Load Balancing

SYT - 5A HIT

Martin Haidn, Nikolaus Schrack

December 5, 2014

Contents

1 Instruction

The concept of Load Balancing is not new in the server and network space. There are different types of Load Balancing. A Router, for example, distributes the traffic across multiple paths to the same destination. A Server Load Balancer, on the other hand, distributes traffic among server resources rather then network resources. [3]

Typically it is used for balancing traffic over multiple servers and acting as one web front-end. The user usually doesn't know about the existence of multiple backend servers because it seems as there is only one server. The processing load is shared across many nodes, rather than just a single. Thus the performance during times of high activity increases. [4]

1.1 The Need and Goals for Load Balancing

The main goal of Load Balancing is to distribute workload across resources. It is suppose to optimize the traffic, maximize throughput, minimize response time and try not to overload any single resources.

Since the Internet and Intranet have gotten so important for businesses, Load Balancing has become a very essential component for networks and servers. If the network goes down or works poorly, it can critically damage a business. Especially for companies with e-commerce, a long respond time or in the worst case an outage would leave frustrated customers and huge money loss. For other companies, losing access to email would have devastating impact on their business. [3]

There are a couple challenges that Load Balancing tries to solve.

Scalability. The problem of scaling computer capacity is huge. In the old days, if a server wasn't good enough to run an application, they simply bought a more powerful. Nowadays the Load Balancing systems work with multiple servers. It uses load-distribution algorithms to distribute the client request among all the servers. If the traffic becomes more and more, another server can be put into the system easily. [3]

Availability. The health of the servers and application is check by the Load Balancer continuously. If they fail the health check they don't get any request from the Load Balancer until they are fixed. The requests are sent to healthy servers. [3]

Manageability. When software on servers gets upgraded they need to be taken down. Although this can be scheduled so it doesn't collate with the off-peak hours it needs downtime. Some business can't afford downtime and if your application is used around the world there is usually always a lot of traffic. A Load Balancer can simply give the request to another server and the wait for the connection that still exists to be closed until he safely shut downs the server that needs an upgrade. Load Balancer can also help manage with large amounts of content, known as content management. Some Web servers have so much content that can't fit into one server. The application can be distributed into multiple servers.[3]

Security. Since Load Balancer are the front end of a server farm they can protect the servers from malicious users. They normally have security features that can stop certain types of attacks. Usually the back-end servers have a private IP address so they aren't rout-able on the Internet. Anyone that is in the public Internet must go through a network address translation (NAT) in order to communicate with the servers. Because the Load Balancer can do the NAT, every connection is forced to go through it.[3]

There are two main reasons for Load Balancing:

1. Limiting your points of failure
2. Load Distribution

Limiting your points of failure
Limiting your points of failure is essential for every IT Department. The uptime increases by the limitation of available points of failure. *"If you load balance between two or more identical nodes, in the event that one of the nodes in your cluster experiences any kind of hardware or software failure the traffic can be redistributed to the other nodes keeping your site up."* [4] Those identical servers can independently handle the traffic. If there is a failure in one of them the site still runs.

Load Distribution
A single server configuration can only hold a certain amount of traffic, even the most robust, high-end server. But at the traffic peaks of the application it still needs to work just fine. As website grows popularity, multiple servers with Load Balancing are necessary. [4]

1.2 Applications

With the rise of the Internet, the network becomes very important. *"As the Internet connects the world and the Intranet becomes the operational backbone for businesses, the IT infrastructure can be thought of as two types of equipment: computer that function as a client and/or a server, and switches/routers that connect the computers"* [3]

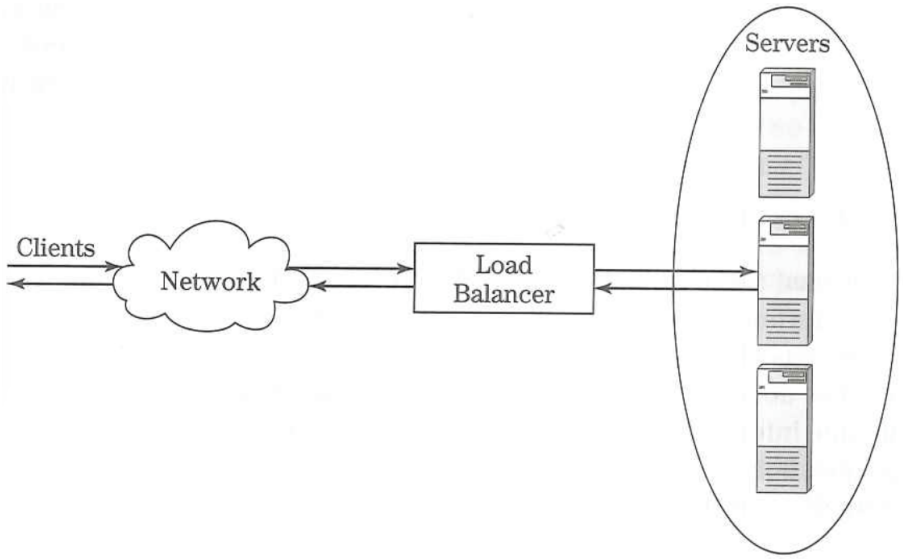

Figure 1: Load balancer's connections between servers and client[3]

