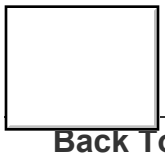


Back To Course Home

Grokking Modern System Design Interview for Engineers & Managers

0% completed

System Design Interviews

Introduction

Abstractions

Non-functional System Characteristics

Back-of-the-envelope Calculations

Building Blocks

Domain Name System

Load Balancers

Introduction to Load Balancers

Global and Local Load Balancing

Advanced Details of Load Balancers

Databases

Key-value Store

Content Delivery Network (CDN)

Sequencer

Distributed Monitoring

Monitor Server-side Errors

Monitor Client-side Errors

Distributed Cache

Distributed Messaging Queue

Pub-sub

Rate Limiter

- [ ]

Blob Store

- [ ]

Distributed Search

- [ ]

Distributed Logging

- [ ]

Distributed Task Scheduler

- [ ]

Sharded Counters

- [ ]

Concluding the Building Blocks Discussion

- [ ]

Design YouTube

- [ ]

Design Quora

- [ ]

Design Google Maps

- [ ]

Design a Proximity Service / Yelp

- [ ]

Design Uber

- [ ]

Design Twitter

- [ ]

Design Newsfeed System

Design Instagram

Design a URL Shortening Service / TinyURL

Design a Web Crawler

Design WhatsApp

Design Typeahead Suggestion

Design a Collaborative Document Editing Service / Google Docs

Spectacular Failures

Concluding Remarks

Course Certificate

Mark Course as Completed

Advanced Details of Load Balancers

Understand load balancers and their usage within a system.

We'll cover the following

- Algorithms of load balancers
  - Static versus dynamic algorithms
  - Stateful versus stateless LBs
    - Stateful load balancing
    - Stateless load balancing
- Types of load balancers
- Load balancer deployment
  - Tier-0 and tier-1 LBs
  - Tier-2 LBs
  - Tier-3 LBs
  - Practical example
- Implementation of load balancers
  - Hardware load balancers
  - Software load balancers
  - Cloud load balancers
- Conclusion

This lesson will focus on some of the well-known algorithms used in the local load balancers. We'll also understand how load balancers are connected to form a hierarchy, sharing work across different tiers of LBs.

Algorithms of load balancers#

Load balancers distribute client requests according to an algorithm. Some well-known algorithms are given below:

- **Round-robin scheduling**: In this algorithm, each request is forwarded to a server in the pool in a repeating sequential manner.
- **Weighted round-robin**: If some servers have a higher capability of serving clients' requests, then it's preferred to use a weighted round-robin algorithm. In a

weighted round-robin algorithm, each node is assigned a weight. LBs forward clients' requests according to the weight of the node. The higher the weight, the higher the number of assignments.

- **Least connections**: In certain cases, even if all the servers have the same capacity to serve clients, uneven load on certain servers is still a possibility. For example, some clients may have a request that requires longer to serve. Or some clients may have subsequent requests on the same connection. In that case, we can use algorithms like least connections where newer arriving requests are assigned to servers with fewer existing connections. LBs keep a state of the number and mapping of existing connections in such a scenario. We'll discuss more about state maintenance later in the lesson.
- **Least response time**: In performance-sensitive services, algorithms such as least response time are required. This algorithm ensures that the server with the least response time is requested to serve the clients.
- **IP hash**: Some applications provide a different level of service to users based on their IP addresses. In that case, hashing the IP address is performed to assign users' requests to servers.
- **URL hash**: It may be possible that some services within the application are provided by specific servers only. In that case, a client requesting service from a URL is assigned to a certain cluster or set of servers. The URL hashing algorithm is used in those scenarios.

There are other algorithms also, like randomized or weighted least connections algorithms.

Static versus dynamic algorithms#

Algorithms can be static or dynamic depending on the machine's state. Let's look at each of the categories individually:

Static algorithms don't consider the changing state of the servers. Therefore, task assignment is carried out based on existing knowledge about the server's configuration. Naturally, these algorithms aren't complex, and they get implemented in a single router

or commodity machine where all the requests arrive.

