Load Balancing Policies of Web Servers: Research Analysis, Classification and Perspectives

Prabu U[1,*], Malarvizhi N[2], Amudhavel J[3], Sriram R[4] and Ravisasthiri P[5]

[1]Department of CSE, Koneru Lakshmaiah Education Foundation, Andhra Pradesh, India, **email:** uprabu28@gmail.com
[2]Department of CSE, IFET College of Engineering, Villupuram, Tamil Nadu, India, **email:** nmalarvizhi16@gmail.com
[3]School of Computer Science and Engineering, VIT Bhopal University, M.P, India, **email:** info.amudhavel@gmail.com
[4]Department of CSE, Rajiv Gandhi College of Engineering and Technology, Pondicherry, **email:** mail4ramz@gmail.com
[5]Department of IT, RAAK College of Engineering and Technology, Pondicherry, **email:** ravisasthiri.p@gmail.com

Abstract

The usage of the internet has increased rapidly from the past few years. Thus it automatically increases the internet traffic. In the internet, a web server always responds to the client or web browser's requests. The major feature of a web server is to be available, scalable and predictable. So the core part to be concentrated is web server load balancing. Load balancing plays a vital role in allocating the jobs to the web server based on its status. There are various policies available for web server load balancing. Each of these policies came into existence based on the certain needs. In this paper, we have examined the latest policies of web server load balancing.

Keywords: Web server, Load balancing, Load balancing policies.

Received on 31 January 2019, accepted on 14 May 2019, published on 10 June 2019

doi: 10.4108/eai.10-6-2019.159098

[*]Corresponding author. Email:uprabu28@gmail.com

1. Introduction

A web server is a computer which responds to the clients or web browsers. The response is in term of the web pages. A computer can become a web server when software is installed to it and connected to the internet. In today's world many software applications available for web servers. Web server load balancing plays a vital role in web server farms.

Load balancing is done in two versions namely static and dynamic. Azar et al., was the first to analyze the static version of load balancing using balls-and-bins model[1]. Vvedenskaya et al., dynamic version of load balancing using the dynamic super market model. The dynamic supermarket model is analyzed beneath under the SQ(d) policy[2]. To dispatch the jobs uniformly among servers application delivery controller is used [3].

The major reason for the load balancing of the servers is to provide scalability, high availability and predictability. Scalability is defined as the capability of adapting load dynamically as it increases. This should not impact the performance. High availability is defined as the capability of being accessible and available even if one or more system fails. Predictability is defined as the capability of assuring and managing that the services are delivered by considering the performance, availability etc. [4-6]. DNS round-robin is used before the availability of load balancing devices. But the disadvantage of DNS round-robin is that it is not aware whether the server is working or not. So the needs of the load balancing device arise. Then the application delivery controller came into existence. The jobs are dispatched to the server using application delivery controller.

Message passing paradigm is used to implement the traditional load balancing policies of web servers. According to W. Winston[7] by allocating the workload evenly between servers optimal response time can be achieved.

Figure 1. demonstrates the load balancing with distributed dispatchers. Using the Equal-Cost-Multi-Path(ECMP) algorithm the requests are routed to a random dispatcher in a router. The service time of the requests gets changed based on the amount of the data in the requests.

Some of the scheduling policies for web server clusters are Random method, Round Robin algorithm, Weighted Round Robin algorithm, Shortest Queue algorithm, Diffusive Load Balancing.

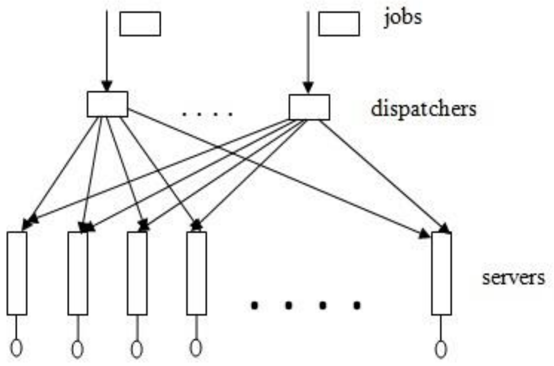

Figure 1. Distributed dispatchers for a cluster of parallel servers

1.1. Motivation

Web servers are playing a major role in the today's world. To concentrate on the load balancing of the web server has become the major thrust area. There are several policies available for web server load balancing. Each of these policies is framed by taking the servers and jobs into consideration. Some policies may take only the servers into consideration and perform load balancing process while others may take jobs. So by extracting these facets we can have an insight into each of the policies and their considerations as well. This in turn will help us to grasp and gather the actual things going on in the web server farms for balancing the load.

