LoadBalancing using MS-Redirect Mechanism

G. Naveen Kumar¹, T. Ram Kumar² and B. Phijik³

Vignan’s Institute of Technology and Aeronautical Engineering
Deshmukhi(v), Pochampalli(M), Nalgonda, Hyderabad

Abstract: In the present time the use of internet is very popular and a primary means of data/information source and data/information sharing. The heart of internet is the web servers. Web servers accept the requests generated from the clients and then generate responses to those requests. With the increasing usage of internet, the number of requests per server is also increasing. After a certain limit of requests, the servers will reach to a point where they can no longer service the requests as the servers have already reached maximum load. Then the clients will not get any response from the servers. So we have to establish multiple servers to accommodate the means of handling many requests simultaneously. This paper discusses a technique of very efficient load balancing among those servers by implementing the simple HTTP Redirect mechanism with the Master and Slave configurations of the web server. This paper also gives the technique for recovery wherein the master server seizes/crashes to work. This technique is adapted from a very widely used load balancing technique by the name of Round Robin DNS Load Balancing.

Keywords: Load Balancing, DNS, Round Robin DNS

1. INTRODUCTION

In the present time, internet has become the source of information exchange. Web servers which are used to host the web applications (or any other application that is communicating through internet) are constantly being improved in terms of features and the complexity of the tasks it can perform, thereby consuming the system resources even at a high level. Web servers respond to the clients by accepting the requests from the clients and then generate the responses as per the application. Consider a simple website sharing simple information in terms of HTML with the combination of images. If the web page contains 10 images along with the text, then a browser which is requesting this information will be sending a total of 11 requests out of which 10 requests are to get the images as “image/jpg” format and the remaining one request as “text/html” format. Therefore even if the web server is hosting small data/information, based on the requests made by clients, the server has to respond to all requests, thereby increasing the load on the server, otherwise the server will be recognized as downtime server (Server which cannot respond to any client request, in shutdown or crashed state). In another scenario, web hosting companies will be utilizing the single server to host many applications simultaneously. Every application will be having its own requirement of bandwidth, CPU resource and memory requirements. The world’s top search engine “Google” will be having around 3 billion hits in a single day, and in www.alexa.com we can see that around 40% of the users worldwide use “Google’s Services” in a single day. To eliminate these load conditions on a single server we are utilizing multiple servers hosting the same application to share the load among them. To manage the load properly we will need a load balancer. Round Robin DNS technique is a very popular load balancing technique among them. “Google” even uses this technique to manage its servers. So first we discuss the Round Robin DNS and identify its disadvantages and then we will see the method of removing these disadvantages using MS-Redirect Mechanism.

2. ROUND ROBIN DNS LOAD BALANCING

ROUND ROBIN DNS Load Balancing Algorithm: RRDNs server works in a very simple manner of iterating through the available servers after reaching to the last of the list it again continues from the first server. For example if we are having 5 servers connected together sharing the same application by the name of “TEST”, with our domain name www.demoexample.com. Our 5 server are having the IP Addresses of “192.168.1.1”, “192.168.1.2”, “192.168.1.3”, “192.168.1.4”, and “192.168.1.5”. Then RRDNs Works in the following fashion as shown in the table below.

<table>
<thead>
<tr>
<th>Req No</th>
<th>Request made by client</th>
<th>DNS Record</th>
<th>Server IP being serviced</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><a href="http://www.demoexample.com">www.demoexample.com</a></td>
<td>192.168.1.1</td>
<td>192.168.1.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><a href="http://www.demoexample.com">www.demoexample.com</a></td>
<td>192.168.1.2</td>
<td>192.168.1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td><a href="http://www.demoexample.com">www.demoexample.com</a></td>
<td>192.168.1.3</td>
<td>192.168.1.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>192.168.1.2</td>
<td></td>
</tr>
</tbody>
</table>

Upon receiving the 6th request again 192.168.1.1 will be serviced.
So we can see here that the requests here are simply being iterated through the DNS record by forwarding the request to the top most IP Address in the DNS record. Here we are having several problems.