Dynamic algorithms are algorithms that consider the current or recent state of the servers. Dynamic algorithms maintain state by communicating with the server, which adds a communication overhead. State maintenance makes the design of the algorithm much more complicated.

Dynamic algorithms require different load balancing servers to communicate with each other to exchange information. Therefore, dynamic algorithms can be modular because no single entity will do the decision-making. Although this adds complexity to dynamic algorithms, it results in improved forwarding decisions. Finally, dynamic algorithms monitor the health of the servers and forward requests to active servers only.

> **Note:** In practice, dynamic algorithms provide far better results because they maintain a state of serving hosts and are, therefore, worth the effort and complexity.

Stateful versus stateless LBs#

While static and dynamic algorithms are required to consider the health of the hosting servers, a state is maintained to hold session information of different clients with hosting servers.

If the session information isn't kept at a lower layer (distributed cache or database), load balancers are used to keep the session information. Below, we describe two ways of handling session maintenance through LBs:

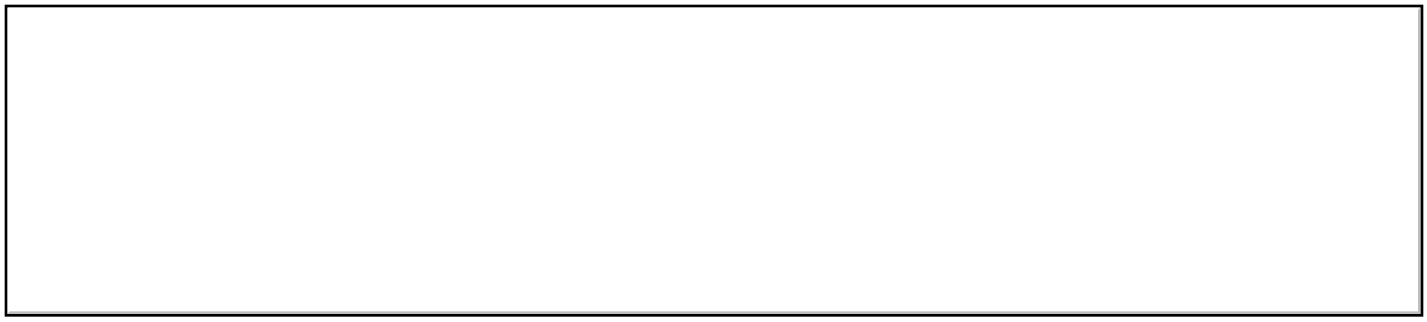
- Stateful
- Stateless

Stateful load balancing#

As the name indicates, **stateful load balancing** involves maintaining a state of the

sessions established between clients and hosting servers. The stateful LB incorporates state information in its algorithm to perform load balancing.

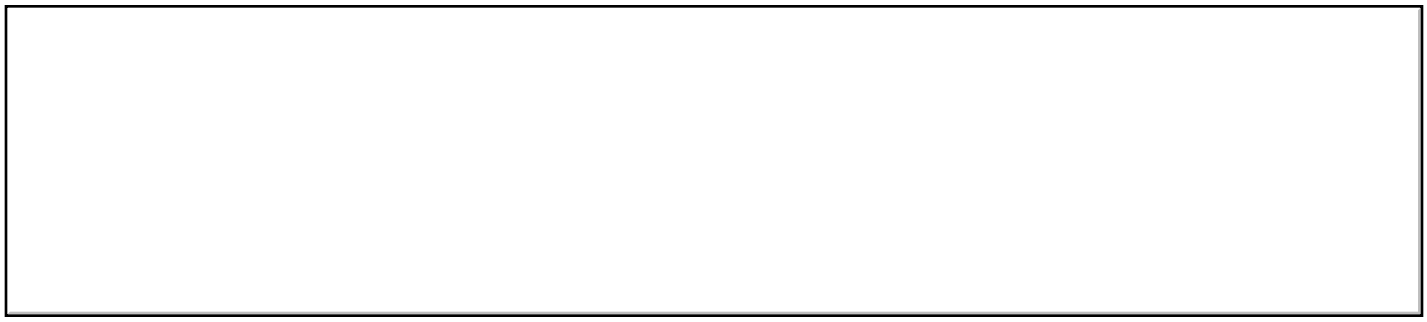
Essentially, the stateful LBs retain a data structure that maps incoming clients to hosting servers. Stateful LBs increase complexity and limit scalability because session information of all the clients is maintained across all the load balancers. That is, load balancers share their state information with each other to make forwarding decisions.



