International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 3, June 2019, pp. 2083~2093
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i3.pp2083-2093

AWSQ: an approximated web server queuing algorithm for heterogeneous web server cluster

Kadiyala Ramana[1], M. Ponnavaikko[2]
[1]Department of Computer Science and Engineering, SRMIST, India
[2]Vinayaka Missions University, India

Article Info

Article history:

Received Jun 13, 2018
Revised Nov 29, 2018
Accepted Dec 24, 2018

Keywords:

Approximated load
Load balancing
Response time
Throughput
Web server clutser

ABSTRACT

With the rising popularity of web-based applications, the primary and consistent resource in the infrastructure of World Wide Web are cluster-based web servers. Overtly in dynamic contents and database driven applications, especially at heavy load circumstances, the performance handling of clusters is a solemn task. Without using efficient mechanisms, an overloaded web server cannot provide great performance. In clusters, this overloaded condition can be avoided using load balancing mechanisms by sharing the load among available web servers. The existing load balancing mechanisms which were intended to handle static contents will grieve from substantial performance deprivation under database-driven and dynamic contents. The most serviceable load balancing approaches are Web Server Queuing (WSQ), Server Content based Queue (QSC) and Remaining Capacity (RC) under specific conditions to provide better results. By Considering this, we have proposed an approximated web server Queuing mechanism for web server clusters and also proposed an analytical model for calculating the load of a web server. The requests are classified based on the service time and keep tracking the number of outstanding requests at each webserver to achieve better performance. The approximated load of each web server is used for load balancing. The investigational results illustrate the effectiveness of the proposed mechanism by improving the mean response time, throughput and drop rate of the server cluster.

Corresponding Author:

Kadiyala Ramana,
Department of Computer Science and Engineering,
SRM Institute of Science and Technology,
Potheri, Mahatma Gandhi Rd, SRM Nagar, Kattankulathur, Tamil Nadu 603203, India.
Email: ramana.it01@gmail.com

1. INTRODUCTION

The volume of the information available online and services available for the internet users increased through the blast of the world wide web. The thriving of various service demands and information has made a sensational burden on the World Wide Web (WWW) infrastructure. To serve a large number of client request they need advanced web server systems. Users can expect less response time and low site downtime. To attract new customers and not to lose the current market web service providers must provide their applications with greater performance. Due to scalability, availability and cost-effectiveness of distributed web server cluster architectures, they became more popular instead of using one web server, which has high processing capabilities.

In 1995, the number of internet users was less than 1% in the world population, whereas today it is 40%. In 2016, there were 3.5 billion internet users while in 2005 there were 1.02 billion internet users [1]. With the fast evolution of internet traffic, maximum popular websites need to scale up their server volumes. The popular way to provide a list of alternative, or equivalent mirrored servers at different locations.

The mirrored servers are not transparent to the users and it is hard to provide load balancing and fault-tolerance [1]. The technique which is used to redistribute the workload from loaded servers to idle servers to improve the performance is called Load balancing. Load balancing is one of the crucial issues, which divides the workload dynamically among the servers by improving the performance of the system [2]. The most promising approach to handle popular web sites is to use a distributed architecture which maintains a virtual single interface. A web server cluster is known to be a compilation of servers which works jointly as a solitary articulate system for providing highly & scalable web services. It relies on load balancing techniques where it shares service traffic efficiently between its back-end servers and visibly to the clients. The scalability is termed as the capacity in system measurement where to meet the escalating demands as service traffic. The capacity of the system is determined based on the support of number of parallel connections of servers per second without affecting of momentous queuing delay in the interior infrastructure.

By taking advantage of the server redundancy, load balancing techniques improves the system availability. The ability of a server to provide endless services over time is called Availability and it is deliberated as uptime percentage. When a cluster server declines or abort, the load will routinely redistribute with slight or refusal brunt laying the service among other available services.

The servers in the Web server cluster are not essentially situated in the equivalent site and they will be located in diverse biological locations. In proxy servers they are all located at different locations. Because of the rapid increase of Internet, the broadcast time is an important recital factor in network service.

In web cluster, load balancing involves a several major concerns. The primary concern is measurement of work load. In different applications, workload has different meanings. In web services, the client request is a basic building block of load balancing and its response lively connections is a simple server load index.

