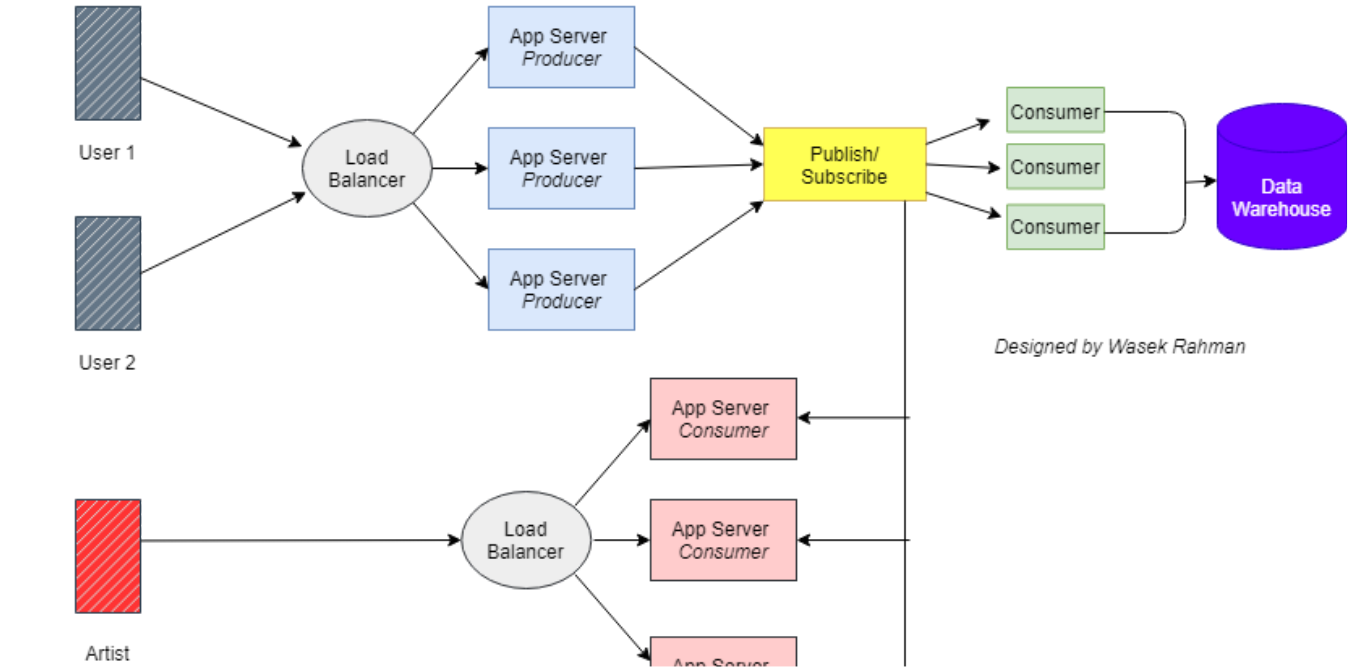
Arslan Ahmad in Level Up Coding

System Design Interview Survival Guide (2023): Preparation Strategies and Practical Tips

System Design Interview Preparation: Mastering the Art of System Design.

· 14 min read · Jan 19

1.8K 8



Wasek Rahman in Bootcamp

System Design Interview Prep: Spotify Real-Time Player

One of the most critical sections of interviewing at a top company or senior role is the System Design section where you would have to come...

