Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 10, October 2013, pg.78 – 85

RESEARCH ARTICLE

Scheduling Algorithms in Web Servers Clusters

Priyesh Kanungo
Computer Centre, School of Computer Science and Information Technology
Devi Ahilya University
Indore-452001, India
Email: priyeshkanungo@hotmail.com

Abstract—***Scheduling of jobs in web cluster of servers is a major research activity in Distributed Computing System (DCS). A key issue in server load balancing in a DCS is to select an effective load balancing scheme to distribute clients' requests to the servers. This paper considers weighted round robin, shortest queue and diffusive load balancing policies. Performance of each of these policies was analyzed and compared. In this paper, we have investigated the problem of server load balancing and evaluated various server load balancing policies. The objective is to identify the techniques that produce good overall system performance.***

Keywords— ***Server Cluster, Scheduler; Load Balancing; Distributed Systems; Round Robin Algorithm; Diffusive Policy***

I. INTRODUCTION

In a client server environment, it is common to have a cluster of replicated servers which accepts requests from the large number of clients. A cluster is a group of servers with identical contents, networked together to act as a single virtual server and capable of growing with the corporate needs. Clustering enables a transparent growth as physical servers can be added without externally visible network changes. Clustering also improves fault tolerance so that a physical server can be taken down for maintenance or repair without network shutdown. A cluster server exhibits high availability and throughput characteristics which are much better than a costly, largest single server [1,5]. An example of web cluster is shown in Fig. 1. Cluster of servers may have heterogeneous servers. Their configuration and load level may also change dynamically. Clusters may also be integrated into a computational grid.

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, 10]. To illustrate the process of DLB of server cluster, we describe the process formally as well as informally in following sections.

II. Scheduling Policies and Methodology

The following scheduling policies are considered for distributing client requests among servers [6, 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.

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.

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 [12, 13].

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. Estimates can 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

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 [8, 16].

III. Informal Description Of The Algorithm

In this algorithm, we assume that:

(a) The scheduler has perfect information while making scheduling decisions.
(b) The scheduling overheads are negligible.
(c) The requests are highly independent and they can be executed at any time and in any order.
(d) A closed queuing network model is considered.

There are n independent processors, each serving its queue and interconnected by high-speed network with negligible communication delay. We examine the system for n=5 processors which is reasonable for medium scale departmental network. The workload is shared among the replicated servers. The arrived requests are scheduled on the servers. In a server queue, requests are executed using round robin method[11, .

Almost all the load balancing schemes use some load indices to measure the server load levels. Prior studies have shown that resource queue lengths are good indicator of load levels [7, 9]. We use sum of execution times of active server accesses as the server load index in shortest queue policy [3,4].
Server on which a request will be executed is decided by a particular algorithm as follows[15]:

(a) In random policy, a server will be selected randomly with each server having equal probability.
(b) In round robin policy, a list of servers is maintained and requests are assigned to the servers in the circular fashion.
(c) In weighted round robin policy, each server is assigned number of requests in proportion to the weight of the server.
(d) In shortest queue policy, a server having minimum number of requests in the queue, will be forwarded a request.
(e) In diffusive algorithm, a request assigned at a server is forwarded to another adjacent server if communication linked exists between the two servers and the new server has lesser load.

