Load balancing in Web applications

The content of this article aims to provide some methods for load balancing on your Web application server (cluster) groups. Cluster is a group of servers running concurrently a Web application, the process of linking groups

Vivek Viswanathan

The content of this article aims to provide some methods for load balancing on your Web application server (cluster) groups. Cluster is a group of servers running simultaneously a Web application, the process of implementing this group link makes the server act as a single server when viewed from an external perspective. To balance server load, the system needs to distribute requests to different nodes within the server cluster, with the aim of optimizing system performance. This will give your network more performance, scalability - avoid falling into a shortage of network resources in a business or Web application.

High availability can be interpreted as redundancy. If a server cannot manage a request, can other servers in that cluster manage it? In a high-availability system, if a Web Server fails, another server takes over immediately to process the request.

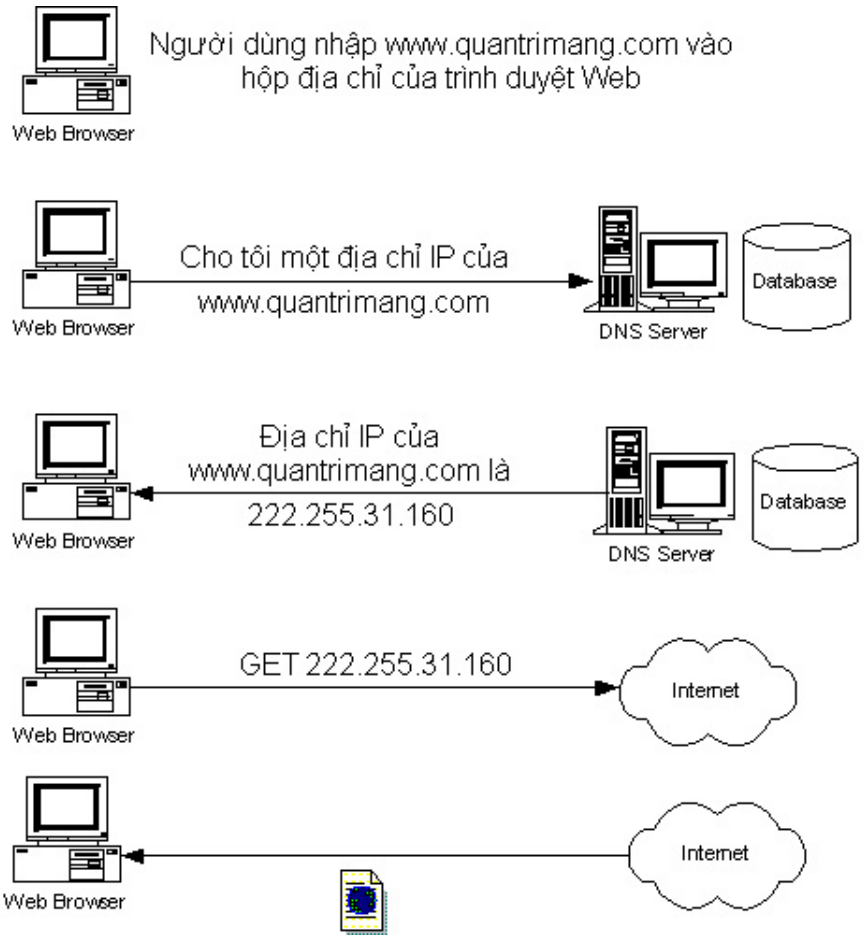
Scalability (Scalability) is the ability of an application to support an increasing number of people. If it takes 10ms for an application to respond to a request, how long will it take for it to respond to 10,000 requests at once? Infinite scalability will allow it to respond to these requests in about 10ms. Scalability is a measure of a range of factors such as the number of concurrent users that a cluster can support and the time it takes to process a request.

There are two main methods for load balancing:

1. DNS round rotation
2. Use hardware load balancers

DNS round rotation

Most of you probably already know that the Domain Name Server (DNS) database maps host names to IP addresses.



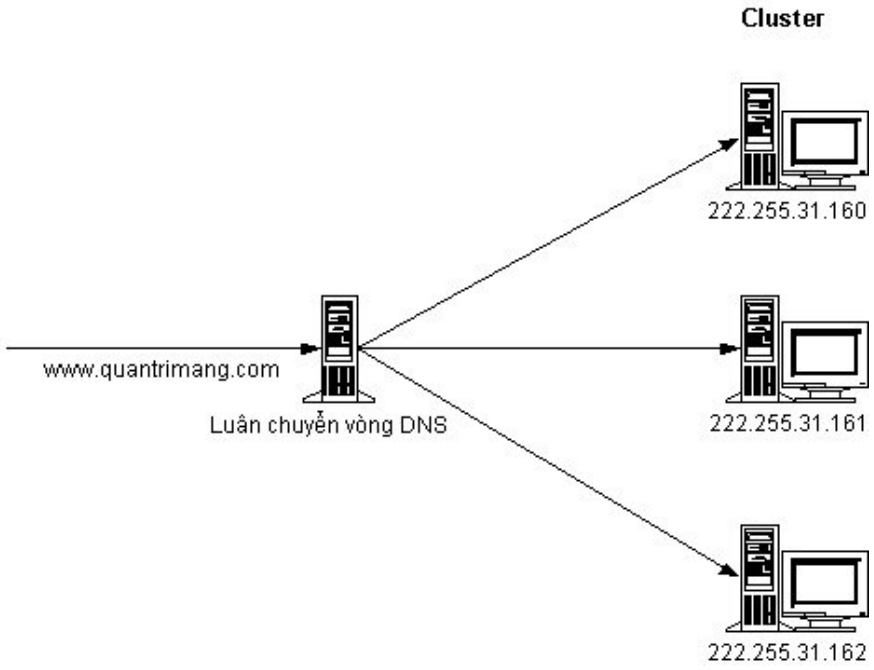
When you enter a URL into a browser (such as *www.quantrimang.com*), the browser sends a request to the DNS asking it to return the IP address of the site. This is called DNS lookup. After the Web browser has obtained the IP address for the site, it will contact the site with the IP address, and display the page you just requested. DNS servers often have an IP address mapped to a site name. In our own example, the site *www.quantrimang.com* maps into an IP address of *222.255.31.160* .

