Design and implementation of server cluster dynamic load balancing based on OpenFlow

Zhihao Shang, Wenbo Chen, Qiang Ma, Bin WU
Lanzhou University Communication Network Center
Lanzhou,China
{shangzh11,chenwb,qma2011,wbin}@lzu.edu.cn

Abstract*—Nowadays, the Internet is flooded with huge traffic, many applications have millions users, a single server is difficult to bear a large number of clients' access, so many application providers will put several servers as a computing unit to provide support for a specific application, usually people will use distributed computing, load balancing technology to complete the work. A typical load balancing technique is to use a dedicated load balancer to forward the client requests to different servers, this technique requires dedicated hardware support, the hardware is expensive, lacks of flexibility and is easy to become a single point failure. There will be a new solution for load balancing with OpenFlow proposed., this paper mainly studies dynamic load balancing technology in the OpenFlow environment, the Controller collected the server running status through the SNMP protocol, and calculated the aggregated load of the severs according to dynamic load balancing scheduling algorithm, the OpenFlow switch will forward the client's request to the server whose aggregated load is smallest, thus minimize the response time of the web server. In the OpenFlow network environment, using this method can brings high flexibility without additional equipment.

Keywords—Load Balancing; OpenFlow; SDN; Load Balancing Scheduling Algorithm

I. Introduction

Currently the network traffic is very large, and grows rapidly, network congestion and server overloading has become a serious problem that we must face. This problem appears to be particularly important in the coming area of cloud computing and big data[1]. With the rapid growth of visiting traffic and data traffic, the processing ability of the core parts of the network should be increased correspondingly. In this case, if we settle it by abandoning the existing equipment, and purchasing new equipment to upgrade hardware, it will greatly increase the cost and waste resources. Load balancing technology can avoid unnecessary waste, load balancing technology is that distributes a large amount of concurrent access or data traffic to more than one equipment to process to improves server's processing ability and reduces the response time of users. Load balancing technology is mainly applied to Web server, FTP server, enterprise key application servers and so on[2].Such as the Web servers of Amazon's data center, there are 454400 servers in Amazon to provide services, they receive hundreds of millions level of requests every day. When user's requests arrive at the data center, a dedicated load balancer receives the requests and decides which server in the server pool to handle the request according to the load balancing strategies made in advance.

Traditional load balancer is very expensive (whose price is generally higher than $50k), the policy sets of load balancer needs to be set in advance, flexibility is very low, it cannot deal with emergency situations very well. It requires specialized administrator to maintain and cannot make flexible strategies according to its own actual network condition. Because all of the requests are transmitted by a single hardware load balancer, therefore, any failure on the load balancer will lead to the collapse of the entire site[3].

Due to the great differences between the peak load of Internet application and the normal load, traditional load balancing architecture is difficult to adapt to their needs. Under the traditional architecture, we either allocate excess configuration resources for potential peak, resulting in a large amount of waste of resources, or only allocate in accordance with the normal load configuration, thus it is unable to cope with the peak condition. In order to solve the above mentioned problems, we use a cheap OpenFlow Switch[4] which based on NetFPGA and an ordinary commercial server as a Controller to achieve an alternative dynamic load balancing architecture. The architecture is not only low costs but also can provide flexible programmable modules which is used to achieve the policy sets that appropriate for your network in the Controller. With the flexible configuration ability of this architecture, we can monitor and acquire the server's load in real-time, and distribute client's requests to the server which has minimal load in server pool according to Controller's load balancing strategy. With the arrival of the next generation of OpenFlow technology, an OpenFlow Switch can connect multiple Controllers; we can improve the robustness of the architecture by configuring any server which attached to the switch as the Controller[5].

1.1 The technology of traditional load balancing

Load balancing technology build on the existing network structure, it provides an effective and transparent method to improve the ability of network devices and servers, increase throughput, enhance network data processing capability and network flexibility and availability. It mainly completed the following tasks: solve network congestion problem, provide service nearby, and implement location independence. It provided a better user experience, improved the responding speed of servers and the utilization of server's resources [6].