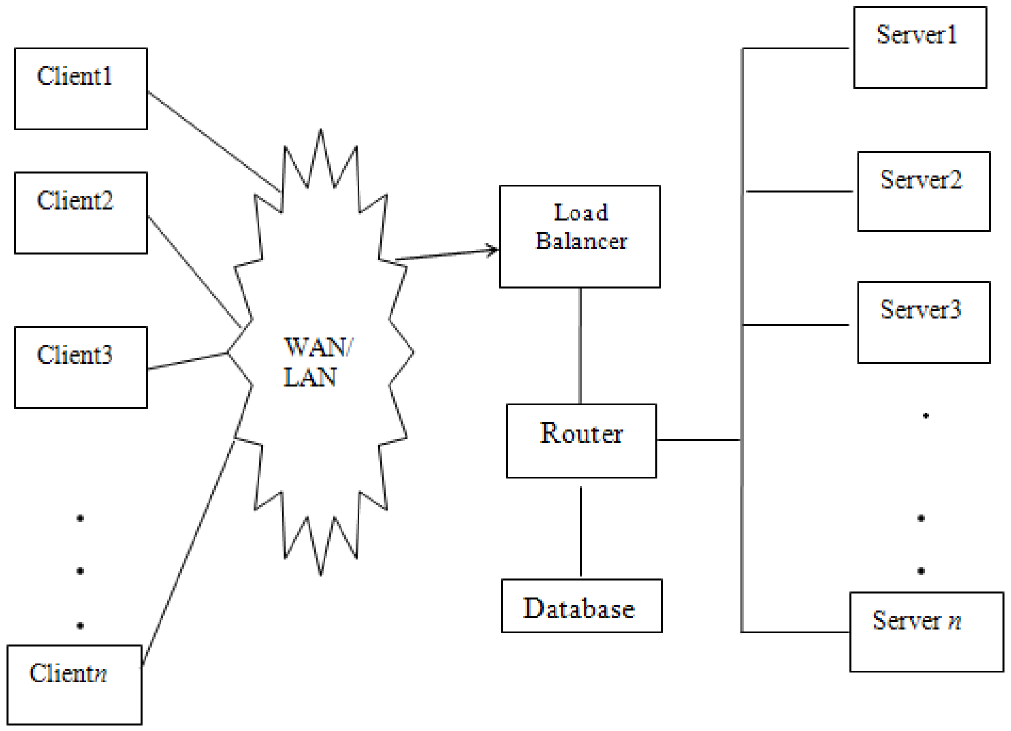

Fig. 1 Load balancing in a server cluster

We compute the load on a server as:

$$W_i = \sum_{j=1}^{p_i} t_j \qquad (1)$$

where,

t_j is the service time of the request j
p_i is the number of processes on node i
W_i is the workload on server i

The status of each server is computed upon arrival of a new request.

The response ratio R of a process is computed as:

$$R = t / (t + w) \qquad 0 < R < = 1 \qquad (2)$$

where,

t is the service time of the request
w is the waiting time or missed time

We have used two load indices, queue length on the server and utilization of the processor.
Processor utilization is computed as:

$$U_{mean} = \left(\sum_{i=1}^{n} U_i\right) / n \qquad (3)$$

where,

U_{mean}	is the mean utilization.
U_i	is the utilization of server i
n	is the number of servers

Mean response time:

$$R_{ean} = \left(\sum_{i=1}^{n} R_i\right) / n \qquad (4)$$

where,

R_{mean}	is the mean response time
R_i	is the response time of server i

Standard deviation of the response time is:

$$\sigma(R_i) = \text{sqrt}\left(\sum (R_i - R_{mean})^2\right) / n \qquad (5)$$

Load balancer collects load index information from each server so that the systems load distribution l is

$$l = (l_i \mid 0 < = i <= n\} \qquad (6)$$

Load distribution at time t is described by the mean value

$$l_{mean} = \left(\sum_{i=1}^{n} l_i\right) / n \qquad (7)$$

and the variance:

$$\sigma(L_i) = \left(\sum_{i=1}^{n} (l_i - l_{mean})\right) / n \qquad (8)$$

This load information is collected from all servers periodically. Load balancing on each server is formalized by random arrival time and random service time. We can also measure the inaccuracy in load measurement. Load inaccuracy for certain delay Δt is defined as the statistical mean of difference in queue lengths measured at arbitrary time t and t+Δt. When the server is moderately busy, say 50%, the load inaccuracy is only moderate even with high delay. But when server is too busy, say 90%, the load index accuracy is much more. Therefore, at higher load levels, information dissemination delays should be small otherwise the results will have higher magnitude of errors [18].

Communication overheads may also be computed as:

$$C_o = t_c / mst \qquad (9)$$

where,

C_o	is the communication overhead
t_c	is the sum of time to send load from node i to the supervisor and time to receive message from the supervisor
mst	is the mean service time

Value of t_c *can be computed as*:

$t_c = t_s + t_r$ (10)

where,

t_s is the sending time

t_r is the receiving time

Migration overhead may be computed as:

$m_o = t_m / mst$ (11)

t_m is time to migrate a request from source to destination and includes time for queue manipulation, load table operations etc.

Equation (1) to Equation (11) constitute steps in the formal algorithm. 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.