✨ · 4 min read · Feb 18

112 1

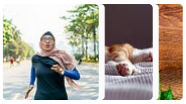
Lists

Staff Picks

374 stories · 139 saves

Stories to Help You Level-Up at Work

19 stories · 127 saves

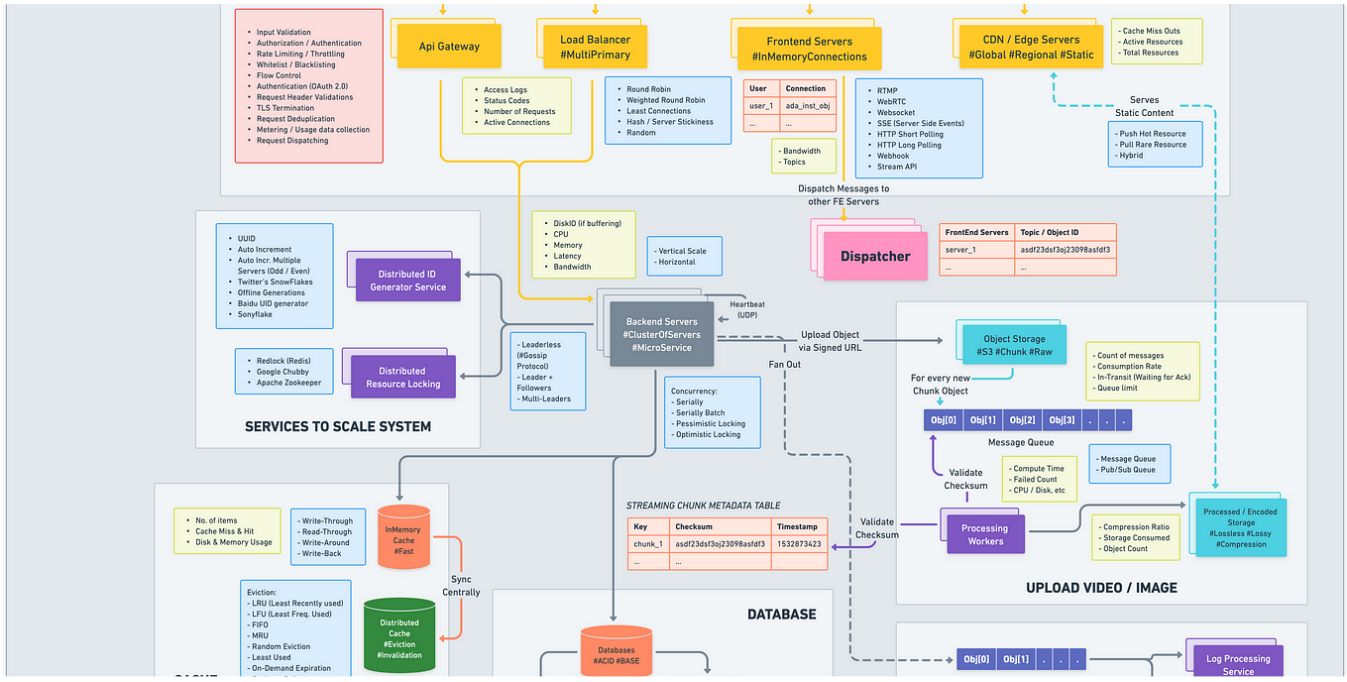


Self-Improvement 101

20 stories · 217 saves

Productivity 101

20 stories · 235 saves



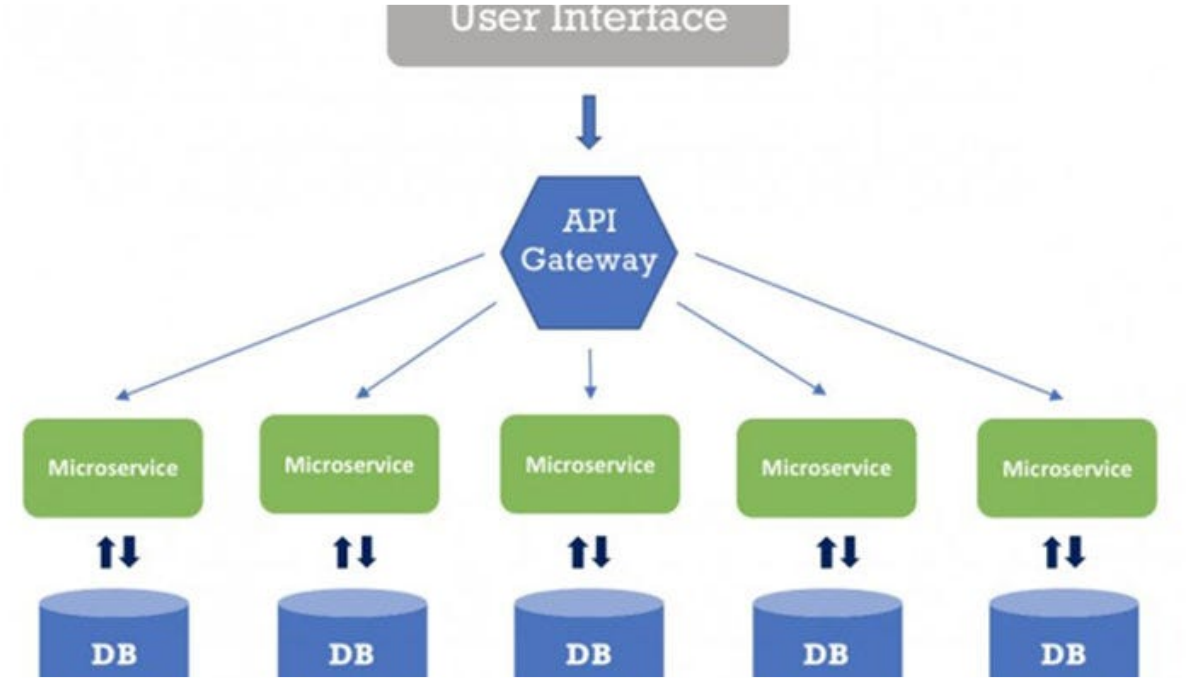
Love Sharma in Dev Genius

