STUDY OF SERVER LOAD BALANCING TECHNIQUES

Priyesh Kanungo[1]
Professor and Senior Systems Engineer (Computer Centre)
School of Computer Science and Information Technology
Devi Ahilya University
Indore-452001, India[1]
Email: priyeshkanungo@hotmail.com

Abstract— One of the critical scheduling problems in distributed computing environment is load balancing on a cluster of replicated servers which face a constant pressure of increased network traffic and diverse load levels. The key issue in server load balancing in a DCS is to select an effective load balancing scheme to distribute clients' requests to the servers. In this paper, we have investigated the problem of server load balancing and evaluated various server load balancing policies. We have also conducted simulation study to compare the performance of various policies.

Keywords-server load balancing, admission control, stateful servers, weighted round robin, shortest queue, diffusive algorithm

I. INTRODUCTION

In client server environment, clients are usually large in number. Much of the processing work of clients is now being shifted to servers which are primarily used for providing web services. Web servers are the means of interoperating between different software applications running on variety of platforms, operating systems and programming languages [29]. Most of the commercial application servers support web services. Web services are presenting enormous opportunities as well as a number of challenges by fundamentally changing the method of doing business and recasting the vendor customer relationship. More and more businesses are deploying network solutions being used by increasing number of people. With the phenomenal growth of IP traffic due to market expansion, server sites are overwhelmed with processing load. Even small companies have to establish their Internet and Intranet presence to survive. In this paper, we will study the methods of performance improvement in server cluster with the help of DLB techniques [18].

Although, in recent years, both network and server capacities are improved, web applications are no more used for simple communication and browsing for getting static information. WWW has become the medium of conducting personal and commercial transactions that require dynamic computation and secure communication with large number of servers through the use of middleware and application software. With the increase in heterogeneous client devices and network bandwidth, the use of techniques for improving performance of web server system has become necessary. There are essentially two ways for server sites to manage increased traffic; deploy a more powerful server or add additional servers to a cluster of replicated servers without disrupting service [5,13,21].

For a server cluster to achieve its high performance and high availability potential, DLB technique is required. Combining load balancing with cluster of low cost servers is a cost effective, flexible and reliable strategy to support web-based services. Load balancing optimizes request distribution among servers based on factors like server capacity, availability, mean response time, current load, historical performance and administrative weights. It also improves the scalability and overall throughput of the distributed computing system [2,11].

Main advantages of using load balancing in server cluster are[18]:

(a) 24X7 availability with consistent response time and resource availability without failure.

(b) Manageability and monitoring of server cluster to suit different needs and requirements.

(c) Performance improvement by evenly distributing the clients' requests among the servers in the cluster.

(d) Scalability so that more servers can be added or removed from the cluster dynamically in a transparent way.

(e) Cost effectiveness as compared to using a single costly server.

II. STEPS IN SERVER LOAD BALANCING

Load balancer sits between internet and the server cluster. It intercepts client requests transparently before they are dispatched to a server. Upon arrival of a request, it takes instantaneous intelligent decision about the server which can best satisfy the request. Thus, the load balancer is a middleware, which distributes client workload equitably among various backend servers in order to obtain the best response time for a workload [6,14].

Load balancing may even be supported by admission control mechanism, which controls the rate at which new requests from clients are accepted for processing by the web servers. The requests that may result in bottleneck

are not admitted in the cluster system. Admission control should be performed as early as possible, as, by the time the request is rejected, it might have already wasted significant resources. Incorporating admission control in server load balancing scheme reduces processing load of the servers and further improves their performance. This ensures that accepted requests receive a good response [8,15]. Admission control mechanism uses various performance measures of the servers, for example, server queue length, server utilization factor, memory consumption etc. for accepting or rejecting clients' requests. For this purpose servers' performance levels has to be periodically monitored [4]. Load balancing mechanism distributes the incoming requests across the web servers in the cluster in proportion to their capacity [23,24]. Steps in balancing load on server cluster are shown in Figure 1.

These steps include:

(a) The client sends a request which is intercepted by the load balancer transparently.

(b) The load balancer collects the state of the servers.

(c) The load balancer selects an underloaded server.

(d) The load balancer redirects the client's request to the newly selected server.