Present web server clusters have some difficulties in providing services to the clients. First, in current websites dynamic workloads are becoming crucial, which imposes significant performance drop in web clusters with the shortcomings of present load balancing algorithms. When compared with the static web pages, the dynamic content requires high resource demands which leads to poor performance without suitable load balancing mechanisms in cluster-based web servers. Due to versatile demands, sometimes the request rate is greater than the cluster capacity. This is unpredictable with the flash crowds using the internet.

In this paper, a dynamic and robust load balancing mechanism is proposed for content aware dispatchers. In this work, three contributions are provided in the load balancing mechanism for web server clusters. The primary contribution is calculation of approximated load of a web server. Web requests are classified according to service time. The second contribution is a robust load balancing algorithm named Approximated Web Server Queuing Algorithm. The final contribution is instigation of a web server cluster using the proposed load balancing mechanism. To estimate the effectiveness of the proposed algorithm some experiments are conducted and compared with some of the present algorithms. The investigational results prove that the proposed algorithm will provide substantial gains in drop rate, throughput and mean response time.

The rest of the paper is ordered as follows: Section 2 catalogue some of the related works. Section 3 elucidates the architecture of web server cluster. Section 4 presents the proposed load balancing mechanism. Section 5 gives the experimental outcomes of the proposed algorithm. Section 6 outlines the conclusion.

2. RELATED WORK

Eager [3] *et al* projected that the idea of load sharing was to increase the performance by reallocating the workload between the servers available in the system. They demonstrate that effortless adaptive load sharing strategies, which mount up extremely modest amounts of state information and uses in very simple ways produce noteworthy performance enhancements [4]. They conclude that in practice, simple policies provide the greatest potential, for the reason that of their mixture of nearly ideal performance and innate stability.

Some of the presented works demonstrate that to administer web server clusters there is a need of load balancing algorithms [5], [6], admission control and overload [7], [8], performance optimization and architectural design [9], [10], job dispatching and redirection mechanisms. So many algorithms are proposed for load balancing in web clusters. The load balancing algorithms are classified as content aware (layer-7) and content blind (layer-4) algorithms [11], [12].

2.1. Content blind algorithms

These algorithms are broadly divided into various subset of algorithms. Most popular approaches among those are Round Robin, Random Server Selection, Least Connection, Least Loaded, Weighted Round Robin, Request counting, Weighted Least-Connection, Weighted Traffic Counting and Pending Request Counting. There are numerous additional algorithms like Locality-based Least Connection, Source and Destination Hashing, Never Queue and Shortest Queue First, which have need of out of the ordinary acquaintance to predict the best scheduling are discussed in a review paper [1].

2.2. Content aware algorithms

The researchers Pao and Chen projected a load balancing explanation by means of the remaining capacity of the replicas to regulate how the next request should be accomplished [13]. This enables the experts to estimate the behavior primarily to perceive the characteristics of the approach. The capacity is computed by means of available memory and CPU, the network transmission and number of active connections pending at the server. Nevertheless, due to the circumstance that brownout applications indirectly control CPU utilization, by fine-tuning the execution of optional content, so as to formulate for probable request bursts, conclusive on residual capacity alone is not a pointer of how a brownout replica is acting.

Lin *et al*. proposed a Server Content based Queue (QSC) load balancing algorithm by classifying the web request and considering the heterogeneity of web server [14]. In this algorithm, the client request is dispatched to the appropriate server which is least loaded. The load is calculated based on load state and server effectiveness. For each client request, random distributing base probability was used for server load distribution to select the appropriate server based on their weights. The selection course is carried out in a methodological approach such that there are no glitches during the processing.

Singh and Kumar [15] proposed a web server queuing approach for improving the efficiency of the web server. Overloaded server can't provide best service. In this algorithm, load collector and status monitor are introduced as two new components, which compute the overloading condition of the web server. Analysis of current serving capacity of the web server is also done.

2.3. Workload classification

Workload measurement of web services agrees on the load balancing on the internet. One of the prevailing protocols of internet is HTTP which overrides TCP to carry the web traffic. Earlier studies on Web workloads found that some important characteristics like reference locality, file popular distributions, target file types, file size and client request patterns are common to the conventional information provider sites. When the requests are independent and same size random and round-robin strategies are good enough [16].