The graphic shows that the Load Balancer is the connection between the clients and the servers. The Load Balancer understands many higher-layer protocols, so they can communicate with servers intelligently. They

also understand networking protocols, so they can work with the network effectively.

Load Balancer have four big applications

1. Server load balancing
2. Global load balancing
3. Firewall load balancing
4. Transparent cache switching

Server load balancing deals with multiple servers to scale beyond the capacity of one server and to handle a server failure. Global load balancing directs users to different data centers consisting of server farms so they can provide quicker response and handle a data center failure. The Firewall Load balancing can distribute load between multiple firewalls to again, handle a failure of one of them. Transparent cache switching directs traffic to caches to minimize the response time.

There are three main forms of products for load balancing.

1. Software load-balancing
2. Appliances
3. Switches

Software load balancing are products that run on load balancing servers. They have algorithms to coordinate the traffic among them. For example Apache Module mod_proxy is a popular tool. Nginx would be another tool that allows software load balancing. They are mostly for free which is great if one runs on a low budget.

Appliances are a all-in-one product that include necessary hardware and software to do web switching. It may has some special operating system and custom hardware. For example Cisco has hardware appliances that handle load balancing for you.

Switches have to their traditional functionality in OSI Layer 2/3, are also able to do load balancing on Layer 4-7. Mostly though, they a significant amount of work done by software. For example Zen Load Balancer and Cisco as well have switches appliances that handle load balancing for you.

1.3 Use Cases and Examples

When big companies face the problem of Load Balancing, they have to focus on every aspect of their IT infrastructure. How would a business like *Google* handle such a challenge? They have millions of request every day and need to have their service available 24/7.

Let's look at the website request "google.at".

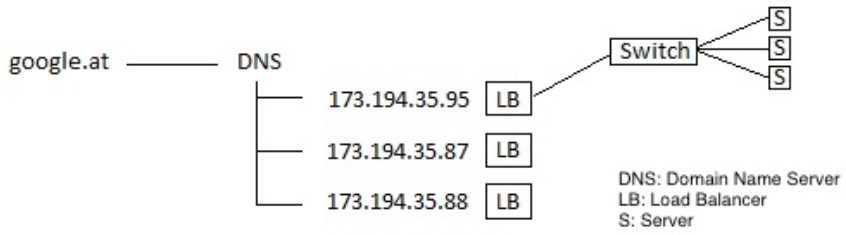

Figure 2: How Load Balancing works in the big picture

The DNS Server answers with a list of IP addresses of Load Balancers. In this example it gives three IP's back. The browser would use the first one by default. The order of the list varies at every call which is already kind of a load balancing because the traffic is distributed among the list of servers. At the server centers of big companies, the application is spread amongst the servers. So not every server can do everything. The Load Balancer can look into the request and then give it to the right switch, which has servers that can answer the request. The Load Balancers are the only one with a public IP. The servers that actually answer the request are in the private network. The way back to the client can either go through the Load Balancer or from the server to a NAT device an then to the client.

2 Basic Concepts

2.1 Networking Fundamentals

The OSI model contains seven layers and every single one provides it's own functionality and data.
If we take a closer look on the deeper layers like data link and network, which is representative for layer two and three, we can see that their header information contains IP and MAC addresses. These addresses can be used to decide where a package has to be send when it's revived by a switch.
This basic concept of routing packages builds the fundament for load balancing. It's about making a decision if, and where the data has to go. [3]

