Back-End Forwarding Scheme in Server Load Balancing using Client Virtualization

Shreyansh Kumar
School of Computing Science and Engineering
VIT University
Chennai Campus

Parvathi.R, Ph.D
Associate Professor- School of Computing Science and Engineering
VIT University
Chennai Campus

ABSTRACT
Load balancing can be done at two levels, the HTTP level and the network level. It refers to the distribution of the HTTP request made by the user across the web server in a server cluster and is achieved by various algorithms such as Random Selection, Round Robin, weighted Round Robin etc. At the network level load balancers are used for the purpose of balancing the load. The approach such as DNS forwarding and spraying the load across the servers is very useful and effective at the network level [11][12]. The load balancers receives the request and proxies the destination ip for forwarding the request to the less utilized servers in order to gain better performance. This paper proposes a server load balancing scheme mainly for the webservers by making the limited number of clients to work as a virtual server which are called virtual clients in the webserver network. This approach is done at the HTTP level by forwarding the HTTP request to the nearest and less loaded virtual client. The HTTP request is only forwarded to get the static content in order to save the current bandwidth utilization of the web server and process more dynamic request on the server side. The daemon routine controls the whole process of web server load balancing but it only processes the HTTP requests. The daemon routine only resides on the web server backend. The client virtualization provides a new anti-overloading technique to reduce the access time of the server and to utilize the server resources without any additional cost. The Client Virtualization also provides a better way to utilize the system bandwidth in web server network.

General Terms
Web-Server, HTTP, URL

Keywords
Virtual Clients, dynamic load balancing, daemon routine(process)

1. INTRODUCTION
Since the deployment of the World Wide Web (WWW) servers the growth of internet user has vastly increased. In the era of rapid developing internet technologies always a huge number of internet users are expected across the globe. To satisfy the needs of the internet users there is a need to serve them according to their requests. The term internet user is called as client. The WWW webserver should provide high quality service [5]. When the client request for a particular resource then the requests has to be serviced so, for servicing these clients we need a Web server. Since, the amount of internet users is quite large a webserver has to process large number of requests thus, at times resulting in overloading the webserver. To process the large number of requests from the clients the web server easily gets overload thus, it's better to migrate [10] (distribute) the load on other available servers too. The server load also has to be distributed so as not to overload the other available servers in the web-server network. The network congestion in system can reason the overloading which prompts break-down of servers. The best possible appropriation of load on the server is carried out by the load balancer. The load balancer is answerable for the equally assigning the requests to the server pool in order to disallow the over-loading of the single server in the web-server system.

The load balancing techniques could be exceptionally adequate for the server hubs to process the client's demands. The load balancing strategies helps in minimizing the reaction time and to legitimately use the throughput of the server network. A large number of algorithms are being utilized by the load balancers to verify which server to send the request to. Random choice, round robin, saturation load balancing, load based, least connection and failover load balancing are the algorithms which are being very frequently used by the load balancers to distribute the load. To enhance the load adjust around the servers various parameters, for example server's recent load, recent reaction time and how much traffic it has been recently assigned are taken into account.

2. RELATED WORK
The correspondence between the web client and a web server is two layered architecture. The architecture is characterized by the client that shows the information content and web server that transfer information to the client. A web server is a procedure which runs on a framework which accepts the HTTP requests from the clients over the web server system and is equipped for transforming the solicitation with the asked for a particular web document. The remote clients can connect to the web server through their web browser by clicking the valid web address in the URL field and the request is forwarded to the DNS server then the DNS server forwards the request to the actual mapped server [3][4][7].The web server gets truly extensive number of HTTP requests from the clients for the requested web content, after appropriating the client's demands the web server delivers the response for the web content which comprises of both the static and the dynamic part. Provided that a web server appropriates an immense demand it will make diverse threads for every client and the servers request and response handling. This thread creation for every client may require some serious time from the webserver to serve the number of HTTP requests from the client.

The general network level load balancing architecture with DNS forwarding is shown in the Figure1. This DNS

forwarding is done to find the right choice of the server in the server pool. The DNS server and the load balancer maps the servers ip address to the alternate ip address of the servers in the pool.