The first load balancing technology is achieved through DNS. It configures a same name for multiple IP addresses, so the client who queries the name will get one of the addresses, thus allowing different clients to access different servers and achieve the purpose of load balancing, DNS load balancing is a simple and effective method, but it cannot distinguish the difference among the servers, and also cannot reflect the current running status of the servers [7]. Once a server broke down, even if modifying the DNS settings in time, it also need take a lot of time (refresh time) to let it work normally.

Subsequently, client-based load balancing appeared, client-based solutions require each client with the basic knowledge of server clusters, regularly or irregularly collects running parameters of server group, and sends the request to the server that can provide best service. Take Berkeley, for example, Smartclient is a client-based solution. The problem of this technology is versatility, it requires each client install a specific server information collection program.

In order to overcome this problem, server-based load balancing scheme was put forward, the most common method is the reverse proxy technology, using the reverse proxy server to forward client requests to different servers evenly, or send the cache to the client directly. For example, Squid proxy server and Netra proxy server are the representatives of reverse proxy technology. Reverse proxy technology can combine the load balancing and the high-speed cache technique of proxy server together, effectively reduce the response time of server. But it is required to develop a reverse proxy server for each service, which greatly increased the cost of development. In addition, the proxy server itself has very large load, finally the reverse proxy server itself may become the bottleneck of service.

Now many load balancers are based on NAT technology, this technology has a good advantage in cost and efficiency. Address translation gateway which is supported by load balancing technology, it can map an public IP address for multiple private IP addresses, for each TCP connection request ,it dynamically uses one of the private address, thus achieving the goal of load balancing[8]. Because the address translation was relatively close to the bottom of the network, so it is usually integrated in the hardware equipment, it will need to purchase additional hardware.

Due to the defects in the design of the Ethernet itself, traditional load balancing technology mentioned above has failed to achieve the ideal effect. The presentation of OpenFlow technology provides a new way to implementation load balancing; it can well solve the deficiency of traditional load balancing.

1.2 OpenFlow technology

Because at the beginning of Ethernet designing, it has not fully taken into account the development trend of future network, now Ethernet has shown some disadvantages, especially for doing some innovative network experiments on the existing protocol, the influence of experimental data on existing data become the biggest obstacle to the experiment. The generation of OpenFlow technology is to solve these problems, and OpenFlow has now become the representation of SDN, it opens up a new way for network innovation.

Now there are several universities have deployed OpenFlow network, including Stanford University, which has obtained perfect experimental result [9]. Unlike traditional switch to bundle hardware and software in a box, OpenFlow switch separates the data forwarding and routing control, respectively accomplished by OpenFlow switch and Controller. Eventually it achieved a more open, more flexible, easier to manage and scalable network. In order to achieve the purpose of controlling flow forwarding, OpenFlow switch forwards the packet which already arrived at the switch according to the flow table, and the Controller operates the flow table in the switch through OpenFlow protocol. The Controller decides how packets of a new flow should be handled by the switch. When new packet arrives at the switch, it gets redirected to the Controller which decides whether the switch should drop or forward it. The Controller can also delete or modify existing flow entries in the switch. The OpenFlow switch model is shown in Figure 1.

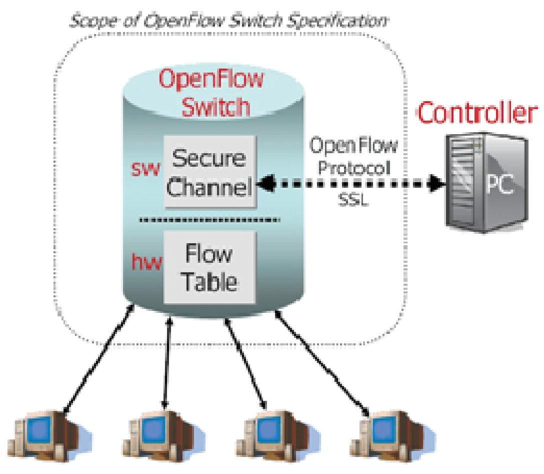

Figure 1.The idealized OpenFlow switch model [9]