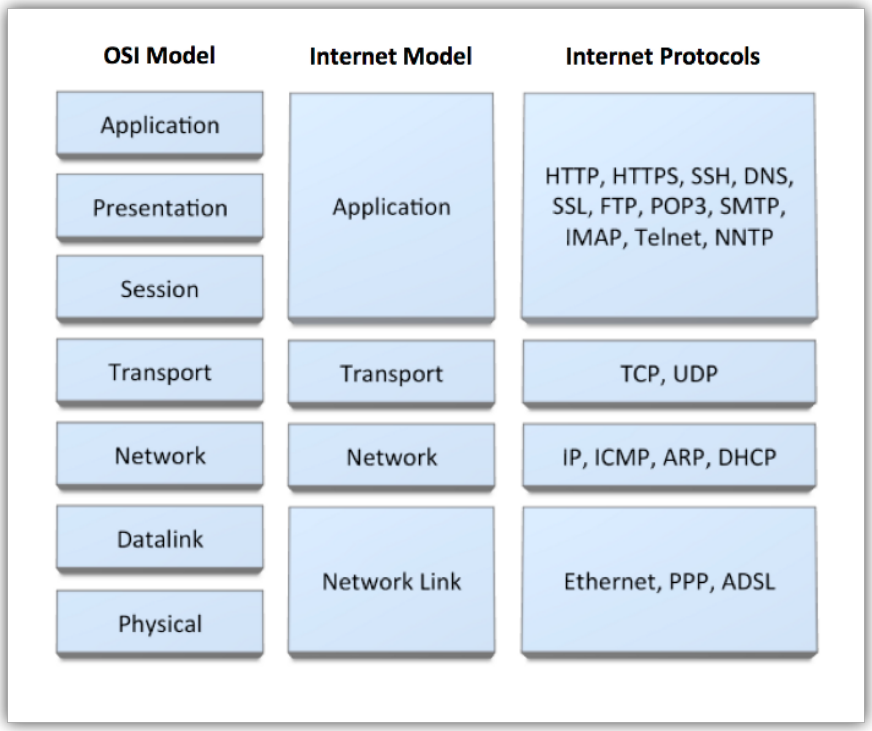

Figure 3: OSI-TCP Model [2]

Layer-4 Loadbalancing
Using the transport layer the load can be balanced from a router if a larger application network is initiated. Depending on the request the router can choose a responsible server a server with an appropriate hardware to process the job.

2.2 Application Layer

Layer-7 switching, which is also known as application-level load balancing, describes a way of load distribution based on the content of the client request. Parsing the requests causes a high overhead on the balancer's side, so it's scalability is limited.
The appliances which are responsible to perform layer 7 load balancing are called Application Delivery Controllers (ADC). ADC's presents a "virtual sever" to the world wide web (WWW),accepts requests and distributes them to the right application server, determined over the use of application data.

The knowledge about the requested data allows to serve specific types of content. A useful distribution of content are server- and client-side-scipts for example.

Layer 7 load balancing offers additional features through the use of ADCs. Based on the content type the balancer is able to decide which policies are applicable for the request and only executes the necessary ones. Further this can be a way to let only authorized users use specific servers or other hardware components. So application level balancing increases the efficiency of the application infrastructure because by investigating the content the jobs can be forwarded to a responsible server with an appropriate hardware performance. [6]

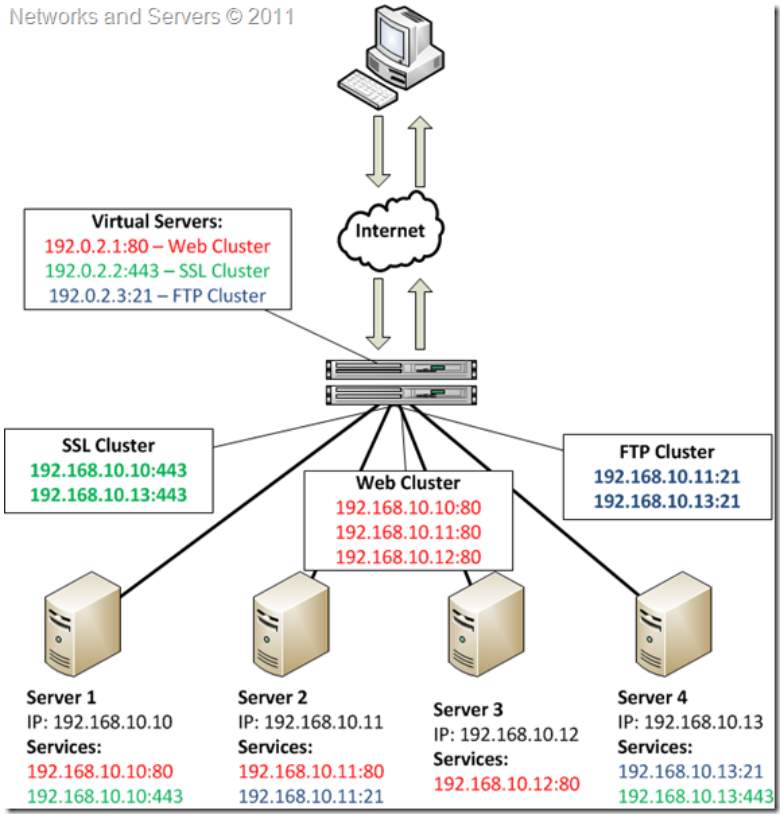

Figure 4: Application-level load balancing [6]

3 Advanced Concepts

3.1 Session Persistence

A way of load distribution where the balancer has to store the whole session information during the time of an application transaction is called Session Persistence. In addition to the right client address the balancer also has to know about the correct server, the request got forwarded to. This concept is mostly used on web services where the server has to respond with user specific content or data.
For Example you can imagine an on-line book store which provides a shopping cart function that keeps your articles during your stay. If the load balancer would not store the session information during your purchase the selected article would probably get lost or in the worst case inverted with the goods of another user.
The way to get the information which is needed is broadly based onto two sources:

TCP SYN Packet
TCP SYN is the first request the clients sends to the server if he wants to establish a new connection.
It contains the source IP address and port to identify the user and also the destination Ip and port to identify the server.

