ISSN
Volume 1, No.1, September – October 2012
International Journal of Science and Applied Information Technology
Available Online at http://warse.org/pdfs/ijmcis01112012.pdf

Efficient DNS based Load Balancing for Bursty Web Application Traffic

Mei Lu Chin[1], Chong Eng Tan[2], Mohamad Imran Bandan[3]
[1, 2, 3]Faculty of Computer Science & Information Technology,
Universiti Malaysia Sarawak,
94300 Kota Samarahan, Malaysia.
meilu850422.chin@gmail.com[1] ,cetan@ieee.org[2], bnimran@fit.unimas.my[3]

ABSTRACT

This research proposes a new efficient load balancing algorithm which applies to the local Domain Name Service (DNS) server for web based applications and services to ease the sudden increase in demand for the services. Owing to the existing load balancing algorithms still experience server's resource congestion and slow connection to the system resulted by sudden bursty demand of services especially during special events. This is mainly due to the unbalanced distribution of workload and the insufficient of physical computing resources in service provision. To overcome this problem, most web based application service providers will have to constantly improve the capacity of their physical computing resources by either adding new server nodes to the existing server farm or renting cloud computing resources from cloud computing service provider to meet the sudden demands of the end users during the peak period. However, it is not economical to maneuver and reconfigure huge amount of permanent computing resources just to satisfy the instantaneous and short period of service demand. As a result, the need to have a more efficient load balancing algorithm which can adaptively utilize the resources available in the farm of computing resources will be of advantageous. The new algorithm will be able to directly decrease the operation cost and web services will no longer be interrupted by sudden high demand of traffic request. The proposed algorithm is evaluated via computer simulation and modeling where its performance is verified against the few selected algorithms of the same nature. Enhancement on the DNS system for load balancing is beneficial to most organizations such as government agencies and service providers running their own local DNS service, which allow the proposed algorithm to be easily implemented. Moreover, DNS setup is standard across the IP networks hence the adoption can be easy achieved with minimal changes without altering the architecture of the services provided especially in coding as well as physical set up of the server farm itself.

Keywords: Computing Resources, Load Balancing, local DNS, Performance Parameters, Sudden Traffic Web Application Services**.**

1. INTRODUCTION

Nowadays, the web application has become a common instrument in daily life, due to it easy accessibility through of operation such as towards the datelines, during promotion period and etc. During these critical periods, millions of users will be accessing a single Internet service site at the same time creating instantaneous bursty traffic to the site.

Therefore, some organizations have very little options but to spend lots of money upgrading their server farm or to purchase additional computing resources so that the problems of server traffic congestion and bursty demand can be resolved. It is not economical to acquire huge amount of permanent computing resources just to satisfy the instantaneous and periodical service demand. As a result, an efficient load balancing approach has been proposed to enhance the DNS service for handling burst demand in Web based application services.

Load balancing is used to distribute the load among overloaded servers to underutilized servers so that all of the servers in the cluster can be fully utilized. The conventional Round Robin type DNS load balancing mechanism has a few disadvantages which do not necessary reducing the server congestion and burst demand problems under certain environment, but may increase congestion on specific servers.

For example, the DNS service does not aware of the real-time status of each server in the cluster. This means that the DNS server does not know whether a server in the cluster is congested or down and will still keep directing the client requests to a problematic server.

Secondly, DNS service does not take into consideration of the computing capability of a server in its operating procedure, regardless of whether the server is overloaded or underutilized. Lastly, the DNS caching takes days to propagate and updates due to the DNS servers cache the records and will not update it until the Time to Live (TTL) expires. TTL is used to determine the duration a DNS record will be cached by the DNS server. Therefore, it can take hours or days for the whole Internet to recognize changes in the DNS information for a particular domain.

Consequently, enhancements to the current Round Robin DNS load balancing approach are needed to ease the current limitations faced by the bursty traffic environment today. First of all, the request should be distributed not just in

@ 2012, IJMCIS All Rights Reserved

round robin order but to be distributed intelligently based on the current server load and its relative performance matrix.