1.2. Contribution

A research analysis on load balancing policies of web servers is presented in this paper. Web servers are having a major role in today's internet world. Without these web servers communication among users through the internet cannot be imagined. The policies of the web server load balancing are discussed in depth. To the best of our understanding, we admit this is the first research analysis on the load balancing policies of the web servers.

1.3. Organization

The rest of this paper is organized as follows. Section 2 presents Load balancing policies of web server with each of its advantages and disadvantages. Section 3 has the discussion part the paper. Section 4 concludes the paper.

2. Load Balancing Policies of Web Server

2.1. Cooperative Adaptive Symmetrical Initiated Diffusion

SafiriyuEludiora et al.,[8] proposed a policy for load balancing. It deals with the unregulated jobs or tasks relocation among the servers. In the distributed environment often there is wastage of bandwidth when there is an unnecessary migration of the jobs.

The major difficulty faced by the load balancing policies so far is to obtain the exact server for exchange of jobs. The accomplishment of the load balancing policy is based fully on the job relocation and bandwidth minimization. Bandwidth minimization is not considered as a major issue in the prior works. The job relocation should be managed when load balancing policies are deployed. In the Cooperative Adaptive Symmetrical Initiated Diffusion (CASID) policy, both the sender and receivers are servers such as web servers and database servers. Once the communication is established between servers, job relocation begins. The sending and receiving servers are identified by the load threshold. The server agents gather the server status. It consists of methods such as IDLEServerBehaviour, BUSYServerBehaviour, registerDFServices, MigrateJobCompleted, jobLoader, jobLoaderFinishedMigrate, ServerIsIDLE. Job relocation is carried out by the mobile agents. It consists of methods such as ServerInteractionBehaviour, registerDFServices. CASID policy aims to enhance the system throughput and reduce the response time. The load threshold of the server is calculated through the queue length. The servers know each other's status as it broadcasts each other's status. The beneficial job relocations are only allowed in this policy. Thus this policy tries to minimize the bandwidth utilization. The experimental results show that CASID policy performs better than Platform for Load Balancing (PLB).

2.2. Join-Idle-Queue

Yi Lu et al.,[9] proposed an algorithm for distributed load balancing. This distributed load balancing takes place in large systems. In this algorithm when the job arrives, there is no overhead between the processors and dispatchers while they communicate.

The JIQ algorithm is made up of two load balancing systems namely primary and secondary. The communication occurs between these two through a queue called I-queue. An I-queue consists of idle processors. The dispatchers can access the processors.

2.2.1 Primary Load Balancing

The primary load balancing system utilizes the details of the idle servers which are in the I-queue. As soon as the job arrives the dispatcher checks with the I-queue. If the I-queue consists of idle processors, the dispatcher picks the earliest idle processor and assigns the job to it. If the I-queue doesn't consist of idle processors, the dispatcher assigns the job to any random server. As the processor finishes the job assigned to it, it joins the I-queue. There is a challenge that the job arrives the I-queue which doesn't consists of idle processors but it is not aware of idle processors in some other I-queue.

2.2.2 Secondary Load Balancing

Based on the load balancing algorithm as the processor becomes idle, it selects any one of the I-queue and joins it. In the secondary load balancing, two algorithms in the reverse directions are considered namely JIQ-Random and JIQ-SQ(d). JIQ-Random, at random chooses the I-queue consistently. JIQ-SQ(d), d random I-queues are chosen by an idle processor. An idle processor then joins the queue which is of smallest length.

For dynamically scalable web server farms JIQ algorithm is proposed. JIQ algorithm performs well than SQ(d) algorithm[10-14]. The performance is observed in terms of response time. On the critical path there is also no communication overhead. The complexity of JIQ is also less than SQ(d). At high load JIQ confirms that it will be useful.

2.3. Join-The-Shortest Queue

SatheeshAbimannan et al.,[15] proposed an algorithm to calculate the possible worst case when the job arrives and joins the shortest queue. The modified power-series algorithm is used to calculate the fixed queue length.

The servers are considered as homogeneous in the web server farms. Thus all the servers have the same service rate. In web server farms, the worst-case of JSQ routing policy is analyzed. When a new job arrives and joins the shortest queue it is supposed to be worst case because the job would have started earlier if it joins other queue. When the system load is extremely close to 1 in homogeneous system and if there are more than three servers, then the possibility of worst case surpasses 12.