Application Request
If the user calls an application or a method on the server the request has to be defined in the send packages.
This information is used by the balancer to forward the request to the right providing server.

3.2 URL Switching

Services which provide a huge amount of information may have the problem that a single server cant hold the whole available content. The content has to be divided among a few servers and therefore URL Switching is used.
Servers which are responsible for the same kind of information can be combined to a group. This makes it more easy to address the distributed contents and map the appropriate requests. The requested content from the client gets identified by names or values in the URL. This is accomplished by specifying the URL switching policies on the load balancer.

Seperating Static and Dynamic Content
Another way of URL switching is to separate the static and dynamic content in server farms. Static content is defined as information that does not change very often and can be separated from the dynamic content which may has to be generated per request.

Current URL and Cookie Switching
After our balancer has detected what server group to choose because of the requested URL we might need a way to stay in communication with the content server. Therefore a cookie-read method can be used whereby a cookie gets inserted from the server which can be read from the balancer. If no cookie is found the load balancer detects the request as new session and returns to it's URL switching policy. [3]

3.3 SSL Termination

SSL Termination which is also called SSL offload describes the ability of a balancer to establish a secure tunnel with the client. For this purpose an SSL certificate is required which can be self-generated or signed by a liable authority.
SSL termination is often used for Layer 7 trickery such as cookie insertion. [6]

4 Principles of Web Distributed Systems Design

4.1 Contents

A thing that should be thought of is the detention of data which is necessary for your web application. Most users assume a very short querying time when loading a page, therefore it's important to store the data in a place where a quick access is guaranteed.

The Aoasabook Org. shows an accurate example based on a picture sharing platform like Flickr.
The challenge defines in offering a fast up- and download facility which is able to sustain a huge count of users. The plan is to distribute the process of delivering pictures by distributing the request handler from the real image server. This brings the advantage of creating a high scalability on the side of the image database.

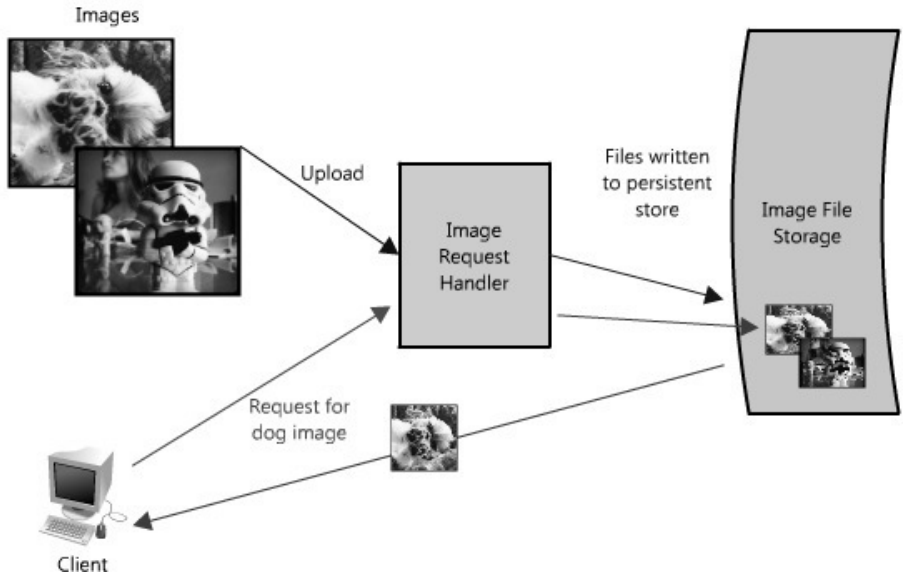

Figure 5: Simplified architecture diagram for image hosting application [5]

5 Service

Another big step in designing web applications is trying to picture every component, as it's own service. By decoupling functionality and clearly define interfaces the scalability gets raised and the ability to service and enlarge parts of the application is given. This is the reason why we talk about Service Oriented Architecture (SOA).
Every single brick of the application has it's inner functional context and the interaction with everything outside is handled over an abstract interface like an public-facing API.