System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However, understanding the key concepts and components can make the...

· 9 min read · Apr 20

4.8K 37



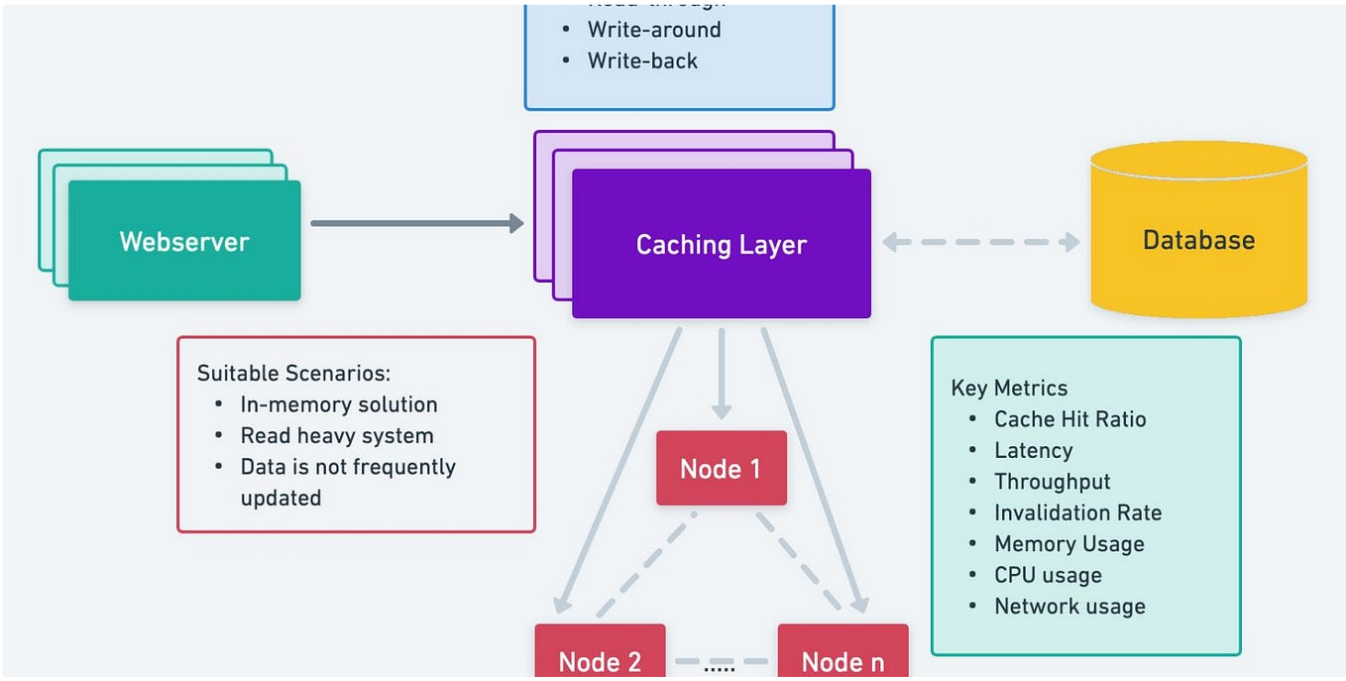
Soma in Javarevisited

50 Microservices Design and Architecture Interview Questions for Experienced Java Programmers

Preparing for Senior Java developer role where Microservices skill is required? Here are 50 questions which you should know before going...