All of the flow entries in OpenFlow switch are stored in the flow table. They consist of match fields, counters, and actions. Each flow entry stores Ethernet, IP and TCP/UDP header information. Match Fields are used to match the input packet fields, in OpenFlow1.0 Match Fields includes ingress interface, Ethernet source address, Ethernet destination address, Ethernet type, VLAN ID, VLAN priority, IP source address, IP destination address, and IP protocol type/APR operation code, IP TOS, TCP/UDP source port, TCP/UDP destination port. Each flow entry also maintains a counter of number of packets, and bytes arrived per flow. A flow entry can also have one or more action fields that describe how the switch will handle packets that match the flow entry. If a flow entry does not have any actions, then the switch drops all packets for the particular flow. Each flow entry also has an expiration time after which the flow entry is deleted form the flow table. This expiration time is based on the number of seconds a flow was idle and the total amount the time the flow entry has been in the flow table.

OpenFlow switches are like a standard hardware switch with a flow table performing packet lookup and forwarding. OpenFlow switch uses an external Controller to add rules into its flow table. That is to say, all operations of OpenFlow switch are controlled by the Controller. The Controller that this paper used is Floodlight [10]; it is an open SDN Controller which is developed by an open community of developers including a number of engineers form Big Switch Networks. It is designed to work with the growing number of switches, routers, virtual switches, and access point that support the OpenFlow standard.

II. THE DESIGN OF LOAD BALANCING ARCHITECTURE

2.1 Summary

The load balancing architecture this paper described consists of OpenFlow switch network with a Floodlight Controller and multiple servers connected to the ports of the OpenFlow switch. Each server equips two network interface cards, one connects to the OpenFlow switch, communicating with client through OpenFlow network; the other connects to the Controller, and communicating with Floodlight Controller by local network. The load balancing architecture in OpenFlow environment is shown in Figure 2.

As shown, Floodlight Controller maintains a server address pool and generates current network topology in real-time, it equipped a static IP address for each server interface connected to the OpenFlow network. In the Controller, it is equipped with a virtual address that provides external service. From the client's perspective, the server reflects to be an IP-based system image by OpenFlow switch service, all servers share this virtual address and through this virtual address, the client can put the whole system as a host system that has independent legal IP address, all access from the clients are sent to the virtual IP address. When the client sends a request packet to the switch, OpenFlow switch uses the packet header information to compare with flow entry in the switch, if the client packet header information matches up with the flow entry, and then increase the corresponding packet counters and byte counters, and forward the packet using actions in the flow entry which is matched it. If has no flow entry that matched up with the packet, the switch will forward this packet to the Floodlight Controller, and then let the Controller determine how the switch forwards the packet. The Controller adds corresponding flow entry to switch through OpenFlow protocol. Figure 3 shows the procedure of OpenFlow switch processing packet.

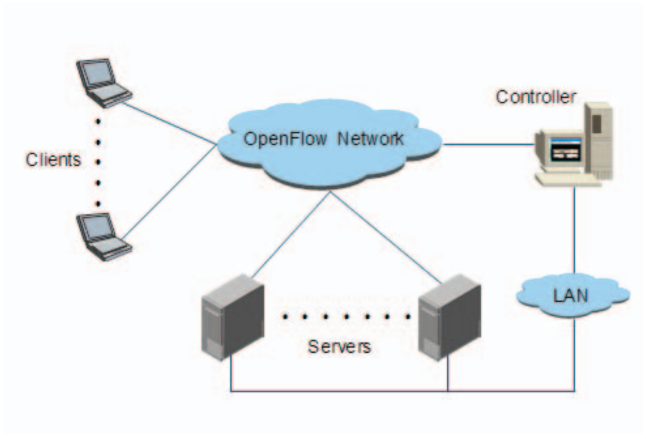

Figure2. The load balancing architecture in OpenFlow environment

2.2 Load balancing strategy

1) Random

For each new flow that is forwarded to the Floodlight Controller, the Controller randomly selects a server from a list of registered servers which server will handle the request[11].

2) Round Robin

For each flow that is forwarded to Controller, the Floodlight Controller selected a server to process the client request according to a certain order [12].