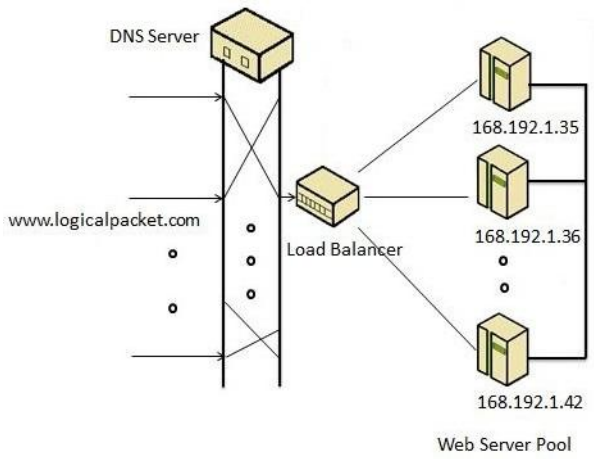


Figure1. DNS Forwarding of HTTP request

In the network, the web-servers are the end of the flow. For better scalability the clusters of servers are used [4]. The load balancing schemes such as Random Algorithm will distribute the resources and load due to random selection. It can lead to under-utilization of the other servers. The Round Robin algorithm distributes load equally to each server, regardless of the current connections and the response time. Round Robin is best suited for the servers with equal processing capabilities [2]. If the servers are not homogeneous in their processing capabilities then the Round Robin algorithm will not give the optimum result. Weighted Round Robin algorithm overcomes the deficiencies in the plain Round Robin algorithm but increases the complexity of the algorithm [2].

In the network only one server will serve a request. The request and the response process is shown in the Fig1. The request for several clients is processed by a server [9] in Figure2.

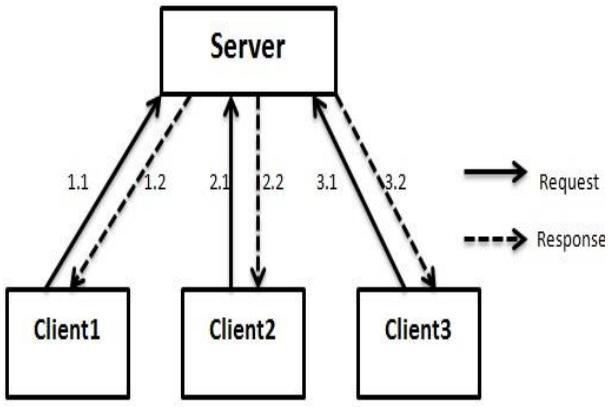
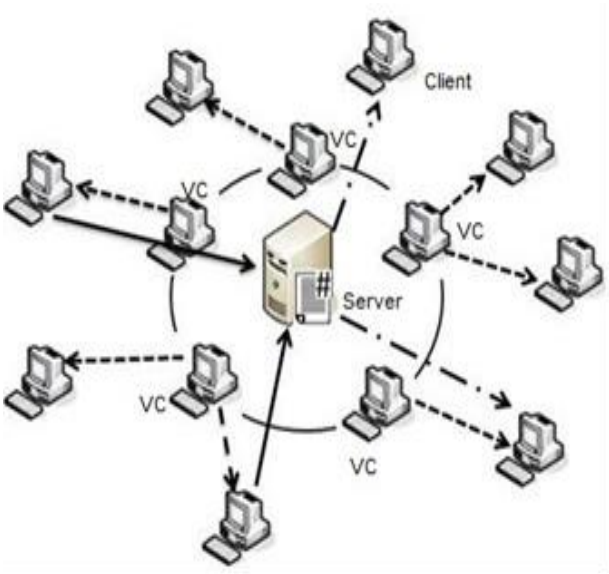


Figure2. Basic Client Server Interaction

3. PROPOSED SYSTEM

The whole system setup follows the peer to peer system architecture. The client and the virtual client cannot directly connect for the request processing. The web server creates an abstraction layer between the client and the virtual client. The system tries to distribute the load dynamically [9] according to the number of requests from the clients. The clients in the web server network form a cluster through which the server can gather all the information in the virtually clustered state itself. The web content consists of the dynamic and the static part. By introducing a Daemon Process (software) on the server side the server can forward the request to the Virtual Client which in turn will reply back with the static content to the client which has requested for the page in the meanwhile the actual server will respond to the client with the dynamic content of the requested web content. This can reduce the extra bandwidth required by the static content so, this saved bandwidth can be used to process another requests. This system therefore, utilizes the bandwidth accordingly which the virtual client can be loaded with the requests in the web server network. In the Figure 3 the system on the circular ring are the virtual clients which can send the static document to the clients in the web server network. The clients and virtual server utilizes P2P [6] communication to utilize available resources [1] but in a restricted manner for this architecture in figure3.