Second, a server status reporting mechanism to report and update the real-time status of servers in the cluster to the DNS server is mandatory where a failed server can be identified and removed from the list of operational servers. When a new server is added, it can be included immediately to serve the client requests.

By having such real-time reporting mechanism, any temporary server can be added to the server farm seamlessly; computer systems either from the office, lab and etc, can be borrowed for a certain period of time to ease the sudden heavy traffic loads to the servers. Once the peak period is over, the temporary computer systems can return to its normal operation tasks. Lastly, the DNS caching issue can be resolved by adopting a local DNS service dedicated to the server farm.

2. BACKGROUND

Each of the nodes in the server farm has its own system resources and processing power [1]. Therefore, load balancing performs an important position in distributed system to ensure the response time can be minimized and the performance of each node is improved so that the system resources are fully utilized.

Consequently, a suitable load balancing algorithm needs to be adopted in order to achieve better performance in various aspects. This is because different types of load balancing algorithms are aimed to achieve different types of outcomes and objectives **[2]**. In general there are two categories of load balancing algorithm, namely system stateless algorithms and server state based algorithms.

2.1 System Stateless Algorithms

System stateless algorithms do not taking any system state information, for example system state information or server load or a combination of them into account in request distribution. Round Robin DNS (DNS-RR) was the first distributed homogenous Web-server architecture that derives from this class **[3]**.The purpose of this algorithm was to support the fast growing demand of the National Center for Supercomputing Applications (NCSA) site.

By implementing the DNS-RR, the DNS of the NCSA domain is required to modify to meet its address mapping. However, the address caching mechanism resulted the unbalanced of load distribution under DNS-RR. The reason is that the DNS can only control a small fraction of the requests via this mechanism which leads to large number of end users from a single domain are assigned to the same Web-server [4][5]. Hence, the loads are distributed imbalance among the server nodes and this indirectly causes overload to the server nodes.

Furthermore, DNS-RR ignores the server performance capacity as well as its availability to serve. The algorithm will continue sending the requests to the overloaded or fault servers due to the use of cached address of the server. It also assumes all of the servers are equally capable in serving client request which assume as homogenous server environment. Owing to that, there is a strong need for the study of a replacement to the DNS-RR scheme.

2.2 Server State Based Algorithms

If the system state conditions of the servers are known, the unreachable servers which are either overloaded, congestion or faults can be then excluded from the task distribution **[3]**. In this category, the DNS policies will combine with a simple feedback alarm mechanism that applied in the highly utilized servers to inform the DNS which of the Web-server is overloaded so that during the overloaded period the subsequent request is excluded from those unreachable servers.

In addition, lbmnamed algorithm proposed to use the current load of the Web-server nodes for scheduling decision. The server with the lightest load will be selected to receive the address request but the DNS of lbmnamed is required to set the TTL value to zero to avoid the address caching at name servers. However, the setting of TTL value to zero also has its drawback because nearly all intermediate name servers are configured in a way do not accept very low TTL values [3].

Hence, further research is required so that a more efficient load balancing can be developed to achieve more effective task distribution and at the same time decreases the waiting time at the end user end.

3. THE PROPOSED ENHANCED ALGORITHM

In this paper, a new load balancing algorithm is proposed by introducing a new performance indexing mechanism and a task distribution scheduler at the local DNS server. The flow chart of the proposed algorithms is shown in Figure. 1.

The proposed load balancing algorithm is based on a performance indexing of each node in the server farm in order to select the most suitable nodes for the next task distribution. The performance indexing takes into account the current server load and the server performance metric of the server itself.

The load metric indicates the current server workload of a server and the performance metric indicates the performance capability of a server [2]. The task distribution scheduler at local DNS server will be updated by performance indexing algorithm running at each server so that the latest workload information and performance metric from the nodes can be used by the task scheduler for the next client request distribution.