IV. FORMAL DESCRIPTION OF THE ALGORITHM

The algorithm for server load balancing is formally described as under:

```
Algorithm Server-Load-Balancing
/*Algorithm for load balancing a server cluster. Following techniques have been used: 1=Random,
2=Round Robin, 3=Shortest Queue, 4=Diffusive*/
{
for each server in the cluster store following data
        server-queue, number-of-processes, server-load, mean-response-     ratio, server- utilization
for each request store
        pid, ser-time, arr-time, dep-time, response-time
/* new requests arrive at load balancer randomly with random service time requirements*/
CreateLoadBalancerQueue(struct processes())
RandomAlloction(queue lbq, queue server())
        {
        for each process in the load balancer queue
                assign a request $p_i$ from lbq to server $S_{j;}$
                increment $i$ and $j$;
                if the serverlist is finished assign $i$=1;
       }
RoundRobinAllotment(queue lbq, queue server());
      {
       for each process in the load balancer queue
                assign a request $p_i$ from lbq to server $S_j$;
                increment $i$ and $j$;
                if the serverlist is finished
                      assign $i$=1;
       }
ShortestQueueAllocation(queue lbq, queue server())
       { int i, sid=0, bt,j,n;
       for each process in the load balancer queue
                select a server $S_i$ with minimum load;
                assign request $p_i$ to $S_i$
       }
DiffusiveAllocation(queue lbq, queue server())
      {
       ComputeThreshold( );
                /*compute propagation threshold of the system */
       for each server in the serverlist
                    assign request $p_i$ from lbq to $S_j$;
                do while granting server is not found
                    if ($server\text{-}load_i$ - $server\text{-}load_{i+1)}$ > threshold)
                          server=s(i)
      }
```

```
}
```

End of Algorithm

V. SIMULATION AND RESULT DISCUSSION

Equation (1) to Equation (11) constitute steps in the formal algorithm. Software simulator was designed and implemented to evaluate DLB in web servers. The simulator was driven using artificial workload instead of real workload. Artificial workloads have a greater flexibility as compared to real workloads and are easier to reproduce. We assume 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]11, 14]. The system was examined with *n*=5.

The results of comparison of server load balancing techniques are shown in Table I, Table II, Table III, Fig.2, Fig. 3. and Fig. 4. Table II Table III, Fig. 2 and Fig. 4 show the comparison of round robin and weighted round robin 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. Weighted round robin technique yields better results than round robin in an environment with different server capabilities. 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.

TABLE I
COMPUTATION OF MEAN RESPONSE TIME OF THE SERVERS USING DIFFERENT LOAD BALANCING TECHNIQUES

Mean Response Time				
Server id	**Random**	**Round Robin**	**Shortest Queue**	**Diffusive**
1	285.5	252.6	157.20	180.1
2	268.2	126.4	150.67	161.2
3	76.1	101.8	151.25	113.6
4	172.9	200.8	154.98	170.2
5	83.1	93.9	126.9	141.2

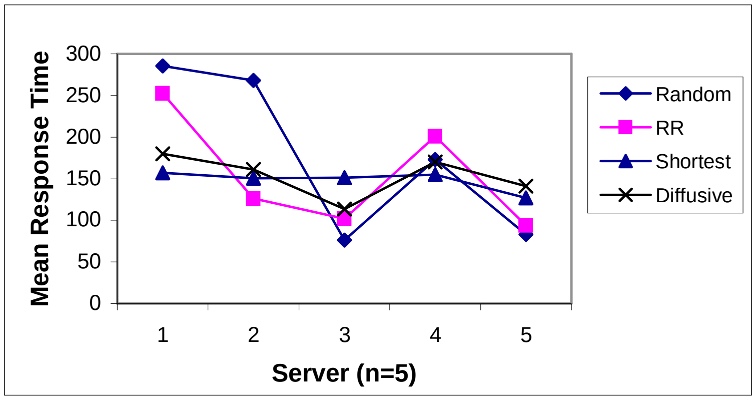

Fig. 2 Comparison of mean response time on each server using different load balancing techniques

TABLE II

COMPUTATION OF MEAN RESPONSE TIME OF THE SERVERS FOR RR AND WRR TECHNIQUES

Mean Response Time			
Server id	**Weight**	**Round Robin**	**Weighted Round Robin**
1	2	278.1	163.3
2	1	279.2	152.6
3	1	277.8	155.9
4	1	291.8	144.7
5	2	266.8	147.1