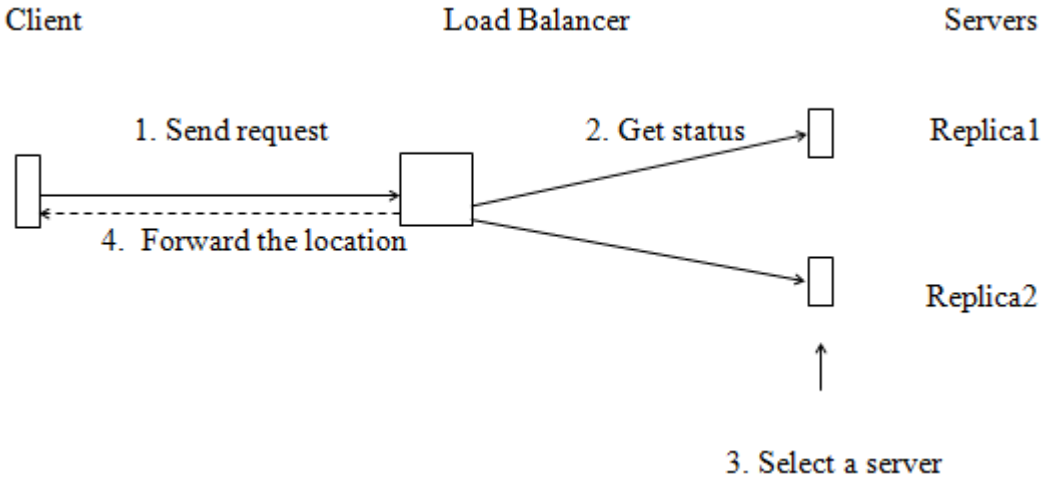


Fig. 1. Steps in server load balancing

Load balancing algorithms may be state blind or state aware. In state blind policies, the dispatcher assigns requests to servers using static information. No dynamic information is used. *Random* and *round robin* are the state blind policies. In *random* allocation, the incoming requests are distributed uniformly to the server nodes with equal probability of reaching any server. *Round robin* method uses a circular list and pointer to last entry for making dispatch decisions. Modern day clusters are being developed with heterogeneous computers, a number of interaction devices and variety of communication medium. Randomized load distribution schemes may not be sufficient. The load balancing system should be able to support a heterogeneous system of servers whose configuration may vary frequently. Configuration may change as a result of addition or removal of servers, server breakdown problem or link failure. *Weighted round robin* technique is used as a variation of *round robin* in which each server has integer weight in proportion to the it's capacity. The servers are assigned requests in proportion to their weights [7,10].

In state aware policies, the dispatcher makes use of state information received from the client and/or server. Server state aware algorithms use server information e.g. server load to assign requests to the servers. The *shortest queue* algorithm and the *dynamic weighted round robin* techniques are the examples of state aware algorithms. In the *shortest queue* technique, the server with minimum load is selected for dispatching the current request. In the *dynamic weighted round robin* technique, dynamic weights are assigned to the servers in proportion to the server state. Weights are computed periodically and incremented when a new connection is assigned. Client state aware algorithms are more sophisticated as they examine the HTTP request. Information in the URL may be used for different purposes e.g. cache affinity to use locality of reference or to make use of services provided by some specialized servers [12]. In client and server state aware policies, the dispatcher assigns requests to the servers on the basis of combined state of the server and the client. Client state aware policies are easier to implement as compared to server state aware policies [3,5].

III. REQUIREMENTS OF SERVER LOAD BALANCING ALGORITHM

Main requirements of a good load balancing service are [24]:

(a) Replication transparency of servers: For improved performance, scalability and reliability, distributed applications are replicated on many servers. But existence of multiple servers must be concealed from users and

programmers. Transparency is one of the major design goals of the DCS. Load balancing service should be designed to communicate with the applications and accept load control requests from it without modifications in server application software.

(b) Stateful servers for distributed applications: A stateful server maintains the current status of the requests between subsequent calls by the client to the server. In case of stateless servers, servers does not maintain any information about clients. Stateless servers have distinct advantages of scalability and fault tolerance. However, a load balancer must have state information of the replicas, particularly in heterogeneous environment, for marshalling operation which is required in case of difference in data representation formats [28].

(c) Fault tolerance using decentralized load balancing: Centralized load balancing algorithms are simple. But in case of failure, load balancing service will be disrupted. Fault tolerance in load balancing may be achieved by using decentralized load balancing. This will also enhance scalability and reliability of the load balancing system.