@ 2012, IJMCIS All Rights Reserved

The local DNS server task scheduler will look into the latest performance metric table when it received service request from the end users. From there it will determine which server IP address should be resolved to the client request based on a good balance in current workload and performance metric of the each server. Thus, the client request will be directed to the server which presently has the best index.

Under this algorithm, each server in the server farm will have to perform a self-load evaluation from time to time based on their current workload and computing performance ability. The information is then reported to the task distribution scheduler at the local DNS server periodically.

The node will be defined as a fault or dead server when the local DNS does not receive any report from the nodes within the specified time frame. Anode is defined as healthy until the next report is received by the local DNS server in the next update. As a result, the DNS lookup table for the server in server farm is completely based on the reports received from all the servers within the server farm.

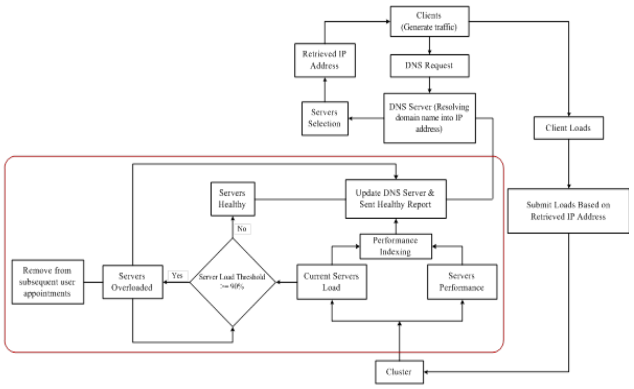
Clients (Generate traffic)
Retrieved IP Address
DNS Request
Servers Selection
DNS Server (Resolving domain name into IP address)
Client Loads
Servers Healthy
Update DNS Server & Sent Healthy Report
Submit Loads Based on Retrieved IP Address
No
Performance Indexing
Remove from subsequent user appointments
Servers Overloaded
Yes
Server Load Threshold >= 90%
Current Servers Load
Servers Performance
Cluster

Figure 1: The proposed algorithm

Once the server workload is more than or equal to the defined threshold for server overloaded then the respective server will be removed from the subsequent client request distribution. It will only be assigning a new client request when it has reached a healthy stage or in another mean, the server workload is less than overload threshold. Of course, the server after being removed from the list of distribution will still be serving its existing client request until the entire service session is completed [2].

The proposed performance indexing algorithm has an advantage where the client requests are distributed based on the overall server performance metric which is a more accurate measurement of the server performance compare to the conventional one that is based on round robin ordering. Besides, through the reporting mechanism, the local DNS server is aware of the current performance status of each server in server farm.

Moreover, the server farm can be expanded based on the needs during certain critical or peak period. For example, the server can be dynamically added to or remove from the server farm depending on the client request volume and the overall workload of the entire server farm. Additional server can be taken from the existing computing resources to act as server and release back to its normal operation after the critical period. This helps to minimize unnecessary investment in upgrading the computing resources just to accommodate very short period of bursty service demand.

The proposed algorithm can adaptively distribute the task base on the current server farm status. Besides, the use of a local DNS server at the service provision domain can resolve the DNS address caching problem. Therefore, the right task will be given to the right server for more efficient task execution.

To further test the performance of proposed algorithms, an adaptive algorithm is proposed so that it can adaptively base on the current environment status and numbers of servers in server farm to distribute the tasks.

4. SIMULATION AND PRELIMINARY RESULTS

Figure 2 shows the simulation model for a simple network configuration used in our simulation scenario. This network consists of a local DNS server and a server farm, which can be expanded based on the load at a certain peak period.