Now that the clear relations in our application network are defined we want it to take effect on our example. Because most of the network intern rules define a 3:1 download-, upload-ratio the process of retrieving images and store them on the server can impact our requirement on a fast upload. To solve this issue the image request handler gets split into two fundamental parts of it's job namely reading and writing files. We are splitting the image request handler into a read and write part which The change in our application network allows us to

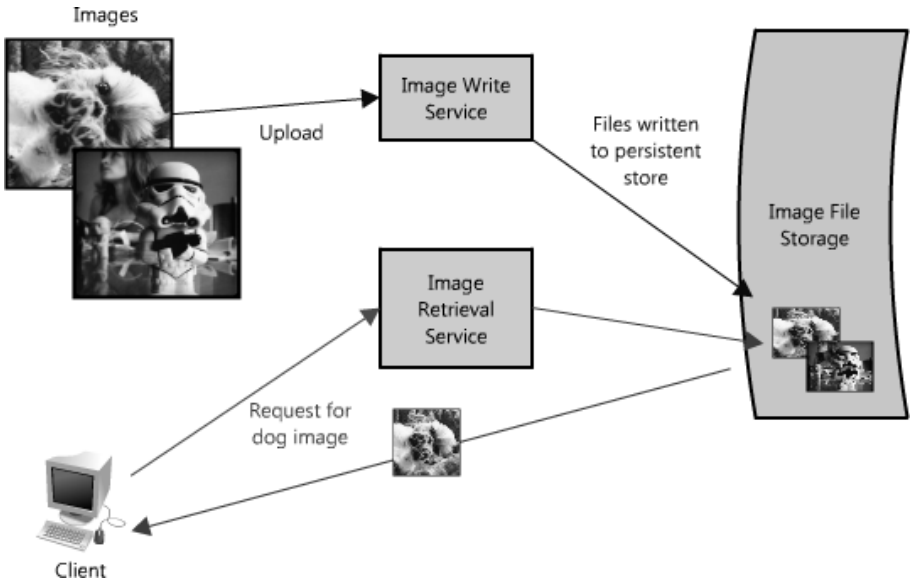

Figure 6: Splitting out reads and writes [5]

change both parts independently from another, helps to make clear what happens in each step and enables an efficient way to troubleshoot bugs and problems in the application.
Another benefit is that we can now apply balancing and scheduling concepts on each part which fits best concerning the offered service.
For example we could cache the most popular loaded images to deliver them faster and even when one of the services is overloaded, it is taking no effect to the other.

6 Caches

Caches are a lot of faster than a normal database request. It takes advantage of a simple idea: recently requested data is likely to be requested again. A cache has limited amount of space but it is faster and contains recently accessed items. They can be used at all levels in architecture, but mostly at the front-end so they can answer quickly and don't get in the way of the lower levels.

This following graphic shows the connection between cash, server and database.

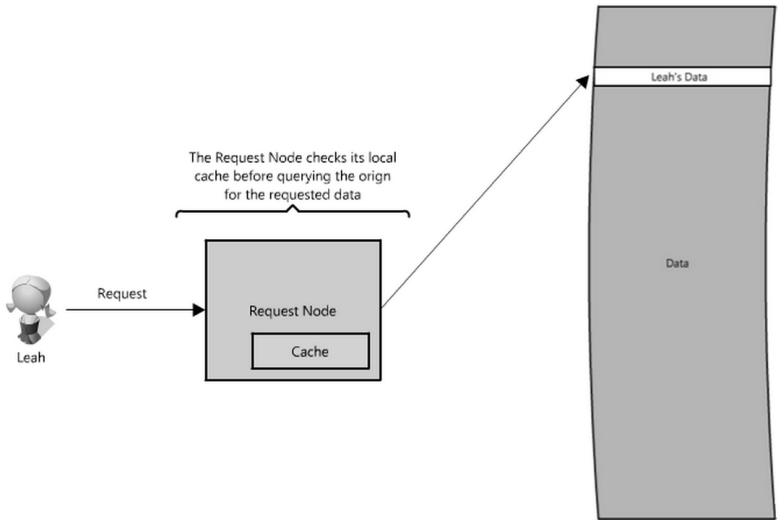

Figure 7: One Cache with a Database [5]

The node checks its cache and sees if the data is there. If it isn't, it queries it from the database. Now this is great but for most applications there are multiple Request Nodes.