(d) Diverse load monitoring algorithms: The load level on a distributed application may vary frequently within a given period of time. These variations may be unpredictable. In case of different load conditions, it is desirable to use different load balancing algorithms. For example, in case of heavy load level, fine grain services are suitable [8].

(e) Dynamic replica activation: Depending on varying load levels, e.g. in case of increased load condition, additional replicas may be added to the system and vice versa. Load balancing service must be able to support dynamic creation and termination of such replicas. This will provide more flexible load balancing.

IV. LOAD BALANCING POLICIES FOR SERVER CLUSTER

Primary objective of most of the existing research is to find ways of improving performance by minimizing request execution time, minimizing communication and other overheads and/or maximizing resource utilization in conjunction with fairness in job execution. I/O scheduling is also an important criteria in measuring performance. With the improvement in processing speed and main memory size, I/O subsystem impose a significant bottleneck that prevents applications from achieving maximum system utilization. Problems in implementing I/O based load balancing algorithms in a server cluster are that they require mechanism to collect and analyze the data thereby incurring in potentially expensive overheads and large amount of state information [16].

Scheduling algorithms have substantial impact on performance of the system. The complexity of workload characteristics requires robust load balancing policies. The client requests rates fluctuate dramatically even within short periods of times due to wide disparity in processor and I/O resource requirements of requests. Adapting a load balancing policy to schedule workload without human intervention is critical for swift operation of the cluster of servers. Workload on a server is determined by the amount of time needed to execute all the requests received from the clients in the system. But ideally, the workload cannot be accurately measured before the requests are actually processed. Therefore, it is necessary to use techniques to measure the load by using other parameters like the queue length and utilization of the processor [25].

The following scheduling policies are considered for distributing client requests among servers [28]:

A. Random

In random allocation policy, the incoming requests are forwarded to a randomly selected server. Each of the servers has equal probability of getting the request. The algorithm may result in poor performance. *Random* method can also be extended to solve the heterogeneity issue servers [19,20].

B. Round Robin

This algorithm rotates through a list of servers. Address of any one of the servers can be mapped to a client request. All the servers are treated equally regardless of the number of connections to the server or its response time. Advantages of round robin algorithm are that it is simple, cheap and predictable. Although this algorithm gives better results, it may not be sufficient for heterogeneous group of servers, as this method does not take into account the servers capability. The algorithm has no knowledge of current status of the server workload, software or applications. Also, it does not have information about availability of the servers. It is assumed that the incoming client requests do not have any affinity to a specific server. Figure 2 shows the order of execution or requests in round robin method.

C. Weighted Round Robin

This algorithm tries to eliminate the deficiencies of simple round robin method by pre-assigning static weights to each server. This is done by assigning each server numerical weights between 1 and 10. Capacity of a server can be considered as a static parameter. A server will be assigned load in proportion to its weight. To use weight-based algorithm, relative weights are assigned carefully to each server instance. Weights may be determined on the basis of server configuration, for example, processing capacity of the server's hardware in relation to other servers. If the weight of a server is changed and it is rebooted, new information is propagated throughout the cluster.