· 11 min read · Jan 14

459 8



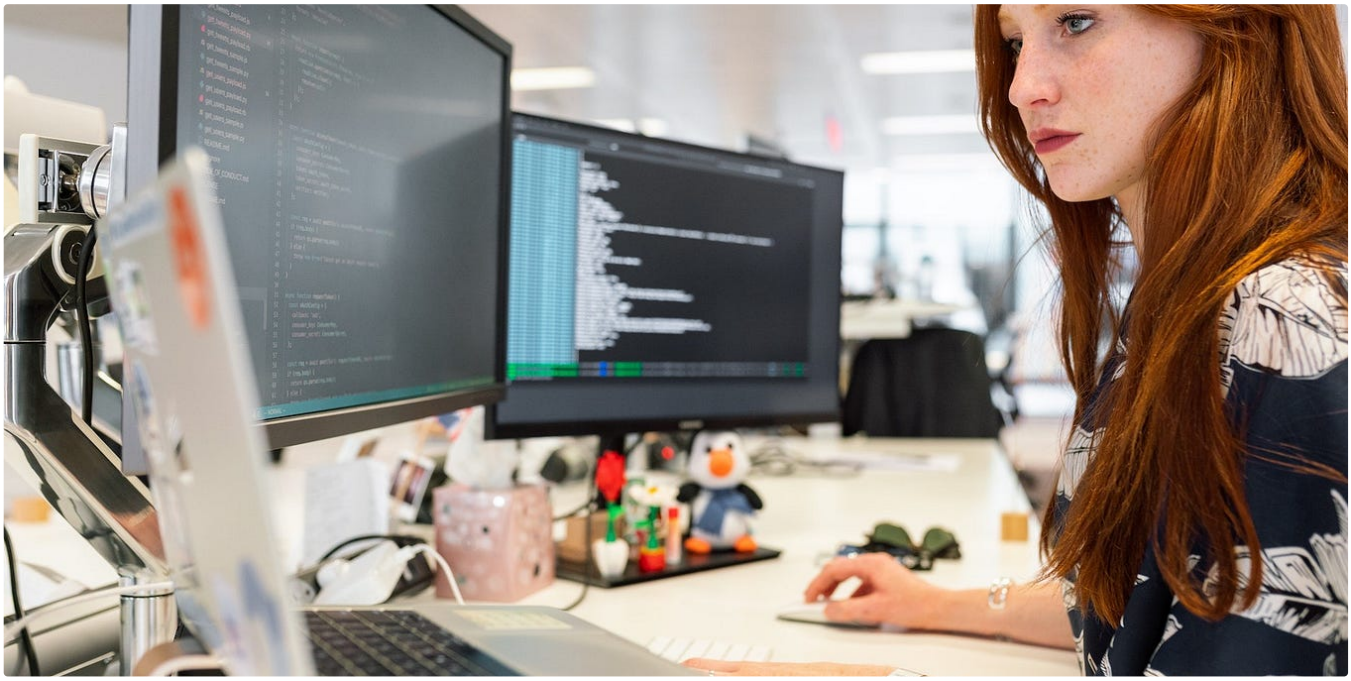
Love Sharma in Dev Genius

A Comprehensive Guide to Distributed Caching

An essential website requires a web server to receive requests and a database to write or read data. However, this simple setup will only...

· 13 min read · Feb 9

512 4



The Coding Diaries in The Coding Diaries

Why Experienced Programmers Fail Coding Interviews

A friend of mine recently joined a FAANG company as an engineering manager, and found themselves in the position of recruiting for...

· 5 min read · Nov 2, 2022

4.5K 102