3) Server-based load balancing (SBLB)

a) Dynamic feedback the current server load, mainly collect CPU occupancy rate $L(C_i)$, memory occupancy rate $L(M_i)$, and response time $L(R_i)$,but this information does not directly represent the load of a node, so it needs a function to convert these indicators, and then got the load of the server $L(S_i)$:

$$L(S_i) = r_1 * L(C_i) + r_2 * L(M_i) + r_3 * L(R_i), \quad i = 0,1,\ldots,n-1; \text{sum } r_i = 1 \qquad (1)$$

Due to there are different type service, which can have different influence on each part of the node, therefore, we introduced the parameter r, it is used to emphasize different degree of influence that this type service to various parts.

b) Processing ability computation of server node: When we compute load balancing, if the service nodes are heterogeneous, we should not only consider the node load, but also must consider the node's processing capacity. For the processing ability $C(S_i)$ of node s_i, it is mainly considered from the following several indicators: CPU quantity n_i,CPU computation ability $C(C_i)$, CPU type, memory capacity $C(M_i)$, disk I/O rate $C(D_i)$, network throughput $C(N_i)$, maximum process number $C(p_i)$.

$$C(S_i) = k_1 * n_i * C(C_i) + k_2 * C(M_i) + k_3 * C(D_i) + k_4 * C(N_i) + k_5 * C(p_i), i = 0,1\ldots,n-1; \mathrm{sum}(k_i) = 1; \quad (2)$$

c) Calculating weight: When the server load balancing system comes into use, the administrator should set $\mathrm{DEFAULT_{WEIGHT_i}}$ as initial weight to every server, as the server load changes, the weight to be adjusted. In order to avoid the weight becomes a too large value, we set a range of weights that is $[\mathrm{DEFAULT_{WEIGHT_i}}, \mathrm{SCALE} * \mathrm{DEFAULT_{WEIGHT_I}}]$, where SCALE can be adjusted, and its default value is 10. Controller runs SNMP module periodically to queries the load parameters and calculate the aggregate load value $\mathrm{AGGREGATE\_LOAD_i}$. And using a sigmoid function to map aggregate load to$[0, +\infty)$, if the result is 1, we think the server is in the best condition. We introduce the following formula, depending on the server aggregate load value to adjust its weight.

$$w_i = w_i + A * \sqrt[3]{0.95 - \mathrm{AGGREGATE_{LOAD_i}}} \quad (3)$$

In the formula, 0.95 is what we want to achieve system utilization, and A is an adjustable coefficient (the default value is 5). If the aggregate load is 0.95 server weight will not change, if the aggregate load is greater than 0.95, it will become smaller, and if the aggregate load is less than 0.95, it will becomes larger. If the new weight is greater than $\mathrm{SCALE} * \mathrm{DEFAULT_{WEIGHT_I}}$, we set the new weight of $\mathrm{SCALE} * \mathrm{DEFAULT_{WEIGHT_i}}$. As can be seen that this is a dynamic feedback formula, it will make the weights adjusted to a stable point, such as a system to achieve the desired state, the weight will be constant. In practice, if it is found the weights of all the servers are less than their $\mathrm{DEFAULT_{WEIGHT}}$, the server cluster should be overloaded, then it need to add new nodes to the server cluster. Conversely, if the weights of all servers are close to $\mathrm{SCALE} * \mathrm{DEFAULT_{WEIGHT}}$, then the current system load is very small.

d) For each new incoming requests, it selects node to response the client by the following method. First, find out the server $S_m$which has the maximum value of weight, and then find out a server S_i, the weight of which is fitted $W(S_i) > W(S_m) - f, i = 0,1,\ldots,n-1$,where f is a constant, then add $S_i$to collection J. After that we calculate the probability $P(S_k)$ for each sever in collection using formula (4).

$$P(S_k) = C(S_k)/\mathrm{sum}\, C(S_i) \quad (4)$$

Then the system selects a server based on the probability.

III. Demonstration and Results of Data Analysis

Our load balancing architecture consists of an OpenFlow switch based NetFPGA with a Floodlight Controller and three server machines connected to the ports of the OpenFlow switch, as is illustrated in Figure 3.