Past two decades had a lot of changes in web applications subsequent to vast developments. For the majority part important one is "web page content is changing from static to dynamic leading to e-commerce became foremost web application; and continuous media gaining interests". For users, dynamic pages will endow with a distant better experience than static pages, but they impose some additional overhead on server resources like Disk I/O and CPU, thus this may indulge in monetary problems. For existing load balancing techniques these changes in workload characteristics will impose a challenge. Some strategies are no longer pertinent as their versions and corresponding applications change day by day. As an instance, size-based strategy will not work for dynamic contents for the reason that of its unknown size, the service time is unpredictable [17]. This is an inherent predicament in more or less all types of dynamic techniques well-known in literature. For the reason that of the dynamic page generations, the likelihood for caching to requested files declines and some of the requested files are even non-cacheable. This has to be addressed well with proper experimental investigations and analysis such that this constraint can be worked out for a feasible elucidation.

Zhang *et al*. projected novel load sharing policies in his research work [18], which concerned with the efficient usage of both Memory and CPU resources. This research has paved a way for many fascinated researchers to pursue the policies and look for fruitful practical results through appropriate trialing. These policies accomplish high performance underneath Memory and CPU concentrated workload circumstances. Lee *et al*. [19] considered two file assignments approaches for load balancing from corner to corner all disks, by this means making it achievable to perk up overall performance of system by completely making the most of hard disks to be used. Zhang *et al*. projected three I/O aware scheduling policies which aware of the job's spatial preferences. The preferences constantly cooperate an imperative responsibility in proper scheduling.

Zhang Xiayu *et al*. [20] consider CPU, Memory, Bandwidth, Disk I/O and Buffer pool slice rate to compute the load index in a cluster. They employ the operation of extension set, matter-element theory and dependent function which exists in extension theory. Xiao Qin *et al*. planned a load balancing approach

considering CPU, Disk I/O/ and Memory resources to calculate the load. The IOLB algorithm provides better memory and CPU utilization under memory and CPU rigorous workload circumstances. This algorithm is able to deliver the similar level of performance as two already existing memory and CPU aware load balancing approaches [21].

Ajay Tiwari *et al*. [22] proposed a dynamic content aware load balancing algorithm for web cluster in heterogeneous environment. This algorithm uses utilization ratio, queue length and server's processing capability as load indices. As the content awareness is given importance in this work, the processing part is maintained stringently to augment the utilization ratio. Saeed Sharifian *et al* in his research paper [23] categorizes dynamic requests into quite a lot of classes based on their impact on server resources. The CPU is the most important basis of tailback in the conception of dynamic contents.

3. ARCHITECTURE OF WEB SERVER CLUSTER

To improve the cluster performance, the load balancing algorithm which will run on load balancer plays a significant role. Distributed System is important to distributing the work load on the servers [24]. The Figure 1 represents the architecture which is widespread as today's web server cluster. The major components are collection of web servers and a content aware load balancer, in which the load balancing algorithm is deployed. In this model, all web servers are capable of handling both static and dynamic web pages and each web server have same pages. Load Balancer places a vital role in fulfilling the request of the clients through servers and for this work load balancer routes requests to those servers, which has the capability of doing its job in an effective way that is maximization of speed, maximum utilization of capacity and can fulfill the client's requests [25].