VC – Virtual Clients

Request

Response from Virtual Clients

Response from the server

- Server load parameter table.

Figure3. The Proposed Architecture

The web server forms a small cluster with clients acting as virtual server i.e. virtual clients. The web server node maintains a routing table which helps in redirecting the requests to the virtual clients from the actual server.

Role of client:

Request's the server for particular web content which has been hosted on the webserver.

Role of Virtual Clients:

These clients have the static data of the web content. These, client will go about as a virtual server to the clients which has asked for the web content from the actual server. The virtual customers are straightforwardly connected to the actual server so they are continued the web server network i.e the network

portrays which ever client falls under the particular network will go about as a virtual client. The server chooses which client can turn into a virtual client consistent with different parameters of the client's network framework.

Role of server:

The essential part of a server is to process the HTTP requests of the client consistent with the web content requested. It additionally keeps up a load parameter table which holds the data about the virtual clients joined with the server in the system. It reviews its load table each few clock ticks. It additionally considers the load parameters for every virtual client.

Role of a Daemon routine:

The fundamental work of the daemon routine is to redirect the solicitations for the dynamic content to the virtual customer. It will tune in the determination procedure of the virtual client. It will sense weather the virtual client is loaded or not.

4. WORKING OF THE SYSTEM

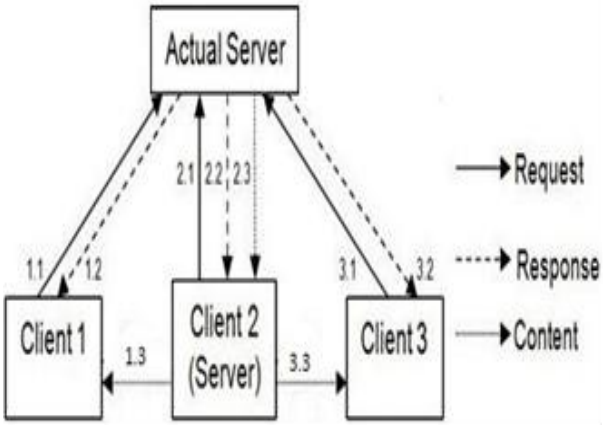


Figure4. Proposed System Architecture

In the Figure4 as shown above the system is layered into two phases called Request and the Response phase. In the above system consider the Client2 is the first one to make a request (2.1) for the web content. The server will not have any entry for the virtual client corresponding to the web content in the routing table (load table). If the content information (URL) of virtual client is not present then the server will send (2.2 and 2.3) the whole content i.e both dynamic and the static content to the Client2. The web-server will add an entry of the Client2 in its routing table. The Client2 will have the static content saved in its browser cache. The Client2 can become virtual client. If Client1 has requested (1.1) for the same web content as of the Client2 then the entry of the Client2 will be present in the server load table. While if the entry is present in the load table then the Daemon Routine will redirect the request to the Client2 then the Client2 will forward (1.3) the static web content to the Client1 and the server will send only the dynamic web content to the Client1. The web-server will keep the few entry of the virtual client corresponding to particular web content according to the load parameters. This is how the web server will form a small cluster with all other virtual clients in the web-server network. If in case of the Client3 the situation can be similarly analyzed as Client 1.