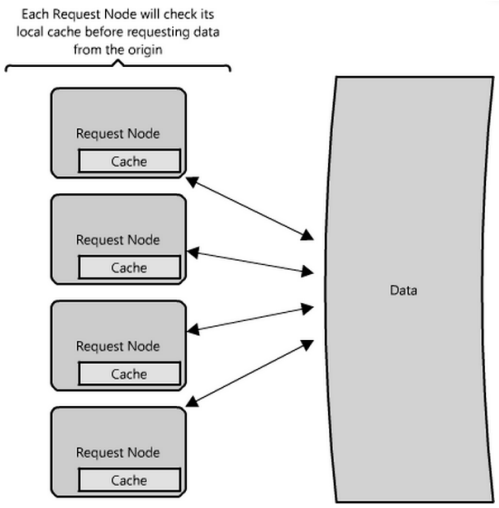

Figure 8: Multiple Cache with a Database [5]

As seen in the graphic above, we now have multiple servers with caches. But when the Load Balancer randomly distributes the request to the nodes, the same request will go to different nodes. Therefore we would have a lot of cache misses. To handle this problem we there are two strategies: Global and Distributed Caches.

6.1 Global Cache

In the global cache, all the nodes have the same cache. The nodes request the cache like it is a local one. The disadvantage of this method is, that a single cache gets overwhelmed by the numbers of requests and clients easier. However, in some architectures it can be very effective. For example with specialized hardware that make the cache very fast, or a fixed data sheet that is cached.

Following graphic shows how this looks works:

When a request comes the node looks into the global cache. If it isn't there, the global cache retrieves it

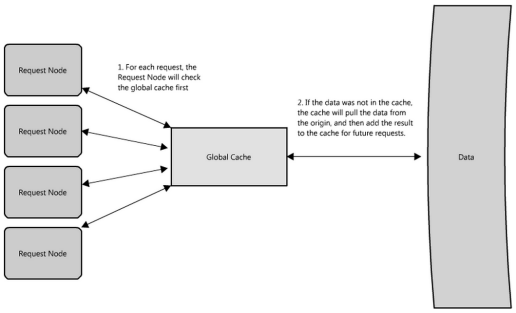

Figure 9: Global Cache where the Cache pulls from the database [5]

from the database. Note that the cache itself queries the data.

There is a second way to do this.

Now the node request the data when it isn't in the cache and then it goes to the cache.

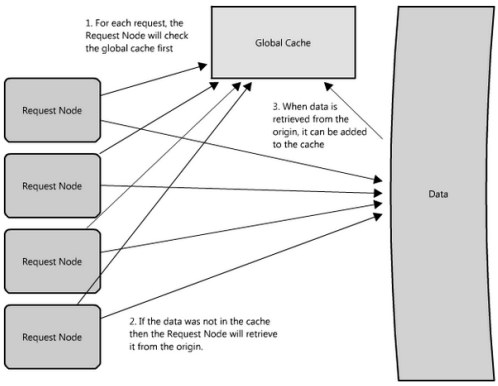

Figure 10: Global Cache where the Nodes pulls from the database[5]

Mostly the first type is used, because if the nodes want the same data, they all query it from the database, instead of just the global cache.

6.2 Distributed Cache

In a distributed cache, every node has its own part of the cache. *"...if a refrigerator acts as a cache to the grocery store, a distributed cache is like putting your food in several locations-your fridge, cupboards, and lunch box-convenient locations for retrieving snacks from, without a trip to the store."*[5] Usually the cache is divided up using a consistent hashing function. If the node looks for a certain piece of data it can find it through the hash key. In this case the nodes send the request to the node with the right cache.

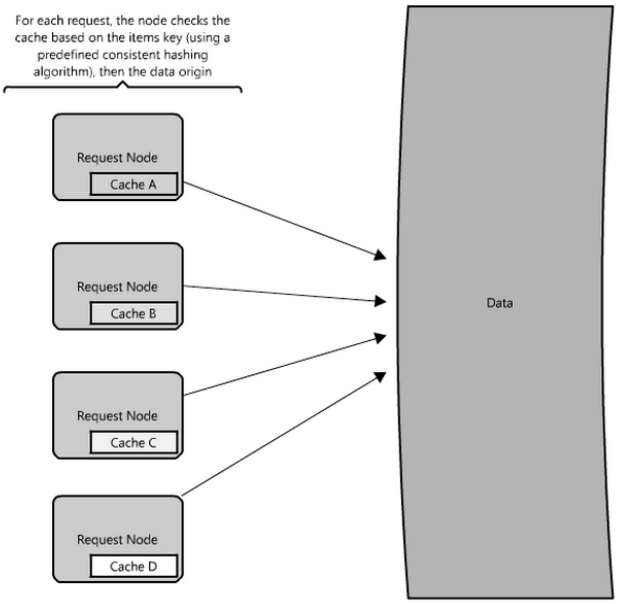

Figure 11: Distributed Cache[5]

One advantage of this method is that the distributed cache increases with ever node that is added to the pool.