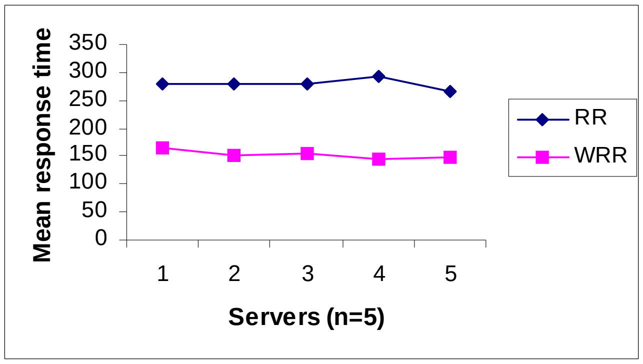

Fig 3 Comparison mean response time of the servers using RR and WRR techniques

TABLE III
COMPUTATION OF UTILIZATION OF THE SERVERS FOR RR AND WRR TECHNIQUES

Utilization of Servers			
Server id	**Weight**	**Round Robin**	**Weighted Round Robin**
1	2	49	98
2	1	98	99
3	1	97	95
4	1	95	92
5	2	46	79

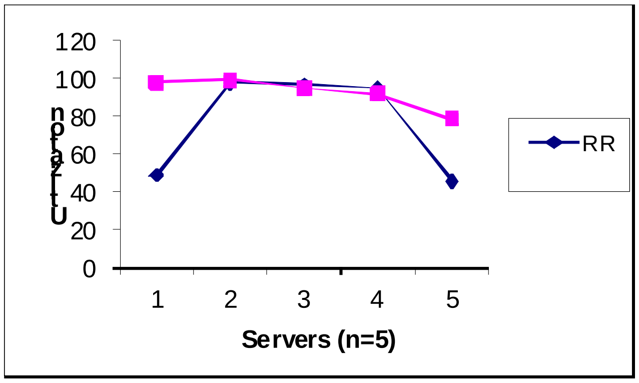

Fig 4 Comparison utilization of the servers using RR and WRR techniques

VI. CONCLUSION

This paper describes various techniques of distributing clients' request among servers in server cluster. On the basis of simulation results, it can be concluded that use of DLB algorithms is necessary to improve the performance of web servers by proper resource utilization and reducing the mean response time by distributing the workload evenly among the servers in the cluster.

REFERENCES

[1] Harchol-Balter, M. et al., "Size Based Scheduling to Improve Web Performance," *ACM Transactions on Computer Systems*, Vol. 21, No.2, May 2003, pp. 207-233.

[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] Cardellini V. et al., "The State of Art Locally Distributed Web-Server Systems," *ACM Computing surveys*, Vol. 34, No.2, 2002, pp. 264-311.

[5] 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.

[6] 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.

[7] Chiang M. L., Wu C. H., Chen Y.J. and Chen N.F., New Content Aware Request Distribution Policies in Web Clusters Providing Multiple Services, ACM Symposiumon Applied Computing, March 8-12, 2009, Honolulu, Hawaii, U.S.A., pp. 79 – 83.

[8] 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.

[9] Ferrari, D. and Zhou, S., "An Empirical Investigation of Load Indices for Load Balancing Applications," *Proceedings of Performance*, North Holland, Netherlands, 1987, pp. 515-528.

[10] 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.

[11] Mehta, H., Kanungo, P**.** and Chandwani, M., "Performance Enhancement in Scheduling Algorithms in Web Server Cluster using Improved Dynamic Load Balancing Policies," *INDIACOM-2008, 2nd National Conference on Computing for National Development*," 8-9 Feb., 2008, Bharti Vidyapeeth, New Delhi.

[12] 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.

[13] 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.

[14] 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.

[15] Sinha, P. K., Distributed Operating Systems Concepts Design, *Prentice Hall of India*, 2001.

[16] Sloklic, M. E., "Simulation of Load Balancing Algorithms: A Comparative Study," *SIGCSE Bulletin*, Vol. 34, No.4, Dec. 2002, pp. 138-141.

[17] 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).

[18] Xu, J. and Hwang, K., "Heuristic Methods for Dynamic Load Balancing in a Message Passing Multicomputer," *Journal of Parallel and Distributed Computing*, Vol. 18, No.1, May 1993, pp. 888-897**.**