2.4. Simulated Annealing Load Spreading Algorithm

Bas Boone et al.,[16] proposed an algorithm for autonomous service brokering. The algorithm is named as Simulated Annealing Load Spreading Algorithm (SALSA) and it is applied in QoS-aware load balancing. To decrease the web server's load it balances the requests and also sometimes drops the requests of the default users selectively. By doing this, premium customers are guaranteed with Service Level Agreement(SLA) and default customers are provided with the best effort. The intent of SALSA is to ensure that server is not overloaded, to reduce the average waiting time of all clients and to reduce the fraction of dropped requests.

Two types of requests are taken into consideration for modeling the dissimilar user profiles. They are premium requests and default requests. Premium requests should not be dropped. Premium clients entail a SLA guaranteeing. In a SLA guaranteeing the overall waiting time for a request should be limited to a threshold. Default clients are served with the best effort and don’t need arithmetical guarantees. Default requests can be dropped to make sure that premium requests are served and limited to a threshold waiting time. As a result, for the acceptable fraction a compromise should be done between default client request drop and beyond threshold.

The correctness and performance of the SALSA is evaluated thrice. The first evaluation is done with different number of server setups for comparing it's applicability with other load balancing algorithms. In the second evaluation mechanism correctness is evaluated in a extremely controlled simulation environment. The last evaluation is done through an experimental setup. The web service broker has to undergo a stress test. The experimental setup is done to calculate the differences among SALSA, priority queue and round robin algorithms.

2.5. An Architecture for Scalable network file systems

Hsien-Tsung Chang et al.,[17] proposed an architecture for scalable network file systems. The architecture is load balanced and fault tolerant. It also explains about the free space and internal fragmentation problem which are not concentrated in the Google File System. The scalable network file system has three design issues and it is presented here. In small files to reduce the internal fragmentation a variable number of objects are used. In the bucket servers to balance the load free space and load balancing mechanism is used. To decrease the disk I/O a mechanism is proposed which caches the accessed data frequently. These mechanisms can efficiently progress the performance of scalable network file system. The proposed architecture consists of control center, bucket servers and NFS(Network File System). The file structures and directory is maintained by the control center in the NFS. The file structures are accumulated in the main memory. There are many buckets in the file and each file has a bucketID which is issued by the control center. The bucket servers store the data of the bucket.

2.5.1 Control Center

Each bucketID's status is monitored by the control center. When the load imbalances the control center starts load balancing. Free storage space of each bucket server is checked by the control center. From the available bucket server the control center will duplicate the data if it is less than three. The responsibility is taken by the standby control center when the actual control center is crashed.

Table 1. Recent Web Server Load Balancing Policies

Sl. No	Author Names	Paper Title	Theme of the Paper	Existing Policy	Policy / Algorithm Proposed	Advantages	Disadvantages
1.	SafiriyuEludiora et al., (2010)	A Load Balancing Policy for Distributed Web Service	This policy deals with the unregulated jobs or tasks relocation among the servers. The beneficial job relocations are only allowed in this policy. This policy tries to minimize the bandwidth utilization.	Distributed Dynamic Load Balancing Policy(DDLB), Platform for Load Balancing(PLB)	Cooperative Adaptive Sym metrical Initiated Diffusion (CASID)	The experimental results show that CASID policy performs better than Platform for Load Balancing(PLB).	This policy considers load balancing within DNS.
2.	Yi Lu et al., (2011)	Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services	JIQ algorithm is proposed For dynamically scalable web server farms. The communication occurs between processor and dispatcher through a queue called I-queue. An I-queue consists of idle processors.	The Power-of -d (SQ(d)) algorithm, Work stealing and work sharing	Join-Idle-Queue	JIQ algorithm performs well than SQ(d) algorithm. The complexity of JIQ is also less than SQ(d).	Load Balancing is done based on only the idleness but job size is not considered.
3.	SatheeshAbimannan et al., (2010)	Join-The-Shortest Queue Policy In Web Server Farms	This algorithm calculates the possible worst case when the job arrives and joins the shortest queue.	A numerically stable algorithm for two server queue models	Join-The-Shortest Queue Policy	Job is assigned to the queue with the smallest number of jobs. No manipulation between queues is permitted.	This policy focuses only on the homogeneous systems.
4.	Bas Boone et al., (2010)	SALSA: QoS-aware load balancing for autonomous service brokering	It is an algorithm for autonomous service brokering. It is applied in QoS-aware load balancing.	Round-trip load balancing algorithm, Weighted round-robin	Simulated Annealing Load Spreading Algorithm(S ALSA)	SALSA performs better than priority queue and round robin algorithms.	Two types of requests namely premium requests and default requests are taken into consideration and load is balanced.
5.	Hsien-Tsung Chang et al., (2014)	Scalable network file systems with load balancing and fault tolerance for web services	An architecture which is load balanced and fault tolerant is proposed. It explains about the free space and internal fragmentation problem which are not concentrated in the Google File System.	Frangipani's File System, xFS, Minnesota's GFS and GPFS	Scalable network file systems	The results showed that the proposed mechanism improves the performance of NFS for large web services.	The bucket sizes used here are fixed and not variable.
6.	Jianzhe Tai et al., (2014)	Load balancing for cluster systems under heavy-tailed and temporal dependent workloads	The efficiency of ADAPTLOAD and JSQ is inherited by ADUS. According to the current size of the job ADUS tries to divide it.	Random policy, AdaptLoad	ADUS	ADUS attains major performance by balancing the system load and user traffic in a cluster.	The algorithm is not able to self-adjust its parameters to transient workload conditions