2.1 Problems of RRDNS:

2.1.1: First, if the Server with IP address 192.168.1.1 is already full with maximum number of requests that it can be accept, then the server will not be able to service the new request made by client. Therefore the server cannot service the request of a client. But RRDNS will continuously provide different future requests to the same server irrespective of load at the server.

2.1.2: Second, if the DNS server is failing then to recover the only option is secondary DNS has to become active and it has to check the load balancing. Therefore the secondary DNS server must also be updated with the information and cache record of web servers. Otherwise, the secondary DNS Server will start from the first server again.

2.1.3: This scenario is very problematic if you have multiple applications running in the same server or the redundancy of the web application is not in all the servers, but only in few servers. In this scenario, RRDNS will forward the request to a web server where in the web application might not even present. So the request will be forwarded to a server which is not capable of handling that request.

2.1.4: Updating of cache record is very resource consuming process wherein it has to check and update the entire DNS record. This is very problematic if the web server farm is containing many web servers (Ranging from thousands of web servers).

So in this topic, we are going to use the HTTP Redirecting mechanism (in case of HTTP request or else a similar manner of redirection mechanism for different application type) to redirect the request of the clients to the respective web server by effectively reducing the load, by considering the present load situation of the web server as well as the capability of the web server to handle the request. This technique not only works for web servers, but is also applicable for other online applications.

3. MASTER AND SLAVE REDIRECT MECHANISM:

This system comprises of entirely 5 modules
1. Master Server
2. Slave Web Servers
3. Client Systems
4. Backup Master Server
5. DNS

Figure 3.1: Architecture of MS-Redirect Mechanism

3.1 MASTER SERVER WORKING PRINCIPLE

The master server works as an administration entity which does the entire work. The master server is the main machine which is having public IP address as well as a domain name associated with it. It will accept the requests from the client systems and then forwards to the appropriate slave web server for completing the request. The MASTER server after receiving the request from a client machine, updates the status its slave servers by sending a “STATREQ” message to all the slave servers, corresponding to this request the slave servers will update the MASTER Server with the Resource load at the current time such as CPU Load, Memory Load, Number of current connections by sending a message “STATSENT” along with the data packet containing the resource load data. The data packet will be as below

Figure 3.2: Packet format of STATREQ

This packet is sent as a TCP/IP Packet from Slave server to the master server, therefore the security and authentication concepts that can be applied on TCP/IP Packets can be applied here.

The below diagram represents the message passing between the master server and the slave server, along with the collaboration diagram where in we can see the sequence numbers of the messages that are being exchanged among the servers and the clients.

Figure 3.3: Message Passing between Master and Slave servers

The individual fields of the STATREQ Packet are explained below:

3.1.1 Application Identifier: Unsigned 32 bit field having a unique application identification numbers, which are described by the user at the time of deployment of a web application or any other type of network based application. This field is given the first priority, because if the client is requesting for an application which is not present in the slave server, then even if the load is less,
the slave server is useless and the request made by client cannot be forwarded to that slave server.

3.1.2 CPU Load: Unsigned 8 bit short integer data indicating the load of the CPU in terms of 1-100. The higher the CPU load, the less will be the priority of that server to service the request. Other than 1 to 100 for any other value it is treated as don’t care value (Master server will simply ignore that data packet)

3.1.3 Memory Load: Unsigned 8 bit short integer data indicating the load of the Main Memory in terms of 1-100. The higher the usage of Main Memory, the less will be priority of that server to service the request. Other than 1 to 100 for any other value it is treated as don’t care value (Master server will simply ignore that data packet)

3.1.4 Number of Connections (Unsigned 8 bit short integer data): As the slave servers are configured to use service multiple client connections, naturally we have to see the number of connections that are being serviced at the moment, because “Microsoft Windows Operating System” is having the limitations on the number of concurrent bidirectional TCP connections.

3.1.5 FLAG(1 BIT): is used to send the MACROs to indicate the status of the slave server

a. STATSENT: if the received flag is STATSENT, then the Master server will take the details of the rest of the data packet to analyze whether to consider the server or not for servicing the client request.
b. STATREJ: if the received flag is STATREJ, then the slave server actively reject any further connections.