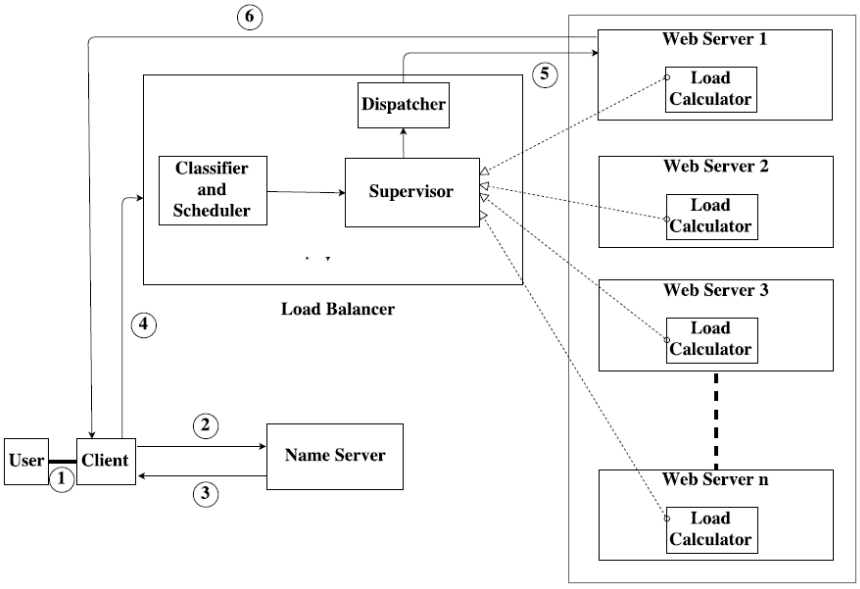

Figure 1. Proposed architecture of web server cluster for load balancing

User invokes the Client by opening the web browser. User requests through the client and enters the web URL and Client forwards the request for converting the URL into IP Address at the DNS. Based on the IP Address List the DNS forwards the IP address of the Load Balancer to the Client. Client sends the web page request from the user to Load Balancer with the received IP Address. The Load Balancer enters the relevant information for session entry into the database and forwards the requests of clients to the minimum loaded web server. Minimum loaded web server responses and serves the web request to the client directly.

Each web server has a procedure called "load calculator". This Load calculator will continuously observe the performance of the web server and calculate the load, by using the parameters Memory Usage, CPU usage, Disk I/O usage and Active Connections. The web server periodically sends the calculated load to the load balancer. Supervisor module in the load balancer will accumulates the load of each web server and dispatch the requests to the appropriate web servers based on this information.

3.1. Load balancer functional description

The load balancer performs the load balancing in web server cluster by classifying the client requests, Monitoring the order of request assignment (FIFO Scheduling) and dynamically assign the requests to the appropriate server (Dispatching).

3.2. Classifier and scheduler

Whenever a new request is received from the client the classifier classifies the request based on URL. Types of Request are:

a. *Issuance page*- It mainly consists of static information. This type of request is the simplest one, which includes html or other undivided documents.
b. *Affair page*- This type of request will be provided by performing operation on dynamic database via a dynamic HTML page (word, pdf or any other document). The weight value is more when compared to issuance page because it needs to access disk to obtain the particular type of document.
c. *Dynamic page*- It desires to inspect the information on the page dynamically without any hassles, which includes jsp, asp and php etc. Here the weight value is potentially large.
d. *Multi-media*- Affords with real-time video and audio services. The weight-value is more large than other types.

The scheduler runs continuously until there is availability of one non-empty FIFO queues. The scheduler chooses next request from the FIFO-queue such that it can be assigned to the dispatcher for prolific service.

3.3. Dispatcher

Assigning of the request to the Web server is depends on Load balancing algorithm once the dispatcher obtains the request from scheduler. Once the request is completed, the supervisor module receives the reports about completion time from the dispatcher.

Supervisor: The number of unresolved requests which are allocated to each webserver are tracked by the inherent supervisor module. Thus, counting the requests of same class and measuring the actual response time is done by supervisor module based on this information. Therefore, the prediction and correction of load is carried out periodically by the supervisor based on number of available requests in FIFO-queue.

4. PROPOSED LOAD BALANCING ALGORITHM

Approximated Web Server Queueing Algorithm

Step -1 Publishing Phase (At Web Server)

Takes place periodically

1. Every server calculates has the five parameters a, b, c and d. Where,
   a= CPU Usage of Web Server
   b= Memory usage of web server
   c=Disk I/O usage of web server
   d= percentage of remaining queue capacity
2. For each web server, the load calculator calculates,
   Server Load Capacity, SLC = (α*a) + (β*b) + (γ*c) + (δ*d)
   Where α, β, γ, and δ are weighting factors such that ($\alpha + \beta + \gamma + \delta$) = 1.
   α=0.4, β=0.3, γ=0.2 and δ=0.1
3. Report the Server Load Capacity to the Supervisor