Stateful load balancing

Stateless load balancing#

Stateless load balancing maintains no state and is, therefore, faster and lightweight. Stateless LBs use consistent hashing to make forwarding decisions. However, if infrastructure changes (for example, a new application server joining), stateless LBs may not be as resilient as stateful LBs because consistent hashing alone isn't enough to route a request to the correct application server. Therefore, a local state may still be required along with consistent hashing.



Stateless load balancers using hash buckets to map requests to end servers

Therefore, a state maintained across different load balancers is considered as stateful

load balancing. Whereas, a state maintained within a load balancer for internal use is assumed as stateless load balancing.

Types of load balancers#

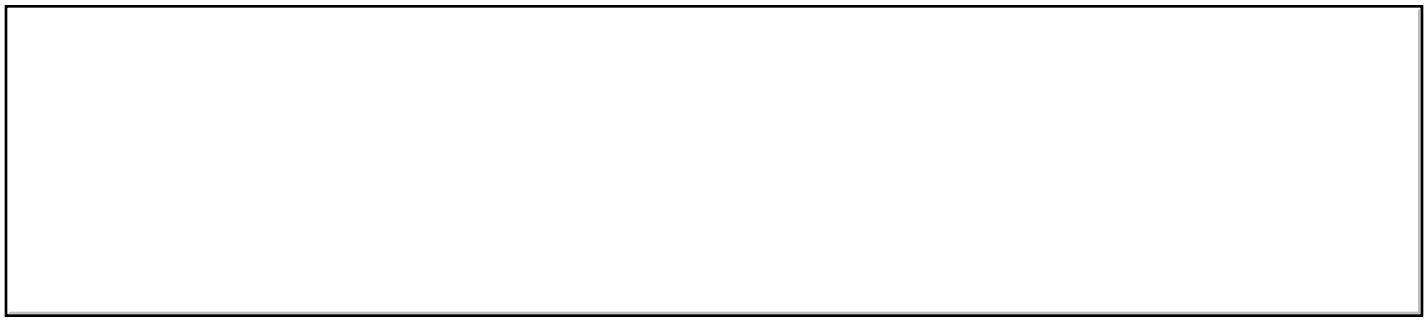
Depending on the requirements, load balancing can be performed at the network/transport and application layer of the open systems interconnection (OSI) layers.

- **Layer 4 load balancers**: Layer 4 refers to the load balancing performed on the basis of transport protocols like TCP and UDP. These types of LBs maintain connection/session with the clients and ensure that the same (TCP/UDP) communication ends up being forwarded to the same back-end server. Even though TLS termination is performed at layer 7 LBs, some layer 4 LBs also support it.
- **Layer 7 load balancers**: Layer 7 load balancers are based on the data of application layer protocols. It's possible to make application-aware forwarding decisions based on HTTP headers, URLs, cookies, and other application-specific data—for example, user ID. Apart from performing TLS termination, these LBs can take responsibilities like rate limiting users, HTTP routing, and header rewriting.

> **Note:** Layer 7 load balancers are smart in terms of inspection. However layer 4 load balancers are faster in terms of processing.

Load balancer deployment#

We discussed the trade-offs of load balancing performed at different OSI layers. In practice, however, a single layer LB isn't enough for a large data center. In fact, multiple layers of load balancers coordinate to take informed forwarding decisions. A traditional data center may have a three-tier LB as shown below:



Three-tier load balancer in a typical data center

Tier-0 and tier-1 LBs#

If DNS can be considered as the tier-0 load balancer, equal cost multipath (ECMP) routers are the tier-1 LBs. From the name of ECMP, it's evident that this layer divides incoming traffic on the basis of IP or some other algorithm like round-robin or weighted round-robin. Tier-1 LBs will balance the load across different paths to higher tiers of load balancers.