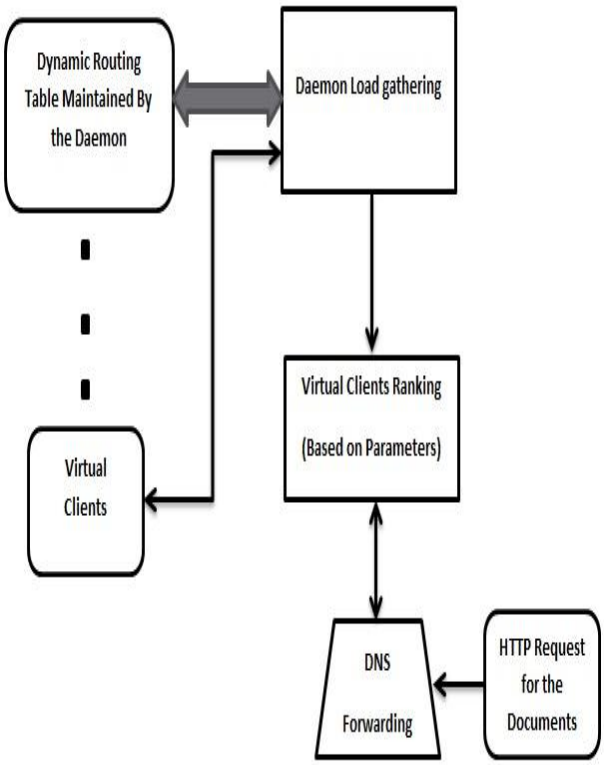


Figure5. Daemon Information Gathering

The above figure5 gives the flow of the daemons load gathering process. The whole process is based on the request made by the client at the HTTP level for the web content. The DNS forwards the daemon routine where the ranking of the virtual client has been done for performance enhancement through the daemon forwarding.

5. ALGORITHM DESIGN

Load balancing problem in the webserver network can lead to delay in fetching the web contents thus, in order to reduce the significant load an algorithm have been proposed which can tackle the distribution of the load so that the main server in the proposed web server network does not get overloaded.

Total Load estimation and balancing:

The load information on the server should reflect the number of request forwarded to each virtual client over a period of 5 seconds, uploading speed of the virtual clients and the average traffic on each client.

$$LI\ n = \sum(re\_fn, up\_sn, av\_tn) \quad (1)$$

Where,

LI - Load Information.

LI n – Load value of node n.

re_fn – Number of request forwarded to each virtual client

over a period of 5 seconds.

up_sn – Uploading speed of each client.

av_tn – Average traffic over a period of 5 seconds on nodes.

Node Classifications:

The node states can be determined from the load values involved. Load re_fn will have the number of requests

forwarded to each virtual client's. Load up_sn will differ client by client based on their uploading speed. Load av_tn will have the average number of request forwarded to the total number of virtual clients for a web document. The av_tn will be taken as a threshold value. The request will be forwarded after comparing the values of re_fn to av_tn.

The nodes are classified into three states named safe state and the loaded state.

IF ((re_fn << av_tn) or (re_fn = 5% av_tn))

Then "Idle" state reached.

Else If (up_sn <= 50% utilized)

Then "Safe State" reached.

Else " overloaded state".

Idle state:

This State is reached when re_fn is very much lesser than the av_tn ie. (re_fn << av_tn). It demonstrates that the effective resources utilization of the system very less.

Safe State:

This state is reached when the value of re_fn is less than av_tn (re_fn < av_tn). This condition is checked over a period of every 5 seconds thus which results in the state transition.

Loaded State:

This state is reached when the value of re_fn is more than av_tn (re_fn >> av_tn). This condition is also being checked over a period of every 5 seconds.

Selection criteria for virtual client:

This process will choose the best virtual client available. This depends on two criteria.

1. If the re_fn value for a virtual client is much less than av_tn (re_fn << av_fn).
2. If the up_sn value of a virtual client is much high in the load table.

6. WORK FLOW

The architectural components in the Figure 6 show how they are connected. The client generates HTTP request and the request is forwarded to the webserver through the DNS server and then the request is forwarded to the Daemon routine [11]. On receiving the HTTP request the daemon routine checks the network table for verifying the client. The table lookup process is carried out at every five seconds. This is to maintain the dynamic table. Upon receiving the HTTP request the daemon routine chooses the best virtual client from the dynamically maintained load table. The virtual client should have the corresponding requested web document. The link of the web document is correspondingly stored in the load table for quick matching and the response. To check for the virtual clients who are alive in the network the load table gets updated in every 5 seconds. The useless virtual clients are removed from the table during the updating process of the load table. The virtual clients will send the static contents to the actual client which has requested for the document through the HTTP request. In the end the response is generated for the static contents to be forwarded to the desired client. The workflow can be directly referred from the Figure6.