2.5.2 Bucket Server

In the bucket server, the version number of each bucket is stored. The data gets synchronized as the bucket server restarts and contacts the control center. The hardware information of the bucket server is registered with the control server when it joins the NFS and a unique bucket serverID is given to it. A new bucket server can be easily inserted into the system when additional storage spaces are required or one bucket server crashes. For a large storage system, the maintenance time is saved by this approach. The bucket server supports functions such as reading, writing and transferring a bucket. To improve the overall performance the bucket server caches the hot spot.

2.5.3 Network File System

To provide a standard file access the NFS is combined with the supported API. This is like a middleware. Since all the data transmissions happens via NFS, NFS itself becomes bottleneck.

2.5.4 Experiment

NFS has been setup with sixteen bucket servers and one center server. Each server runs on Intel Q6600 as CPU, FreeBSD 6.4 as operating system and 1TB hard disk. The results showed that the proposed mechanism improves the performance of NFS for large web services.

2.6. ADUS

Jianzhe Tai et al., proposed ADUS[18] a novel load balancing policy. The efficiency of ADAPTLOAD[19] and JSQ[20] is inherited by ADUS. According to the current size of the job ADUS tries to divide it. Based on the load, servers are ranked accordingly. ADUS attains major performance by balancing the system load and user traffic in a cluster. This is achieved by dispatching same size jobs to the servers based on their ranks. The small jobs are always given to less loaded servers.

2.6.1 ADUS Algorithm

```
1. Initializes the priority list and job size boundaries.
2. When every job completes,
   2.1 All the servers are sorted in the non-decreasing order of system load and priority list is updated
   2.2 Size boundaries are updated such that work is equally divided into N areas.
3. For each arriving job,
   3.1 Based on the job size boundary direct it to server based on the priority server.
```

The algorithm is planned to get refined in future where the window size can self-adjust. It is also planned to apply this policy to real-time systems such as data centers and clouds. In large-scaled cluster environments for resource allocation, it is expected that ADUS will be an efficient approach.

3. Discussion

From the Table 1, We can deduce that every policy varies based on the criteria taken into consideration. SafiriyuEludiora et al.,[5] takes job relocation into consideration and load balancing occurs inside the DNS. Yi Lu et al.,[6] considers I-queue and load balancing is done based on the idleness of the server but the job size is not taken into account. SatheeshAbimannan et al.,[12] focuses only on the worst case analysis of job and his policy considers only the homogeneous systems. Bas Boone et al.,[13] considers two types of requests for load balancing but performs better than priority queue and round robin algorithms. Hsien-Tsung Chang et al.,[14] presents load balancing of the file system like similar in google file system. Here the network file system is considered. Jianzhe Tai et al.,[15] considers the current size of the job in the clusters but the algorithm is not able to adjust its parameter of temporary workload conditions. So here we analyze that there is no policy that takes all these things into consideration. But all these policies are resulting good in the experimental evaluations.

4. Conclusion

In this paper, we have discussed about various load balancing policies and its research perspectives. There are lot of policies so far in existence for web server load balancing among which some of them are analyzed. These policies are framed based on the criteria such as jobs, job queues, server idleness and load on the server. But all these criteria were not integrated in a single policy. Each policy concentrates on either any one of these criteria and a few more than that. Additional researches are required on these policies such that each policy balances the load based on all these criteria which in turn provides the proper load balancing across the web server farms. Thus each policy can be extended further which provides optimization when load balancing process takes place.