Step-2 Selection Phase (At Dispatcher)

For each client request received from the classifier based on its class

1. Receive Load Capacity from all web servers periodically and initialize load values
2. Least Loaded Server, LLS_{Min}= Min (SLC_1, SLC_2, SLC_3, ……, SLC_n), where n represents the number of available web servers
3. The i^{th} server such that $LLS_i = SLC_{Min}$ is considered as the least loaded server to process the current request.
4. Dispatch the request to the i^{th} Server and add Approximated weight value based on the request class type to the Server Load Capacity

$$LLS_i = LLS_i + W_j \quad (i= \text{number of server, } j=\text{request class type})$$

Step-3 Processing Phase (At Web Server)

In i^{th} server:

if ($M = M_{min}$ or $C=C_{min}$), where M_{min} is minimum required memory and Cmin is Minimum CPU Required to process a request
then
    if (Number of requests in FIFO request queue of i^{th} server < queue capacity)
    then
    Add current request into FIFO request queue of i^{th} server.
    else
    Drop the current request
else the i^{th} server processes the current request.

4.1. Workload

The client requests are classified into four classes based on Weight value calculated by CPU, Memory, Disk I/O usage and percentage of remaining queue capacity. By generating constant request flow, the average weight value is calculated for each request type. The request types and weight value values are given in Table 1.

Table 1. Request Types and Weight Values

Class Type	File Name	Approximated Weight Value
C1 - Issuance page	Home.jsp	0.001
C2 - Affair page	Load.pdf	0.002
C3 - Dynamic page	Dynamic.jsp	0.003
C4 - Multi-media page	Video.mp4	0.005

4.2. Implementation setup

Implementation of the investigational test bed with both software and hardware configurations as explained below.

4.2.1. Hardware configuration

The web cluster consists of 60 computers configured as follows. One computer is used as DNS, one computer is used as dispatcher, 3 computers are used as web servers and 55 computers as clients. To provide transparency to the clients, one Virtual IP address is used for each dispatcher. The web server, each has an Intel i5-4590S 3.0GHz CPU with 4 GB of DDR RAM. The dispatcher is an Intel i5-4030 302GHz CPU with 8 GB of DDR RAM.

4.2.2. Software configuration

a. Client-side software

To scrutinize the performance of the proposed system, all modules are implemented using Java Development Kit (JDK 1.7). At Client side, web browser is used to generate requests and obtain responses.

b. DNS software

As discussed earlier, DNS-based schemes for load-balancing require that DNS returns the IP address of server or cluster, depends on the state information. Current application of the domain name server (BIND) provide such support. It supports random and round-robin selections of IP address.

c. Server software

All the server machines will run apache web server. But one could use any other software without necessitating any change in the architecture. In addition to the web server, also execute another process that gathers state information like load averages, Memory and CPU utilization, number of server processes running and number of active connections to handle client requests etc.

d. Load balancer software

Dispatcher is responsible for dispatching requests inside the cluster. Depending on the scheme, it can take into account loads on various servers and previous request rate of the clients, to choose a particular server. Dispatcher selects the web server depends on the load which is calculated as per the parameters mentioned above in the algorithm.

5. EXPERIMENTAL RESULTS

The proposed algorithm is simulated and the outcomes are compared with the other three existing load balancing mechanisms (RC, QSC and WSQ). In simulation only, heterogeneous environment is considered.

5.1. Heterogeneous environment of web server cluster with fixed queue length

The proposed research work for AWSQ approach is investigated based on simulations carried out for heterogeneous environments of web server with a constraint that the queue length is fixed. Each web server has different configuration and the experimental setup for simulation is with respect to the values mentioned in Table 2.

Table 2. Web Cluster Configuration with fixed Length Queue

Web Servers List	RAM Size	Number of Connections	Queue Length
Web Server 1	2 GB	2,00	30
Web Server 2	3 GB	3,00	30
Web Server 3	4 GB	4,00	30

In this work load intensity is varied step wise to compute the cluster throughput, drop rate and response time for the four load balancing algorithms.