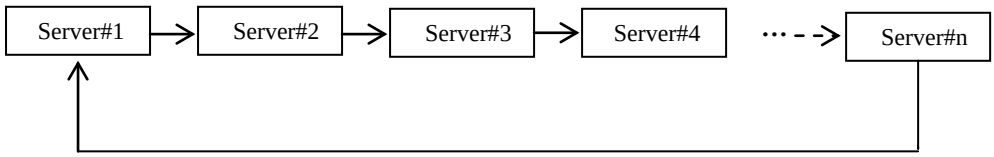


Fig. 2 Round robin scheduling for web servers

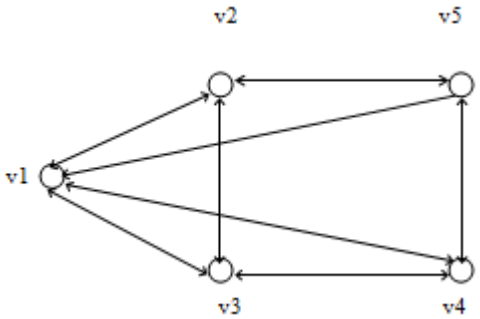
For example, if test results indicate that Server#1 can process 100 requests per second, Serve#2 as well as Server#3 can process 200 requests per second, Server#4 can process 300 requests per second and Server#5 can process 500 requests per second than the weights should be 1 2 2 3 5 for servers Server#1, Server#2, Server#3, Server#4 and Server#5 respectively. This means that out of 13 requests, Server#1 will get 1 request, Server#2 will each get 2 requests, Server#3 will get 2 requests, Server#4 will get 3 requests and Server#5 will get 5 requests. However, just like round robin technique, weighted round robin algorithm does not consider the processing time of clients' individual request. In the situations where some of the requests take longer time, advance load balancing algorithms are required. A variation of round robin technique is **dynamic weighted round robin**, which dynamically evaluates weights based on the load state of the server. These weights are changed periodically. However, in all other policies, requests' allocation algorithms have no knowledge of the system's current state.

D. Shortest Queue

At each server's processor, a queue of incoming request is maintained. In a simple case, the server with minimum number of requests at its processor queue is assigned the new request. But if the requests have too much variation in their processing time, then simply measuring queue length is not sufficient. In such situations, we have to approximate the processing time requirement of each request and the load on the processor is the summation of processing time requirements of the requests in the queue. However, this technique has theoretical significance only as it is not possible to determine exact execution time requirement before actually running the process. We may only find estimate of execution time using statistical techniques like exponential smoothening or identify long processes which have already used execution time more than the average execution time of the processes. Estimates can also be developed by benchmarking of server performance based on real time statistics to determine load level of the server. However such estimates must be constantly updated over time.

E. Diffusive Load Balancing

The network of servers is stored in the form of graph G <V, E>. Here, V is the number of server nodes and E represents communication links between nodes as shown in Fig 3 (a). Figure 3 (b) shows the representation of this graph by means of adjacency*matrix.*



(a) Network topology for diffusive load balancing

$$G = \begin{bmatrix} & v1 & v2 & v3 & v4 & v5 \\ v1 & 0 & 1 & 1 & 1 & 0 \\ v2 & 1 & 0 & 1 & 1 & 1 \\ v3 & 1 & 1 & 0 & 1 & 0 \\ v4 & 1 & 0 & 1 & 0 & 1 \\ v5 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

(b) Representation of the network by graph G <V, E>

Fig. 3 Graph representation on a network of servers

A request assigned at the server is forwarded to another server, if communication link exists between any two servers.The client request is received by the router, which, in turn, forwards request to one of the servers. The search for granting server causes traversal of the network along directed edges in diffusive fashion i.e. edges leading to less loaded servers. Request is moved from a server to its neighbouring server provided the difference of load between the server and its neighbour is above a threshold value. The workload of the server is measured using the length of processor's ready queue. The search finishes when the granting server is found. Performance indicators of load balancing are response time (time which is defined as the difference between finish time of execution of a request and the time when client submits that request), active connection count, server agent response, bandwidth consumption etc [9,27].

A centralized load balancer performs load balancing request distribution by selecting appropriate server. The performance of load balancing algorithm is measured on the basis of response time achieved by using a given algorithm.

The following steps are involved in the algorithm:

(i) Accept the new client request: The request is submitted to admission control mechanism which determines whether there is sufficient capacity to service the request. If sufficient capacity is not available, the request is rejected. Otherwise the request is forwarded to the load balancer.

(ii) Collect the state information: The load balancer collects the status of the servers to find the load information and performance weights etc. depending on the algorithm used for load balancing.

(iii) Server selection: Select the server which is going to process the request.

(iv) Request distribution: Forward the request to the selected server. In case of stateful servers, the load balancer transfers the state of the client from previous server to the selected server.

V. SIMULATION AND RESULT DISCUSSION

Software simulator was designed and implemented to evaluate DLB in web servers using artificial workload. We assumed random process arrival and random service time distribution. Virtual servers are used to process the workloads. We consider close queuing network model of a DCS with n homogeneous servers interconnected by high-speed network with negligible communication delays. The system was examined with n=5.