ECMP routers play a vital role in the horizontal scalability of the higher-tier LBs.

Tier-2 LBs#

The second tier of LBs include layer 4 load balancers. Tier-2 LBs make sure that for any connection, all incoming packets are forwarded to the same tier-3 LBs. To achieve this goal, a technique like consistent hashing can be utilized. But in case of any changes to the infrastructure, consistent hashing may not be enough. Therefore, we have to maintain a local or global state as we'll see in the coming sections of the lesson.

Tier-2 load balancers can be considered the glue between tier-1 and tier-3 LBs. Excluding tier-2 LBs could result in erroneous forwarding decisions in case of failures or dynamic scaling of LBs.

Tier-3 LBs#

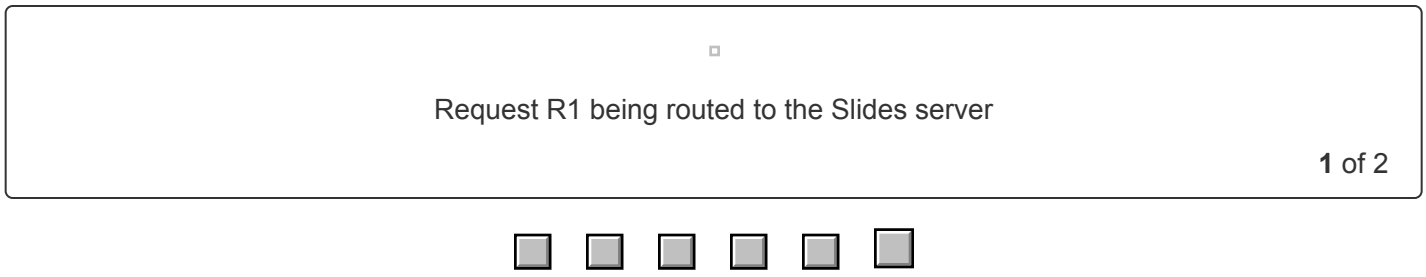
Layer 7 LBs provide services at tier 3. Since these LBs are in direct contact with the back-end servers, they perform health monitoring of servers at HTTP level. This tier enables

scalability by evenly distributing requests among the set of healthy back-end servers and provides high availability by monitoring the health of servers directly. This tier also reduces the burden on end-servers by handling low-level details like TCP-congestion control protocols, the discovery of Path MTU (maximum transmission unit), the difference in application protocol between client and back-end servers, and so on. The idea is to leave the computation and data serving to the application servers and effectively utilize load balancing commodity machines for trivial tasks. In some cases, layer 7 LBs are at the same level as the service hosts.

> To summarize, tier 1 balances the load among the load balancers themselves. Tier 2 enables a smooth transition from tier 1 to tier 3 in case of failures, whereas tier 3 does the actual load balancing between back-end servers. Each tier performs other tasks to reduce the burden on end-servers.

Practical example#

Let's look at an example where requests from a client come in and get forwarded to different application servers based on the application data inside the client's network packets.



Let's look at the illustration above in the following steps:

1. R_1 indicates request 1 coming through one of the ECMP routers (tier-1 LBs).
2. ECMP routers forward R_1 to any of the three available tier-2 LBs using a round-robin algorithm. Tier-2 LBs take a hash of the source IP address (IP_c) and forward

the packet to the next tier of LBs.

3. Tier-3, upon receiving the packet, offloads TLS and reads the HTTP(S) data. By observing the requested URL, it forwards the request to the server handling requests for `slides`.

R_2 takes the same path but a different end-server because the requested URL contains `document` instead of `slides`. Tier-3 LBs are preconfigured to forward requests to application servers based on the application data. For instance, a typical HAProxy server can have the following configuration in tier-3 LBs:

```
mode HTTP                                  //define which mode the LB will work on, T
acl slidesApp path_end -i /presentation //define a category of applications if the
use_backend slidesServers if slidesApp  // use a set of backend servers if the req
backend slidesServers                      // listing servers serving slidesApp
server slides1 192.168.12.1:80             //using slides1 server to serve slidesApp.
```

HAProxy sample configuration for layer 7 load balancers

Quiz

Question 1

After a request reaches a back-end server, should the response be routed back through each tier of the load balancers?