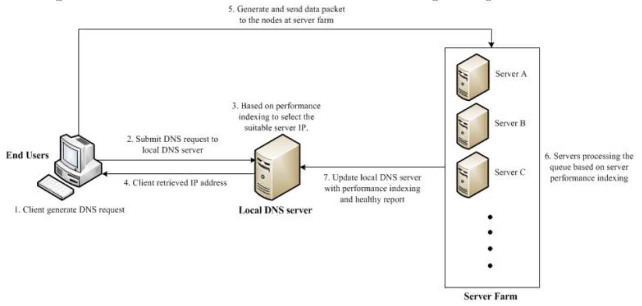
5. Generate and send data packet to the nodes at server farm
Server A
3. Based on performance indexing to select the suitable server IP.
Server B
2. Submit DNS request to local DNS server
End Users
6. Servers processing the queue based on server performance indexing
Server C
4. Client retrieved IP address
7. Update local DNS server with performance indexing and healthy report
1. Client generate DNS request
Local DNS server
Server Farm

Figure 2: Simulation model

The simulation started with randomly generated load metric while the performance metric is based on the overall system score that are collected from the NovaBench for all nodes in the server farm. NovaBench is a software that tests a computer system by benchmarking its CPU, RAM, hard disks read and write, graphics and other system parameters. The overall system score and system component results will be compared to the others. All of this information will be used to update the task scheduler of the local DNS server.

In addition, the client requests and sessions are randomly generated throughout the simulation. It have been carried out for the proposed performance indexing mechanism, conventional Round Robin DNS and server state base algorithm based on the same client traffic model and also the same load and performance metric.

Table 1 shows the total task completion time based on different environment by using of different algorithms and number of servers. Based on the results, the server farm which are considered as having more computing resources will complete the tasks faster. Moreover, the higher the

@ 2012, IJMCIS All Rights Reserved

server performance the faster of tasks completion compare to low server performance.

Besides, the proposed performance indexing algorithm has better result compare to other algorithms except for adaptive algorithm. The reason is the adaptive algorithm used the combination of performance indexing and server performance indexing algorithm. Thus, it completed the entire task slightly faster than them. As a result, all tasks can be completed in a comparatively shorter time as compared to the others existing algorithms.

Table 1: Total task completion time based on different environment

Algorithms	Adaptive Algorithm	Performance Indexing	Server Performance	Current Server Load	Round Robin
Number of Servers (size)	Total Task Completion Time (unit time)				
1	7703.763069	7703.763069	7703.763069	7703.763069	7703.763069
2	3267.042265	3267.034474	3269.173716	3393.899879	3562.241246
3	2061.937763	2062.371891	2063.564588	2126.008958	2226.679269
4	1507.293056	1507.757422	1510.104463	1549.488251	1619.671
5	1183.077901	1183.130711	1185.440747	1212.650209	1264.983687
6	953.1432528	953.1813683	955.8157794	977.3147382	1021.122817
7	797.8009093	797.9445393	800.1960311	816.9912848	853.5580783
8	689.8976226	689.9369911	692.2341753	705.0640319	735.8686731
9	599.3808374	599.4340592	601.5148979	612.0136769	639.5631087
10	489.6709097	490.0086176	491.6346716	519.7695114	560.2296284

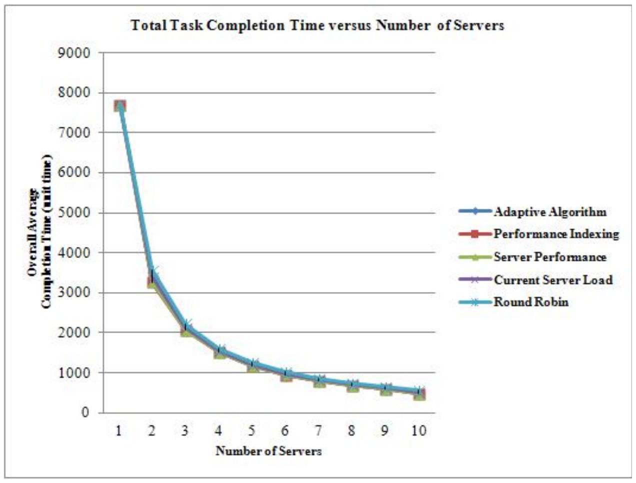

Figure 3: Comparison totaltask completion timebased on different environment

The server farm that contains more servers will be processing the tasks even faster than server farm which has fewer servers. This will decrease the sudden traffic and server overloaded during the peak period. The comparison of the total task completion time versus number of servers based on different algorithms is shows in Figure 3.