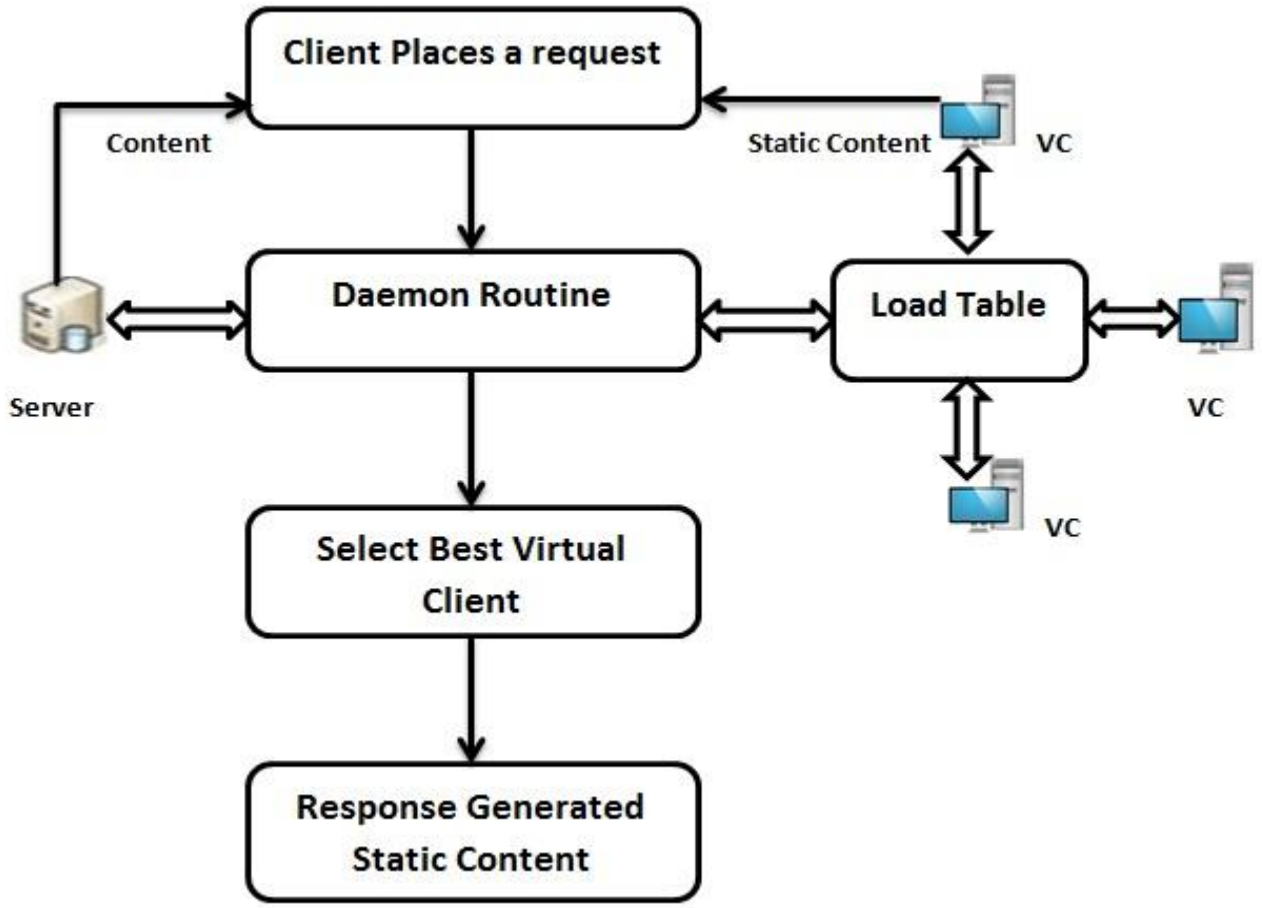


Figure6. Work flow of Client Virtualization architecture

7. BACK-END FORWARDING

The Back-End forwarding scheme [Figure 7] of the web servers can be vulnerable thus an encryption technique can be used with the back-end forwarding done by the daemon process. The dynamic contents take more time if the application consists of videos and multimedia contents. Always the static contents cannot be cashed either. If it is always cached the browser will always face an overhead to find the contents from its cached documents. The security at the back-end layer will lead to share the contents secretly. The software developed for this can be embedded in NIC cards to gain better performances. The NIC very quickly will transfer the requests to the virtual client.