To load balance with DNS, the DNS server must maintain a number of different IP addresses for the same site name. Many IP addresses represent multiple machines in a cluster, all of which map to a logical site name. In our example, www.quantrimang.com can be configured on three servers in a cluster with the following IP addresses:

222.255.31.160
222.255.31.161
222.255.31.162

In this case, the DNS server is mapped as follows:

www.quantrimang.com 222.255.31.160
www.quantrimang.com 222.255.31.161
www.quantrimang.com 222.255.31.162



When the first request reaches the DNS server, it returns the IP address 222.255.31.160, the first machine. When there is a second request, it returns the second IP address: 222.255.31.161. Again, with the fourth request, the first IP address is repeated.

By using DNS loop rotation as above, all requests for a site are distributed equally to all machines in the cluster. Therefore, with this load balancing method, all nodes in the cluster are used.

Advantages of DNS round robin method

The main advantages of this method are that it is cheap and easy:

• **Not expensive and easy to set up** : System administrators only need to make some changes in the DNS server to support round robin, and many DNS servers have this support. It does not require changes to the Web application code; In fact, Web applications are unaware of the load balancing mechanism that it is performing.

• **Simple** : This method does not require network experts to set up or troubleshoot the system in case something happens.

Disadvantages of this method

There are two main drawbacks to this software-based approach: it does not provide real-time relationship support between servers and does not support high availability.

- **Does not support real-time relationships between servers** . The real-time relationship between servers is the system's ability to manage the requirements of this user, server or any server, depending on the session information maintained on the server or at the base level, database level.

Without the ability to support the relationship between servers, the DNS round-robin method relies on one of three methods that have been introduced to maintain session control or user identity for requests. is coming over

HTTP.

1. Cookies
2. Hidden schools
3. Rewrite the URL

When a user makes a first request, the Web server returns a single text tag to distinguish that user. Subsequent requests have this tag to use cookies, rewrite URLs, or hidden fields, allowing the server to appear to maintain a session between the client and the server. When a user establishes a session with a server, all incoming requests usually go to the same server.

The problem here is that the browser stores the IP address of that server. When the Cache expires, the browser will make another request to the DNS server to obtain the IP address associated with the domain name. If the DNS server returns another IP address, another server in the cluster, the session information will be lost.

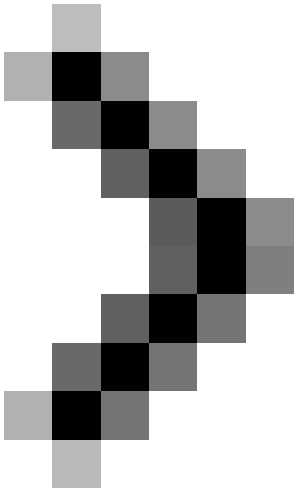
- **Does not support high availability** . Consider a cluster with n nodes. If a node has a problem, every nth request to the DNS server will direct you to this broken node. A smart router can solve this problem by checking the nodes at certain intervals, detecting broken nodes and removing them from the list, so there is no requirement. Any more sent to them. However, the problem here still exists if the node is still there but the Web application running on the node has been corrupted.

Changing the cluster will take a long time to spread to the rest of the Internet. One reason is that in many large organizations - ISPs, companies, or agents - save their DNS requests to reduce network traffic and request time. When users within such organizations make a request, the system will check the list of Cache DNS names that have been mapped to the IP address. If the system detects an entry, it returns the IP address to the user. If it does not detect any entries in the internal Cache, the ISP sends this DNS request to the DNS server and saves the response.

When a saved entry expires, the ISP will upgrade its internal database by contacting other DNS servers. When your list of servers changes, it may take a short time for the entries saved on other organizations' networks to expire and search for the updated list of servers. During this cycle, the client can still perform a "hit" action on the broken server node, if that client's ISP still has an entry pointing to it. In such a case, some users of that ISP cannot access your site from the initial access, even if your cluster has redundant servers that are still active.

A bigger problem arises when removing one node from the addition. When you reduce a node, the user may be executing a 'hit' of a non-existent server. When you add a node, the server is not used until its IP address reaches all DNS servers.

Although this method can balance a number of users on each server, it does not completely balance server load. Some users may require higher load levels throughout their session than other users on another server, and this method cannot guarantee that unfairness.



Part II: Hardware-based load balancing

You finished reading the article "**Load balancing in Web applications**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.