TABLE I COMPUTATION OF UTILIZATION IF THE SERV ERS USING DIFFERENT LOAD BALANCING TECHNIQUES

Utilization of Servers				
Server id	**Random**	**Round Robin**	**Shortest Queue**	DIFFUSIVE
1	46	83	77	79
2	89	81	80	73
3	98	61	79	81
4	79	82	76	69
5	81	38	70	76

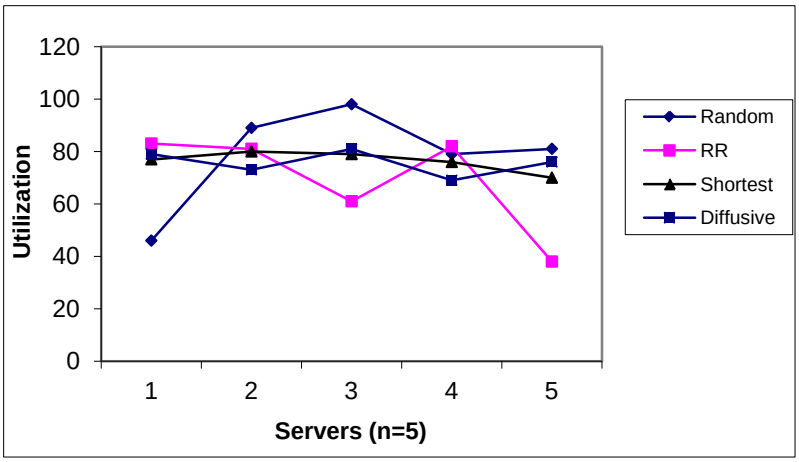


Fig. 4 Comparison of utilization of servers using different load balancing techniques.

The results of comparison of server load balancing techniques are shown in Table I, and Figure 4 which show the comparison of server load balancing techniques. For each algorithm, mean response time and utilization of processor was computed. Load balancing techniques gives much better results than assigning requests to the servers randomly. Round robin method achieves moderate results compared to random load balancing. As expected, the shortest queue algorithm gives best results but as it is not possible to know in advance the processing time for a client's request, this technique has only theoretical significance. However this technique works as a benchmark to compare other implementable techniques. The results also reveal that diffusive load balancing yield better result than round robin technique.

VI. CONCLUSION

In this paper, we have investigated the problem of server load balancing and evaluated various server load balancing policies. From the result of simulation studies we can say that load balancing improves the performance of the server cluster by proper resource utilization and reducing the mean response time by distributing the workload evenly among the servers in the cluster.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil.
[2] Abdelzaher, T.F., Shin, K.G. and Bhatti, N., "Performance Guarantee for Web Server End Systems: A Control Theoretical Approach," IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 1, Jan. 2000, pp. 80-96.
[3] Andreolini, M. Colajanni, M. and Morselli, R., "Performance Study of Dispatching Algorithms in Multi-tier Web Architectures," Performance Evaluation Review, Vol. 30, No. 22, Sept. 2002, pp.10-20.
[4] Aweya, J. et al., "An Adaptive Load Balancing Scheme for Web Servers," International Journal of Network Management, Vol. 12, No.1, Jan-Feb 2002, pp. 3-39.
[5] Cardellini V. et al., "The State of Art Locally Distributed Web-Server Systems," ACM Computing surveys, Vol. 34, No.2, 2002, pp. 264-311.
[6] Castro, M. Dwyer M., Rumsewicz, M., "Load Balancing and Control for Distributed World Wide Web Servers," Proceedings of IEEE International Conference on Control Applications, Hawaii, USA, 22-27 Aug. 1999, pp.1614-1618.
[7] Ciardo, G., Riksha, A. and Smimi, E., "EQUILOAD: A Load Balancing Policy for Cluster Web Servers," Performance Evaluation, Vol. 46, No. 2-3, 2001, pp. 101-124.
[8] Chen, X., Chen, H. and Mohapatra, P., "An Admission Control Scheme for Predictable Server Response Time for Web Accesses," Proceedings of the 10th World Wide Web Conference, Hong Kong, May 2001, pp. 545-554.
[9] Elsasser, R., Monien, C.B. and Preis, R., "Diffusive Schemes for Load Balancing on Heterogeneous Networks," Theory of Computing System, Vol. 35, 2002, pp. 305-320.
[10] Feldmann, A., Rexford, J. and Caceres, R., "Efficient Policies for Carrying Web Traffic Over Flow Switched Networks," IEEE/ACM Transactions on Networks, Vol. 6, No. 6, Dec. 1998, pp. 673-685.
[11] Fu, B. and Tari, Z. A, "Dynamic Load Distribution Strategy for Systems Under High Task Variation and Heavy Traffic," Proceedings of the ACM Symposium on Applied Computing, Melbourne, Florida, pp. 1031-1037.
[12] Gadde, S., Chase, J. and Rabinovich, M., "Web Caching and Content Distribution: A View from the Interior," Computer Communication, Vol. 24, No. 1-2, Jan. 2001, pp. 222-231.
[13] Garcia, D. and Garcia, J., "TPC-W e-Commerce Benchmark Evaluation," IEEE Computer, Vol. 36, No. 2, Feb. 2003, pp. 42-48.
[14] Ghini, V., Panzieri, F. and Roccetti, M., "Client Centered Load Distributions: A Mechanism for Constructing Responsive Web Services," Proceedings of the 34th Hawai International Conference on System Sciences (HICSS-34), Hawai, U.S.A, Jan.2001, Page 9020.
[15] Iyer, R., Tewari and Kant K., "Overload Control Mechanisms for Web Servers," Workshop on Performance and QoS of Next Generation Networks, Nagoya, Japan, Nov. 2000.
[16] Kartza, H.D., "A Comparative Analysis of Scheduling Policies in Distributed Systems Using Simulation," International Journal of Simulation, Vol. 1, No. 1-2, pp. 12-20.
[17] Lindermeier, M., "Load Management for Distributed Object Oriented Environments," Proceedings of the 2nd International Symposium on Distributed Objects and Applications (DOA 2000), Antwerp, Belgium, Sept. 2000, OMG.
[18] Mehta H., Kanungo P. and Chandwani M., "Performance Enhancement of Scheduling Algorithms in Clusters and Grids using Improved Dynamic Load Balancing Techniques," 20th International World Wide Web Conference 2011 (PhD Symposium), Hosted by