3.1.6 IP Address: Indicates to the master server that which slave server is sending the details of current load.

By considering the example below, we can identify the working of Master server as per the received request from client:

Request Received: www.demoexample.com/test

Upon receiving the above request master server sends the STATREQ message to all the slave servers. Then the slave servers will update the Master server as follows:

- 192.168.1.1 | STATSENT | “10,15,18 ” | [80][40][15]
- 192.168.1.2 | STATREJ | ---------------- | [00][00][00]
- 192.168.1.3 | STATSENT | “10,15,20 ” | [15][50][10]

So in the three packets received from the three slave servers, we can identify that the 2nd server is actually refusing the connection because, the FLAG is STATREJ. In the 1st and 3rd server the priority is given for the application identifier. In this example the TEST application is having an identifier value of 10. So the application is present in both 1st and 3rd servers. Therefore next field will be checked. In CPU load the preference is given for 3rd server as the CPU load is only 15 for 3rd server as compared to 80 in 1st server. (Memory load and Number of concurrent connections are not taken into consideration in this scenario, will be explained later in the document). Therefore slave server 3 will get a chance to service the request of the client. Master server sends a redirect message to client with HTTP Status 302 and Location: URI where URI is the address of the slave server which the master server has selected. After sending the redirect message to client, the Master server also sends the IP Address of the client machine to the slave server which has been just selected, to indicate to the server that it can expect the client to be connected soon. This is done because once the client knows the redirected URI, client can directly connect to the slave server without the interpretation of the master server. To prevent this, the client has to first get the authentication from the master server and the slave server will be getting a notification from the master server so that it can expect the client to be connected soon or reject the connection.

3.2 HTTP Redirect Mechanism: As per RFC2616, HTTP Status code 302 is for temporarily redirecting the client to use a new URL instead of going for the previous URL. But in future the client will be using the original URL only to retrieve the information.

Example:

client request:
GET /demo/myfile.html HTTP/1.1
HOST: www.demoexample.com

Server Response:
HTTP/1.1 302 Found
Location: http://www.demoexample.com/newserver

We can use this redirecting mechanism in our own applications also which doesn’t use http.

3.3 SLAVE SERVER WORKING PRINCIPLE

The slave server will be working as per the instructions given by the master server. Once the HTTP 302 redirection is achieved by the client system, then the client will redirect itself to the slave server to get the application that it needs. Slave server will receive a message consisting of two fields, “REQACCP”,”IPADDRESS”. REQACCP stands for Request Accept. Upon receiving this message, the slave server waits for the client to connect and utilize its resources. If the client is not connecting within a specified time limit, then the slave server will remove it information from its record. This is used to prevent the users from connecting to the slave server directly without visiting the master server. Once the connection has been made by the client to the slave server, the client will request the services and the responses will be generated by the slave server. Here the updating of the Master server about its resource status will not put an overhead or load on itself because the entire data packet is of the size 89 bits (32bit for AppID, 8bit for CPU Load, 8 bit for Memory load, 8 Bit for number of connections and 1 bit
for flag, 32 bit for IP address) and the updating is only done when the MASTER Server has an incoming request from a client. Until a request comes from client, MASTER server will not request for Update from slave servers

3.4 MS-REDIRECT MECHANISM TIMELINE:
The identification process of the slave server by the master server is done in the following manner.

3.4.1 Decide a threshold value to parameters such as CPU Load, Memory Load and Number of connections( Can be decided by the administrator of the entire network)
3.4.2 Upon receiving the client request at master server sent “STATREQ” to all the slave servers, gather the responses.
3.4.3 Take the first server from the responses from slave servers; put it in a variable called “CURRENTSERVER”. Check its application identifier and decide whether it is able to service the application or not. If not found then ignore the packet and move onto the next slave server
3.4.4 Suppose the application has been identified. Then put the CPU Load of that server in “CURRENTCPU” Variable. If the value is less than threshold, then use that slave server to service the request. Suppose it is more than the threshold, then find a CPU Load value by iterating through all the packets to find the least one or until a server load which is less than the threshold is found.
3.4.5 After finding the CPU Load compare the Memory load of that slave server with the threshold value. If it is less than the threshold value accept the server, repeat this step for number of connections also. But if the values are more, then look for the next slave server found when comparing the CPU Load values. Look for the next server with next CPU Load, check the difference between the Previous CPU Load and Present CPU Load and likewise for Memory load of the previous server and present server. Then find the optimist slave server to forward the request.
3.4.6 Slave server side identification is removed because; the master server has to forward the request to all the slave servers, there by flooding the network.