References

[1] Azar, Y. Broder, A. Karlin, A. Upfal, E. (1994) Balanced allocations. *In SIAM Journal on Computing* 593–602.

[2] N. D. Vvedenskaya, R. L. Dobrushin, and F. I. Karpelevich (1996) ,Queueing system with selection of the shortest of two queues: An asymptotic approach". *Probl. Inf. Transm* **volume:** 32(1) 20–34.

[3] Salchow, KJ (Ken) . Load Balancing 101: The Evolution to Application Delivery Controllers.

[4] Li M., Sun X., Wang H., Zhang Y. (2009), Optimal Privacy-Aware Path in Hippocratic Databases. In: Zhou X., Yokota H., Deng K., Liu Q. (eds) *Database Systems for Advanced Applications*. DASFAA 2009. Lecture Notes in Computer Science, vol 5463. Springer, Berlin, Heidelberg, **volume:** 5463 441-455.

[5] Le Sun,Jiangan Ma,Hua Wang,Yanchun Zhang,Jianming Yong (2018), Cloud Service Description Model: An

Extension of USDL for Cloud Services, *IEEE Transactions on Services Computing*, **volume:** 11(2) 354-368.

[6] Yuanyu Zhang, Yulong Shen, Hua Wang, Yanchun Zhang, Xiaohong Jiang (2018), On Secure Wireless Communications for Service Oriented Computing, *IEEE Transactions on Services Computing*, **volume:** 11(2) 318-328.

[7] Winston, W. (1977), Optimality of the Shortest Line Discipline, *Journal of Applied Probability,* **volume:** 14(1) 181-189.

[8] Safiriyu Eludiora, Olatunde Abiona, Ganiyu Aderounmu, Ayodeji Oluwatope, Clement Onime, Lawrence Kehinde (2010), A Load Balancing Policy for Distributed Web Service , *Int. J. Communications, Network and System Sciences* **volume:** 3 645-654.

[9] Yi Lu, QiaominXie, Gabriel Kliot, Allan Geller, James R.Larus, Albert Greenberg (2011), Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services, *Performance Evaluation, Elsevier.*

[10] Vvedenskaya, N.D. Dobrushin, R.L. Karpelevich, F.I. (1996), Queueing system with selection of the shortest of two queues: an asymptotic approach, *Probl. Inf.Transm.* **volume:** 32 (1) 20–34.

[11] Mitzenmacher, M. (1996), The power of two choices in randomized load balancing, Berkeley.

[12] Bramson, M. Lu, Y. Prabhakar, B. (2010), Randomized load balancing with general service time distributions, *ACM Sigmetrics.*

[13] Graham, C. (2000), Chaoticity on path space for a queueing network with selection of the shortest queue among several, *J. Appl. Probab.,* **volume:** 37 198–211.

[14] Luczak, M. McDiarmid, C. (2006), On the maximum queue length in the supermarket model, *Ann. Probab.,* **volume:** 34 (2) 493–527.

[15] SatheeshAbimannan, Kumar Durai, A.V.Jeyakumar, Krishnaveni.S (2010), Join-The-Shortest Queue Policy In Web Server Farms, *Global Journal of Computer Science and Technology*, **volume:** 10(4).

[16] Bas Boone, Sofie Van Hoecke, Gregory Van Seghbroeck , NielsJoncheere , Viviane Jonckers, Filip De Turck, Chris Develder, Bart Dhoedt (2010), SALSA: QoS-aware load balancing for autonomous service brokering, *The Journal of Systems and Software, Elsevier.*

[17] Hsien-Tsung Chang, Yi-Min Chang, Sheng-Yuan Hsiao (2014), Scalable network file systems with load balancing and fault tolerance for web services, *The Journal of Systems and Software, Elsevier.*

[18] Jianzhe Tai , Zhen Li, Jiahui Chen, Ningfang Mi (2014), Load balancing for cluster systems under heavy-tailed and temporal dependent workloads, *Simulation Modelling Practice and Theory, Elsevier.*

[19] Zhang, Q. Riska, A. Sun, W. Smirni, E. Ciardo, G. (2005), Workload-aware load balancing for clustered web servers, *IEEE Trans. Paral. Distrib. Syst.,* **volume:** 16 219–233.

[20] Nelson, R. Philips, T. (1998), An approximation for the mean response time for shortest queue routing with general interarrival and service times, *Perform. Eval. Elsevier,* **volume:** 17 123–139.