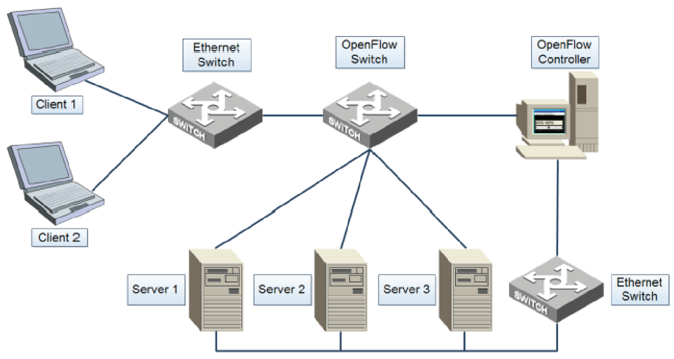

Figure 3. Load balancing architecture using OpenFlow switch based NetFPGA with a Floodlight Controller

The OpenFlow switch uses one interface to connect to the Ethernet switch. Three Web servers in the network map to the same virtual host and provide the same services. Since all client requests are destined for the same IP address. When a packet from a client arrives at the switch, Floodlight then decides on how the packet for this flow should be handled by the switch according to the load balancing strategy. The Floodlight then inserts a new rule into the switch's flow table using the OpenFlow protocol. The rule is that modify the destination MAC and IP address of the request packet with a server's MAC and IP address. After the packet's header is modified, the switch forwards the packet to the output port of the server. When the servers send back a packet to the client, Floodlight adds a flow entry that modifies the source IP address with the IP address of the hostname that the client sends its request to. Hence the client always receives packets from the same IP address. When the connection of client and the server is established, all packets will be forwarded at line speed.

To reduce the average response time of the server, we wrote SNMP module and LoadBalancer module that are executed by the Floodlight Controller. The SNMP module is responsible for using snmp-request message of SNMP protocol to collect each server's load information every 5 seconds and save it to the database. If the server does not respond within a certain period of time that the server is considered be overloaded state. SNMP module tells LoadBalancer reduce the weight value of the server. The LoadBalancer module is responsible for extracting current load information from servers and calculating an aggregate load value according to the load balancing strategy. The LoadBalancer calculates a new set of weight values using each server's aggregate load value and current weight values. When aggregate load value indicates that the server is busy, new calculated weight value will be smaller than the current weight value, such that allocated number of requests to the server will be less. When aggregate load value indicates that the server is in a low utilization, new calculated weight values will be larger than the current weight values, such that allocated number of requests to the server will be added. It selects a range of lightly loaded server to join the candidate set, according to a certain probability distribution of client requests to the server. LoadBalancer module send the selected results to Floodlight, Floodlight then inserts a new rule into the OpenFlow switch's flow table according to the selected results. In this way, flexible and low latency service can be provided to clients. In addition, Floodlight also maintains an entire network topology and real-time flow table information of OpenFlow switch. Administrators can grasp the whole network node information, make up the shortcomings that traditional network is transparent for administrators, as illustrated in Figure 4.

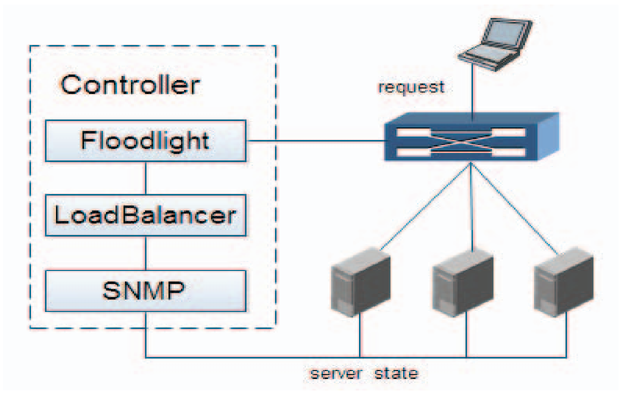

Figure 4. Three functional units implemented in Floodlight Controller

In our experiment, two clients send request to servers which publish web pages for every 0.2 seconds using the virtual address. OpenFlow switch queries the flow table after it receives a packet, if there is no flow entry matching the packet, it sends a PacketIn message to the Floodlight Controller. If destination address and port in the requested packet equal with the virtual host address and port which is used to publish web pages, Floodlight calls the LoadBalancer module, and the LoadBalancer module calculate the current minimum load server using SBLB load balancing strategy, and passes the result to Floodlight. And then, Floodlight adds a flow entry to OpenFlow switch based on the returned result, OpenFlow switch could establish a connection between the client and servers, and forward the client's request to a server. This mechanism can achieve effective dynamic load balancing for servers.