7 Queues

Queues are a way to effectively manage writes. When a system is simple, with minimal precessing loads and a small database, writing can be pretty fast. However, when the system gets complex, writes may take a long time. The data may have to be written in several places or indexes which under a high load, can hurt performance.

In the graphic below is shown how it works without queues.

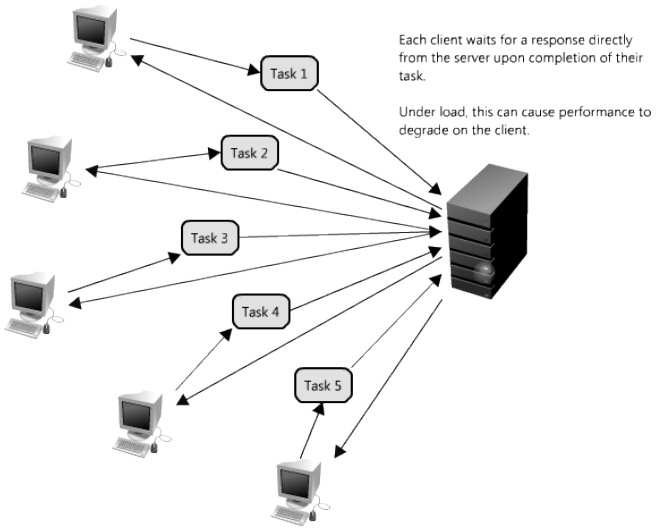

Figure 12: System without Queues[5]

The clients have to wait for the server to answer until they can continue doing something else. This is a synchronous behavior that tremendously degrades the clients performance. The solution to this problem are queues. They abstract the client's request from the actual work performed.

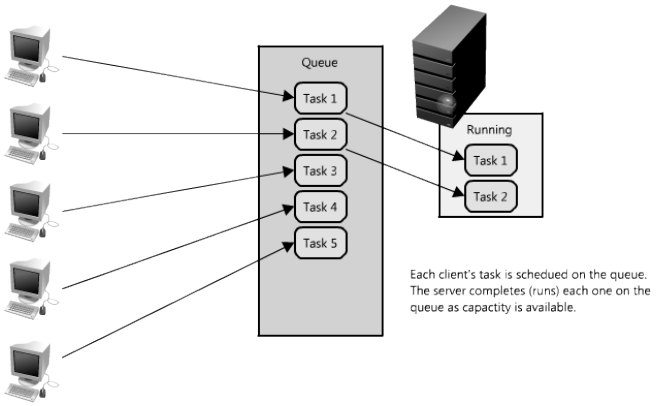

Figure 13: System with Queues[5]

As you can see, the requests are simply put in order as you would expect it form a queue. Then they are run by the server. Queues enable the client to work in an asynchronous manner, because they don't wait for

the answer. They get a message acknowledging the task was received and then they can check periodically if the task has completed. Queues also give protection from server outages and failures. It is quite easy to create a robust queue that is able to retry request.

8 Scheduling Algorithms

8.1 Round-Robin

This is one of the simplest scheduling algorithms. The load distribution happens on every server equally. If this method is used all the servers should have the same capacity. If this is not the case it can mean that a less powerful server receives the next request even though it isn't able to manage it. This could case a overload on the weaker servers in the system.

8.2 Weighed Round-Round

This scheduling algorithms is based on Robin Round. Additionally it has a weight distribution on each server. The administer gives the servers a certain weight and including this, the request are spread. This method takes in account that the servers may have different performance.

8.3 Last Connection

The last two algorithms didn't include the length of the connections. If a weak server only gets a certain amount of weight but those requests stay longer, it still can have an overload. To avoid this problem this method gives the requests to the server with the least connections. *"The server in the cluster with the least number of active connections automatically receives the next request. Basically, the same principle applies here as for the simple round robin: The servers related to a Virtual Service should ideally have the similar resource capacities."* [1] However, in low traffic rates, it is now balanced out because the first server will be preferred. This happens due to the equality of the servers. The first sever will always be selected as long as he has continually active traffic.

8.4 Least Connected Slow- Start Time

This algorithm provides a "ramp-up time" for new servers in the system. When a new sever is added, a time can be configured in which it slowly gets increasingly more requests. This way the server doesn't get overloaded by a flood of connections upon startup. The downside of this method is that it takes a while until the full potential of a new server can be used. This can be a problem if you need to add power quickly.