Show Answer

1 of 4

Implementation of load balancers#

Different kinds of load balancers can be implemented depending on the number of incoming requests, organization, and application-specific requirements:

Hardware load balancers#

Load balancers were introduced in the 1990s as hardware devices. Hardware load balancers work as stand-alone devices and are quite expensive. Nonetheless, they have their performance benefits and are able to handle a lot of concurrent users. Configuration of hardware-based solutions is problematic because it requires additional human resources. Therefore, they aren't the go-to solutions even for large enterprises that can afford them. Availability can be an issue with hardware load balancers because additional hardware will be required to failover to in case of failures. Finally, hardware LBs can have higher maintenance/operational costs and compatibility issues making them less flexible. Not to mention that hardware LBs have vendor locks as well.

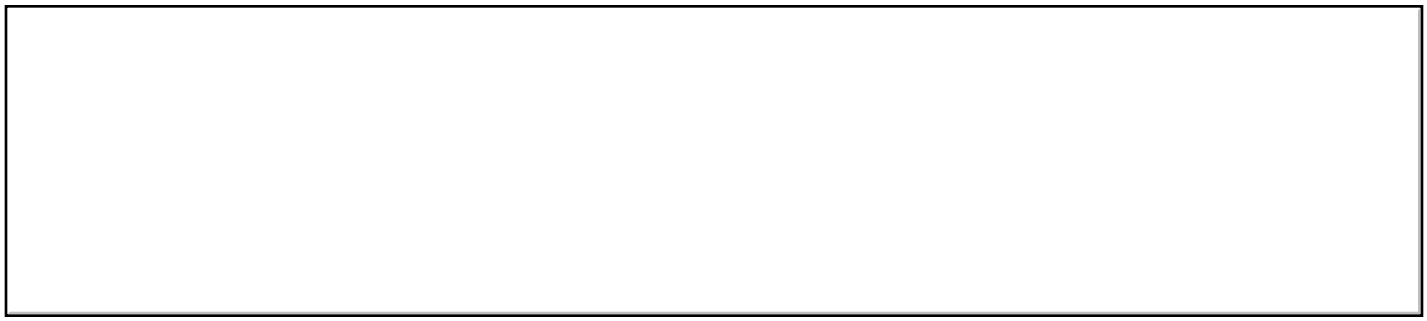
Software load balancers#

Software load balancers are becoming increasingly popular because of their flexibility, programmability, and cost-effectiveness. That's all possible because they're implemented on commodity hardware. Software LBs scale well as requirements grow. Availability won't be an issue with software LBs because small additional costs are required to implement shadow load balancers on commodity hardware. Additionally, software LBs can provide predictive analysis that can help prepare for future traffic patterns.

Cloud load balancers#

With the advent of the field of cloud computing, Load Balancers as a Service (LBaaS) has been introduced. This is where cloud owners provide load balancing services. Users pay according to their usage or the service-level agreement (SLA) with the cloud provider. Cloud-based LBs may not necessarily replace a local on-premise load balancing facility, but they can perform global traffic management between different zones. Primary

advantages of such load balancers include ease of use, management, metered cost, flexibility in terms of usage, auditing, and monitoring services to improve business decisions. An example of how cloud-based LBs can provide GSLB is given below:

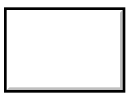


GSLB is obtained through LBaaS, and the regions contain data centers that are the property of application providers

> **Note:** Another interesting implementation of load balancers comes in the form of **client-side load balancing**. Client-side load balancing is suited where there are numerous services, each with many instances (such as load balancing in Twitter). Our focus, however, is on traditional load balancers because most three-tier applications employ these in their design.

Conclusion#

LBs have come a long way to become a service offered in the cloud, starting since their inception in the form of hardware. They're a key component of any enterprise-level service. Horizontal scalability of hosting servers will always require a good load balancing layer capable of providing load balancing, session maintenance, TLS offloading, service discovery, and more.



Global and Local Load Balancing

Next

Next

Introduction to Databases

Mark as Completed

Report an Issue