To assess the response time and usage of server resources, LoadBalancer module uses Round Robin, Random and SBLB load balancing strategy respectively. We got the average response times are 1.1827s, 1.1325s and 0.9631s, by detailed analyzed of the experimental data using the three strategies. As can be seen, using SBLB load balancing strategy which is designed by this paper, servers got the smallest average response time.

In order to see advantages of SBLB load balancing strategy more clearly, we analyze response time by using different load balancing strategies for each server, the response time of server1 is shown in Figure 5.

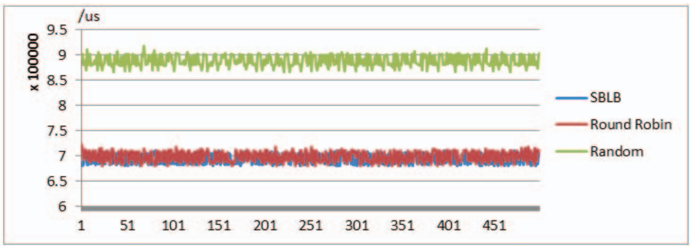

Figure 5. The response time of server1

The response time of server2 is shown in Figure 6.

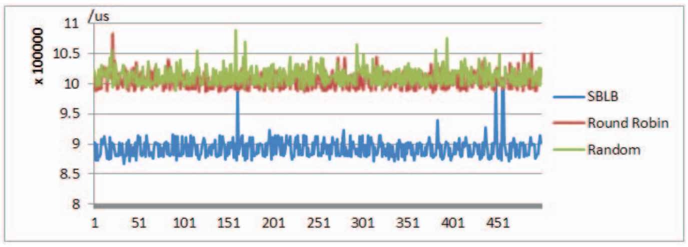

Figure 6. The response time of server2

The response time of server3 is shown in Figure 7.

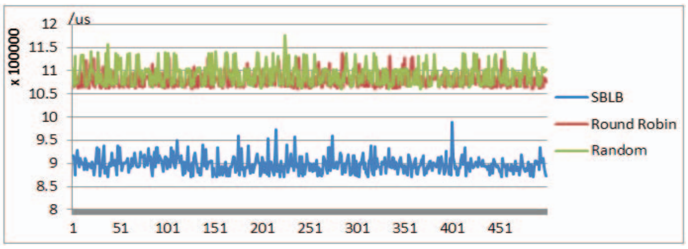

Figure 7. The response time of server3

The three figures above show response time of servers, where x-axis represents the number of 500 records, which is got from server log, and y-axis represents the server response time, in unit of microseconds. SBLB algorithm is slightly better than the Round Robin algorithm as shown in Figure 5, but compared with Random algorithm server response time is much smaller. As illustrated in Figure 6 and Figure 7, SBLB algorithm has a considerable advantage. So we can conclude that using SBLB algorithm as OpenFlow-based server cluster dynamic load balancing strategy not only saves money, has a flexible strategy, but also can significantly reduce server response time.

To illustrate SBLB algorithm is more reasonable than Round Robin algorithm and Random algorithm about resource scheduling, we analyze the CPU load information and memory information extracted from the different servers and get the following histograms.