IIIT, Banglore at Hyderabad, 28 March-01 April 2011, pp. 385-389, Awarded NIXI (National Internet Exchange of India) Fellowship.
[19] Mitzenmacher, M., "Analysis of Randomized Load Balancing Schemes," Proceedings of 9th ACM Symposium on Parallel Algorithms and Architectures (SPAA'97), Newport, RI, June 1997, pp. 292-301.
[20] Mitzenmacher, M. "The Power of Two Choices in Randomized Load Balancing," IEEE Transactions on Parallel and Distributed Systems, Vol. 12, No. 10, 2001, pp. 1094-1104.
[21] Mcwherter, D. et al., "Priority Based Mechanisms for OLTP and Transactional Web Applications," 20th International Conference on Data Engineering (ICDE 2004), Boston, MA, Apr. 2002, pp. 535-546.
[22] Menasce, D.A. et al., "Business Oriented Resource Management Policies for e-Commerce Servers," Performance Evaluation, Vol. 42, No. 1-2, Sept. 2000, pp. 223-239.
[23] Othman, O. Balsubramanyam, J. and Schmidt, D., "The Design and Performance of an Adaptive Middleware Load Balancing and Monitoring Service," Proceedings of Third International Workshop on Self Adaptive Software, U.S.A., Arlington, VA, June2003.
[24] Otham, O., O'Ryan, C. and Schmidt, D., "Strategies for CORBA Middleware Based Load Balancing," IEEE DS Online, Vol. 2, No. 3, Mar. 2001.
[25] Sharma R K, Kanungo P. and Chandwani M., "A Dynamic Load Balancing Method using Workstation Priority," International Journal of Engineering Sc. & Tech, April, 2011, ISSN NO: 0975-5462.
[26] Sinha, P. K., Distributed Operating Systems Concepts Design, Prentice Hall of India, 2001.
[27] Sloklic, M. E., "Simulation of Load Balancing Algorithms: A Comparative Study," SIGCSE Bulletin, Vol. 34, No.4, Dec. 2002, pp. 138-141.
[28] Tiwari A. and Kanungo P., "Dynamic Load Balancing Algorithm for Scalable Heterogeneous Web Server Cluster with Content Awareness," 2nd International Conference on Trendz in Information Sciences & Computing, (TISC) 2010, Satyabhama University, Chennai, India, pp. 143-148 (Print ISBN: 978-1-4244-9007-3, Paper available on IEEE Xplore, Digital Object Identifier: 10.1109/TISC.2010.5714626, received best paper award of the session).
[29] Watts, J. and F., Taylor A., "Practical Approach to Dynamic Load Balancing," IEEE Transactions on Parallel and Distributed Systems, Vol. 9, No.3, 1998, pp. 235-248.