4.ADVANTAGES OF MS-REDIRECT MECHANISM:
- Master server has to gather data of all the slaves, which will send the flag of “STATSENT”. It can freely ignore the slave servers which are sending “STATREJ”, because the slave servers are actively rejecting further requests from the clients. So we are providing a fair opportunity of slave servers to either provide the service or not as per the currently load. So the slave servers are having a chance of either accepting or rejecting.
- Suppose there are 1000 servers with “STATSENT” message, the maximum data size that can get by collecting all the data from 1000 servers is 1000*89 bits= 89000(89K data). Searching for a string in 89K data takes very less amount of time (in the span of Milliseconds only). Therefore to find whether the application is supported by server or not, will take very little time and the redirection will be done effectively within a small time frame.

5.RECOVERY MECHANISM:
Till now we have discussed the Load Balancing of web servers using Master Slave redirect mechanism. But suppose the Master server is crashed, then all the slave servers cannot be serviced because, incoming connections are only accepted when the Message from Master server is coming to the slave server containing the data of IP Address of the client and Application requested. The following technique describes the recovery mode when the master server crashes.

5.1 BEACON MODE: The slave servers will be continuously using a beacon (A small data packet to see the connection between the master server and slave server as in WIFI) with some interval of time (in terms of seconds or milliseconds). So the entire system will be unavailable for a maximum time of seconds. This can be further eliminated if the beacon interval is reduced to micro to milliseconds. Once the beacon is sent by slave server to the master server, it replies to the beacon in the form of acknowledgement. If the acknowledgement is not coming from master server, then the slave server will retry for a predefined number of retries. Once the reply or acknowledgement doesn’t come within the specified number of retries, then the slave will send a multicast message to all the connected slave servers to check for beacon. Upon receiving this message by other slave servers, they will check for the beacon acknowledgement and if an acknowledgement is received by other servers, they will notify to the original slave that retry the beacon. If other slave servers also not receiving the acknowledgement, then they decide that the master server is crashed. A maximum time interval of 30 seconds is taken to find that the master server is not working. Once
the slave servers decide that the master server is not working, they will try to create a new master server. The slaves will invoke the backup master server system which will now be the Master server by acquiring the IP Address of the original Master server. If the Original Master server is recovered from crash, it will not replace the current system, now the original master server will become the backup master server.

6. CONCLUSION:

Round Robin DNS server only redirects the requests coming from clients to the respective web servers by simply iterating through the DNS Record, without check for the server availability or the current load parameters at the server. Using this technology the system can perform efficient load balancing with simple HTTP Redirect mechanism. This technique is also having a recovery mechanism if the master server fails or crashes.

REFERENCES:

[3]. W3C consortium’s HTTP Status codes: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html
[7]. Mei Lu Chin- “Efficient load balancing for bursty demand in web based application services via domain name services”

AUTHORS

G. Naveen Kumar received B.E from DCET, OSMANIA University, Hyderabad and received M.Tech from VNRVJET, JNTU Hyderabad. Presently working as Assistant Professor in the Department of C.S.E in Vignan’s Institute of technology and Aeronautical Engineering. His areas of interest are Artificial Intelligence, Computer Networks and Web services, Computer Organization.

Mr. T. Ram kumar received M.Tech from Bharath University Chennai, pursuing Ph.D from Kurnool university. Presently working as Associate professor in the Department of C.S.E in Vignan’s Institute of technology and aeronautical engineering. His areas of interest are Operating Systems and Computer Networks.

Mr. B. Phijik received M.Tech from JNTU Hyderabad, presently working as Associate professor in the department of C.S.E in Vignan’s Institute of Technology and aeronautical engineering, Hyderabad. His areas of interest are Software Engineering, Software Process and project management and computer networks.