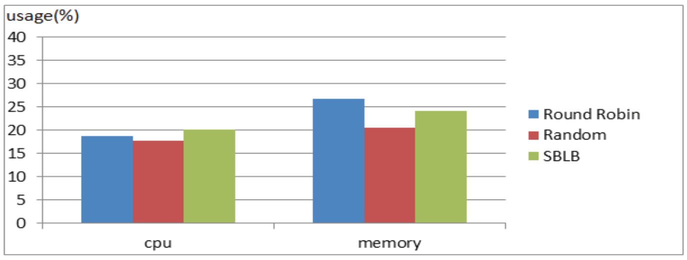

Figure 8.The load scheduling of Server1

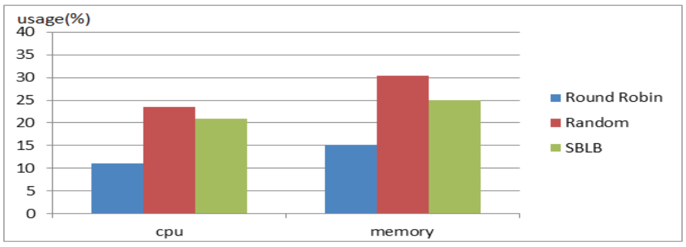

Figure 9.The load scheduling of Server2

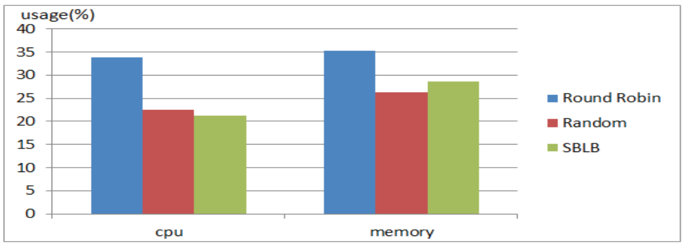

Figure 10.The load scheduling of Server3

As shown above, where x-axis represents CPU load information and memory load information, and y-axis represents the usage of CPU and memory. It is obvious to see that using SBLB strategy can make resources scheduling more reasonable. The utilization of CPU roughly about 20% and memory utilization roughly about 25%. Three servers CPU usage and memory usage is basically the same, while Round Robin and Random is used, the allocation of resources have a greater instability. It is illustrated that using SBLB strategy can make resource scheduling more reasonable, avoid load imbalance between servers, and improve the system resource utilization.

IV. CONCLUSION

This paper presents an OpenFlow-based dynamic load balancing server cluster architecture to solve the problems that traditional dedicated load balancer faced. Experiment shows that our load balancing architecture consists of an OpenFlow switch based NetFPGA with a Floodlight Controller is more flexibility and low-cost. This can replace traditional dedicated hardware load balancer. The OpenFlow switch provides the flexibility to implement arbitrary load balancing policy in software and decouple policy from the switch itself. It is also convenient for us to use different load balancing strategies in different network environment. Compared with Random algorithm and Round Robin algorithm uses our SBLB load balancing strategy can not only improve the response time of Web servers, but also can be more rational deployment of resources. These achieve a more efficient load balancing.

V. REFERENCES

[1] Theophilus, Benson. Understanding data center traffic characteristics[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(1): 92-99.

[2] Cardellini, V, Rome, Univ, Italy, Colajanni, M,Yu, P, S. Dynamic load balancing on Web-server systems[J]. Internet Computing, IEEE , 2002, 3(3): 28-39.

[3] Foundry ServerIron Load Balancer. http://www.foundrynet.com/products/webswitches/serveriron/.

[4] T.O.Consortium, "Openflow switch specification/version 1.1.0," February 2011, http://www.openflow.org/wp/documents/.

[5] M. Koerner, O. Kao. Multiple service load-balancing with OpenFlow.IEEE HPSR, 2012.

[6] Ruud, Schoonderwoerd, Owen, E, Holland, Janet, L, Bruten, Leon, J, M, Rothkrantz. Ant-Based Load Balancing in Telecommunications Networks[J]. SAGE, 1997, 5(2): 169-207

[7] DNS Support for Load Balancing: http://xml2rfc.tools.ietf.org/html/rfc1794

[8] Fanglu Guo ,Jiawu Chen , Wei Li ,Tzi-cker Chiueh. Experiences in building a multihoming load balancing system.Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, 2004

[9] N. McKeown,T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson,J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation incampus networks," ACM SIGCOMM Computer Communication Review,April 2008.

[10] FloodLight.http://docs.projectfloodlight.org/display/floodlightcontroller/Floodlight+Documentation

[11] Murata,Y., Inaba,T., Takizawa,H., and Kobayashi, H.Adistributed and cooperative load balancing mechanism for largescale P2P systems.SAINT-W, 2006.

[12] Adabala,5.,Chadha,V.,Chawla,P.,and Figueiredo, R.From virtualized resources to virtual computing grids: the in-vigo system. Future Generation Computer Systems,6(21),2005

Top

Program Guide

author Index

iCAST 2013
Technical Program

UMEDIA 2013
Technical Program