8.5 Weighed Least Connection

This method is a combination of Last Connection and Weighted Robin Round algorithms. The Number of connection and the weight which the administer gave the servers are taking in account. The next request goes to the server with the smallest ratio between weight and connections. *"The number of active connections combined with the various weights defined by the administrator generally provides a very balanced utilization of the servers, as it employs the advantages of both worlds."*[1]

8.6 Agent Based Adaptive Balancing

Other then the methods above this contains an adaptive logic. It checks at regular intervals the servers for their traffic load. The Load Balancer periodically gets a file from the server in which their work load is stated. This works with HTTP GET function.*"Each server machine should provide a file that contains a numeric value in the range between 0 and 99 representing the actual load on this server (0 = idle, 99 = overload,101 = failed,102 = administratively disabled)."*[1] With this information including the weight each server has the Load Balancer can handle the traffic between the severs very sufficient. Although during a period of very low traffic the Load Balancer switches to Weighted Robin Round. This is because otherwise it would result in uncontrolled distribution.

8.7 Multiple Queue Scheduling

This method separates the queue into two different queues. This way the request is divided and can be worked on at the same time and later being put back together. For example a queue is cut into foreground and background task. Each of these queue can have their own scheduling algorithms. So the first one can use the Robin Round

and second one First Come First Server. However, the two queues also have to be scheduled and put back together.

9 Problems

9.1 Mega Proxy Session

The mega proxy session which is also known as mega proxy problem describes the issue of not being able to determine the correct source IP of an user. This can be caused when the client is located behind a proxy server which determines the client connection, changes the source IP to it's own address and opens a new connection to the actual destination IP.
Most ISPs and enterprises deploy proxy servers in their network to protect their client's identities but this can raise a session-persistence and therefore a load balancing problem on the balancer's side. To avoid this problem

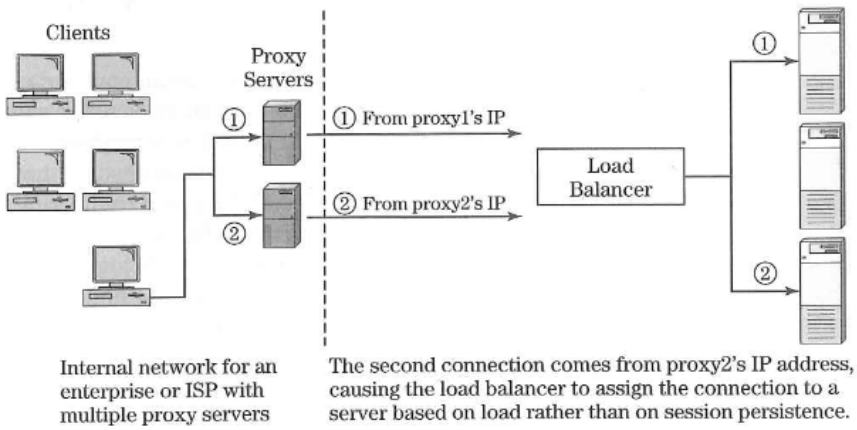

Figure 14: Session persistence problem with megaproxy [3]

the balancer can not longer rely on the source IP address to identify the user and has to adapt it's balancing concept.
The virtual sources of the proxies can be grouped to treat them as one. With this method the balancer is still able to maintain session persistence by directing all requests to a single real server. If this method is reasonable has to be decided considering the traffic scale in your network. On one hand it solves the problem of session persistence but on the other hand it can mess up your load distribution concept because all requests from the proxy network get forwarded to the same server.

List of Figures

Glossary

ADC Application Delivery Controller. 8

API Application Programming Interface. 10

SOA Service Oriented Application. 10

WWW World Wide Web. 8

References

[1] davidquaid. Load balancing scheduling methods explained. Online: http://www.loadbalancerblog.com/blog/2013/06/load-balancing-scheduling-methods-explained, Zuletzt aktualisiert: 06.06.2013, Zuletzt aufgerufen: 03.12.2014. http://www.loadbalancerblog.com.

[2] Vic Hargrave. Tcp/ip network programming design patterns in c++. Online: http://vichargrave.com/network-programming-design-patterns-in-c/, Zuletzt aktualisiert: 8.2.2013, Zuletzt aufgerufen: 04.12.2014. Vic Hargrave.

[3] Chandra Kopparapu. *Load Balancing Servers, Firewalls and Caches*. John Wiley and Sons, Inc., New York, 2001.

[4] liquidweb.com. Understanding load balancing. Online: http://www.liquidweb.com/kb/understanding-load-balancing/, Zuletzt aktualisiert: unknown, Zuletzt aufgerufen: 30.11.2014. http://www.liquidweb.com/.

[5] Kate Matsudaira. Scalable web architecture and distributed systems. Online: http://www.aosabook.org/en/distsys.html, Zuletzt aktualisiert: unknown, Zuletzt aufgerufen: 04.12.2014. http://www.aosabook.org/.

[6] Rui Nataário. Load balancing iii. Online: http://networksandservers.blogspot.co.at/2011/03/balancing-iii.html, Last update: 6.11.2014, Last call: 3.12.2014. Rui Nataário.