5.1.1. Mean response time

The graphical results represented in Figure 2 demonstrates average response time of the discussed Approaches in comparison to proposed algorithm. It is vivid from the graph that, the mean response time curves are arranged exponentially for almost all the algorithms. In the initial stage it is observed that the shape of the response time curve is flat, later it starts to rise with increase in the number of client requests. Hence it is shown that AWSQ attains lower average response time in comparison with the existing RC, QSC and WSQ algorithms. Here weight correction leads to achieve the lowest response time. The higher average response time in the RC, QSC and WSQ approach is instigated by the circumstance of bottleneck in CPU for one or more web servers in the cluster owing to limitation of load balancing. The overall response time is increased sharply due to unbalanced loads, in the cluster.

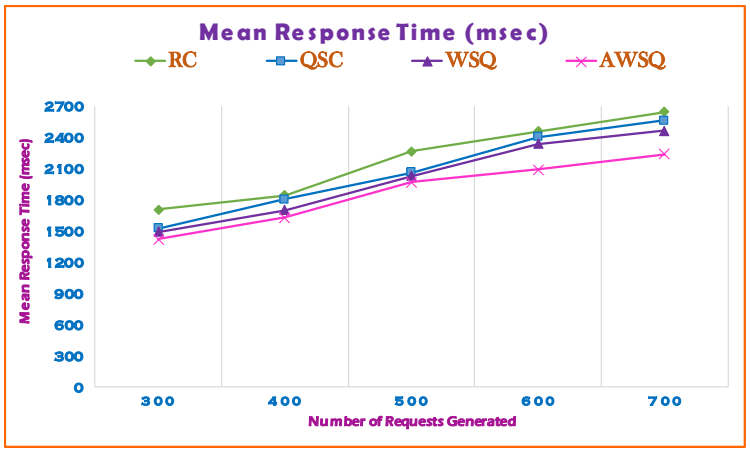

Figure 2. Mean response time of cluster with fixed length queue

For AWSQ approach the Mean Response Time starts at 1421 msec for 3,00 requests and it increases to 2241 msec for 7,00 requests. AWSQ approach performs better than the existing RC, QSC and WSQ approaches. For 3,00 clients request the Mean Response Time for RC approach is 1707 msec, for QSC approach 1522 msec and for WSQ approach 1490 msec. Similarly, as the experimentation is repeated for 7,00 clients request, the Mean Response Time for RC approach reaches to 2648 msec, for QSC approach the value is 2562 msec and whereas for WSQ approach the value yielded is 2465 msec. Thus, it is obvious that the less Mean Response Time is provided by AWSQ Approach from the above said analyses.

5.1.2. Throughput

The graphical results represented in Figure 3 demonstrates throughput of the discussed algorithms in comparison to proposed algorithm. In general, the behavior of throughput graphically raises at initial stage and after reaching a peak value it goes down drastically. The rise happen whenever the request rate is increased and it reaches to a peak during the bottleneck conditions of CPU resources on the web server.