Table 2 shows the overall average task completion time. From the table, the performance indexing algorithm can process the task faster compare to the existing algorithms. This is because it can distribute the tasks evenly among the servers based on the environment needs. While for the adapative algorithms, it will process the task slightly faster than performance algorithm due to it has ability to adapt the current environment status and numbers of servers in server farm.

Table 2: Overall average task completion time

Algorithms	Adaptive Algorithm	Performance Indexing	Server Performance	Current Server Load	Round Robin
Overall Average Task Completion Time	1925.300759	1925.456314	1927.344214	1961.696361	2018.768058

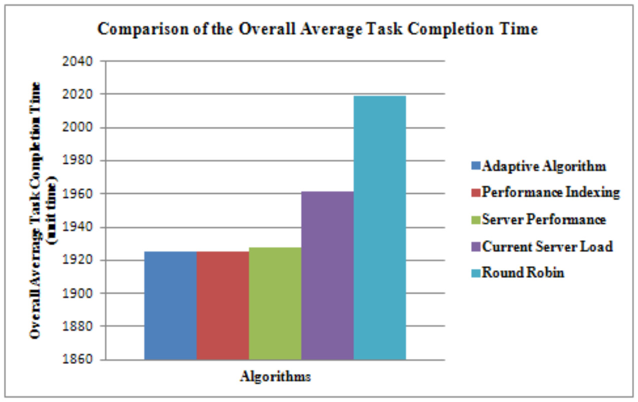

Figure 4 : Comparison overall average task completion timebased on different environment

Figure 4 shows the comparison of overall average task completion time based on different environment by verified against the selected few existing algorithms. Round Robin and current server load algorithm needed longer time to complete all the task compare to another three proposed algorithms. Adaptive algorithm processed the tasks shorter than performance indexing and server performance algorithm.

5. CONCLUSION

As the conclusion, higher system efficiency can be achieved by introducing the load balancing scheme which is optimized based on the right performance parameters that are taken into account in requests distribution. By looking at the preliminary result, the proposed algorithm has improved the load distribution among nodes in local DNS by the introduction of the performance indexing metric.

ACKNOWLEDGEMENT

Acknowledgement to Universiti Malaysia Sarawak, UNIMAS and Kementerian Pengajian Tinggi Malaysia in funding this research project through Fundamental Research Grant Scheme (FRGS).

@ 2012, IJMCIS All Rights Reserved

REFERENCES

1. Mohammadpour P., Sharifi M. and Paikan A. **A Self-Training Algorithm for Load Balancing in Cluster Computing**, Fourth International Conference on Networked Computing and Advanced Information Management, 2008.

2. Mei Lu Chin, Chong Eng Tan and Imran M. **Efficient Load Balancing for Bursty Demand in Web based Application Services via Domain Name Services**, 8thAsia-Pacific Symposium on Information and Telecomunication Technologies (APSITT), 2010.

3. V. Cardellini, M. Colajanni, and P. S. Yu. **Dynamic Load Balancing on Web-Server Systems,** *IEEE Internet Computing*, Vol. 3, No. 3, pp. 28-39, May-June 1999.

4. M. Colajanni, P.S. Yu and D.M. Dias. **Analysis of task assignment policies in scalable distributed Web-server Systems**, IEEE Trans. on Parallel and Distributed Systems, Vol. 9, No. 6, pp. 585-600, June 1998.

5. D.M. Dias, W. Kish, R. Mukherjee and R. Tewari. **A Scalable and Highly Available Web-server**, Proc. of 41st IEEE Computer Science Int'l. Conf, (COMPCON'96), pp. 85-92, Feb. 1996.

@ 2012, IJMCIS All Rights Reserved