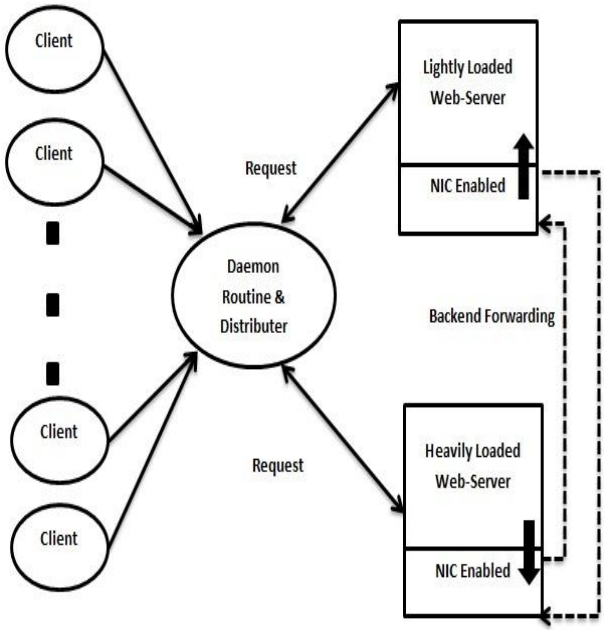


Figure7. Request redirection architecture

8. LOAD LEARNING

The load learning is done by the daemon routine by verifying the log files. There is a provision to set an extra bit for fast

searching for the most used contents. The load table is also known as network or the routing table in this scenario. The routing table consists of the current state scenario. This load learning will help in the quick redirection of the request to the virtual clients.

9. EXPERIMENTAL RESULTS

For the experimental setup a highly customized web browser has been developed in java to support the Virtual Client configuration.

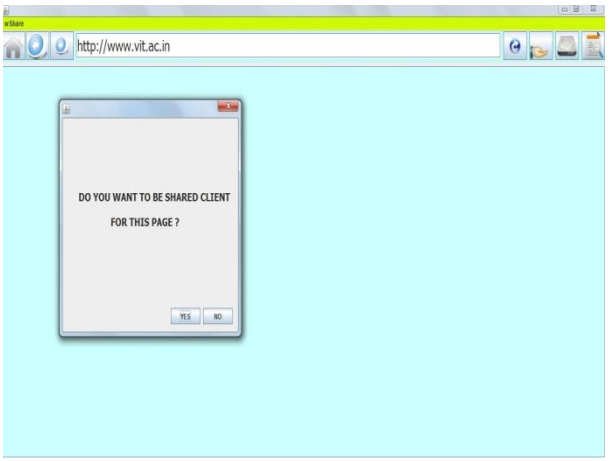


Figure8. Web-Browser Support for Virtual Client

The above figure shows how a virtual client can be made. It totally depends on the user to accept it or not.

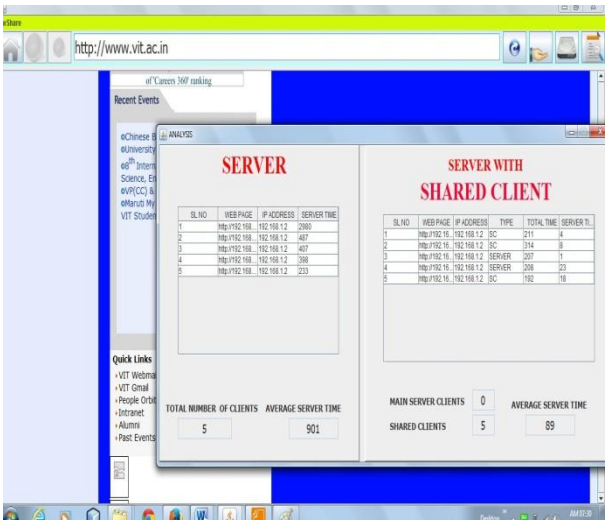


Figure9. Web page being shared with the client.

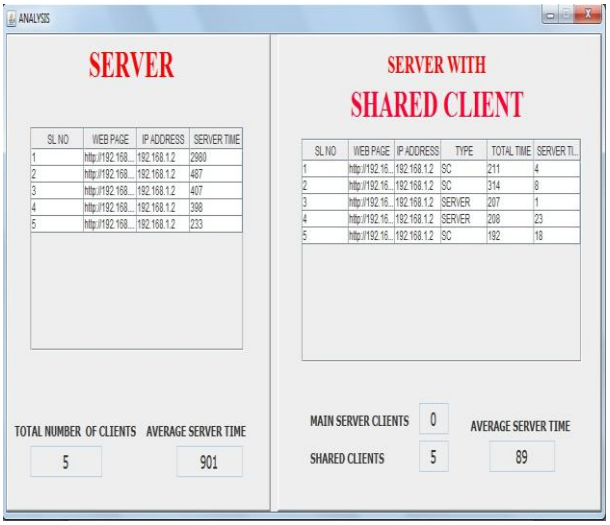


Figure10. The server and the Virtual (shared) Client.

10. CONCLUSION

This load balancing approach can be implemented at network level also by implementing the daemon routine in the Network Interface Card. The HTTP requests can be directly forwarded through the embedded code in the Network Interface Card. This type of load balancing mechanism can reduce the web server overloading and by utilizing the virtual clients with different servicing capabilities and processing speed thus, sometimes it would be very typical to get the optimum performance. Customer virtualization system utilizes the constant information and the band width of the server and in addition the virtual clients productively serve the customers in the web server system. The fundamental server system can be benefited by this client level virtualization in order to gain performance and for this the routing table needs to be very effective and dynamic. All in all this concept of client level virtualization rather than server level virtualization reduces the cost and effectively utilizes the system level resources.

11. ACKNOWLEDGMENTS

I would like to thank my mentor D.R Parvathi R for motivating me and guiding me throughout the work.

12. REFERENCES

[1] B Yingwu Zhu and Yiming Hu,"Efficient, Proximity-Aware Load Balancing for Structured P2P Systems". In Proceedings of the Third International Conference on Peer-to-Peer Computing IEEE 2003.

[2] Zhang Wensong, Linux server cluster system http://www.Ibm.com developer Works cn linux cluster lvs part4 index.shtml.

[3] Carlos Oliveira, "Load balancing on virtualized web servers". 41st International Conference on Parallel Processing Workshops 2012.

[4] Xingxuan Wang and Da-Zhong Zheng "Load Balancing Control of Web-server Clusters: N-Tanks Model and a CTCT Method". In Proceedings of the 7th World Congress on Intelligent Control and Automation June 25-27, 2008.

[5] Satoru Ohta and Ryuichi Andou "WWW Server Load Balancing Technique Based on Passive Performance Measurement".

[6] Ye Xia, Alin Dobra and Sueng Chul Han "Multiple-Choice Random Network for Server Load Balancing". IEEE INFOCOM 2007.

[7] Devarshi Chatterjee, Zahir Tari and Albert Zomaya."A Task-Based Adaptive TTL Approach for Web Server Load Balancing". In Proceedings of the 10th IEEE Symposium on Computers and Communications(ISCC) 2005.

[8] Megumi HISAYUKI, Shinji INOUE, Yoshiaki KAKUDA, Kenji TODA and Kuniyasu SUZAKI "Dynamic Load Balancing Using Network Transferable Computer". In Proceedings of the 25th IEEE International Conference on Distributed Computing SystemsWorkshops(ICDCSW05)2005.

[9] Tony Brouke, "Server Load Balancing", O'Reilly Publications. (2001).

[10] G. Teodoro, T. Tavares, B. Coutinho, W. Meira Jr. and D.Guedes,"Load Balancing on Stateful Clustered Web Servers". In Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing IEEE 2003.

[11] Idira Semwal and C. Edward Chow Web Load Balancing through more accurate server report.

[12] Valeria Cardellini, Michele Colajanni, Philip S. Yu Dynamic Load Balancing on Web-server Systems.