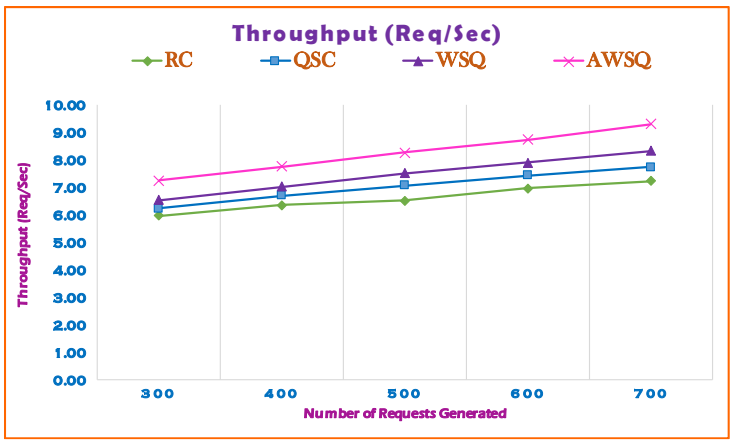

Figure 3. Throughput of Cluster with fixed length queue

As soon as CPU usage reaches to its extreme point, queuing begins which leads to drop in throughput. Based on the generated number of client requests and served requests the throughput has been calculated. For AWSQ approach the throughput starts at 7.25 requests/second for 3,00 requests and it increases to 9.32 requests/second for 7,00 requests. AWSQ approach performs better than the existing RC, QSC and WSQ approaches. For 3,00 clients request the throughput for RC approach is 5.98 requests/second, for QSC approach 6.24 requests/second and for WSQ approach 6.54 requests/second. Similarly, as the experimentation is repeated for 7,00 clients request, the throughput for RC approach reaches to 7.23 requests/second, for QSC approach the value is 7.74 requests/second and whereas for WSQ approach the value yielded is 8.34 requests/second. Thus, it is obvious that the high throughput is provided by AWSQ Approach from the above said analyses.

5.1.3. Drop rate

As the number of requests generated on the web server systems, it serves the requests as per the availability and some of the requests may not be served. In Figure 4 the drop rate of unserved request in the proposed web server system is lower in comparison to the RC, QSC and WSQ but it increases as the number of generated requests are increasing.

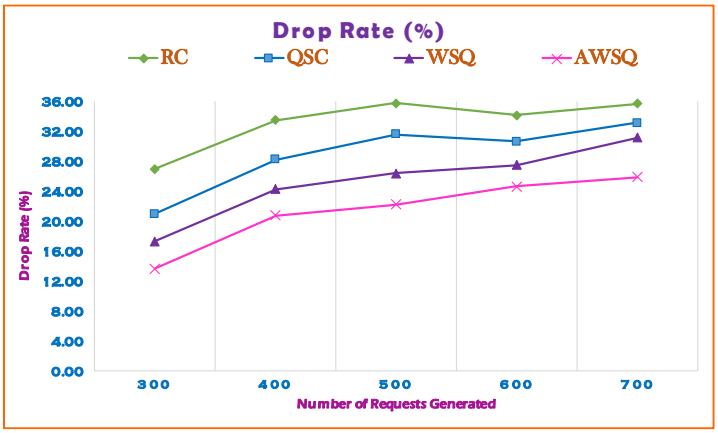

Figure 4. Drop rate of cluster with fixed length queue

Based on the generated number of client requests and Number of requests served, Drop Rate has been calculated. For AWSQ approach the drop rate starts at 13.67% for 3,00 requests and it increases to 25.86% requests/second for 7,00 requests. AWSQ approach performs better than the existing RC, QSC and WSQ approaches. For 3,00 clients request the drop rate for RC approach is 27.00%, for QSC approach 21.00% and for WSQ approach 17.33% Similarly, as the experimentation is repeated for 7,00 clients request,

the drop rate for RC approach reaches to 35.71% for QSC approach the value is 33.14% and whereas for WSQ approach the value yielded is 31.14%. Thus, it is obvious that the less drop rate is provided by AWSQ Approach from the analyses.

5.2. Heterogeneous environment of web server cluster with dynamic length queue

The proposed research work for AWSQ approach is investigated based on simulations carried out for heterogeneous environments of web server with a constraint that the queue length is dynamic. Each web server has different configuration and the experimental setup for simulation is with respect to the values mentioned in Table 3.

Table 3. Web Cluster Configuration with Dynamic Length Queue

Web Servers List	RAM Size	Number of Connections	Queue Length
Web Server 1	2 GB	2,00	30
Web Server 2	3 GB	3,00	40
Web Server 3	4 GB	4,00	50

5.2.1. Mean response time

The graphical results represented in Figure 5 demonstrates average response time of the discussed algorithms in comparison to proposed algorithm. For AWSQ approach the Mean Response Time starts at 1346 msec for 3,00 requests and it increases to 1945 msec for 7,00 requests. AWSQ approach performs better than the existing RC, QSC and WSQ approaches. For 3,00 clients request the Mean Response Time for RC approach is 1587 msec, for QSC approach 1456 msec and for WSQ approach 1407 msec. Similarly, as the experimentation is repeated for 7,00 clients request, the Mean Response Time for RC approach reaches to 2445 msec, for QSC approach the value is 2293 msec and whereas for WSQ approach the value yielded is 2238 msec. Thus, it is clear that the less Mean Response Time is provided by AWSQ Approach from the above said analyses.

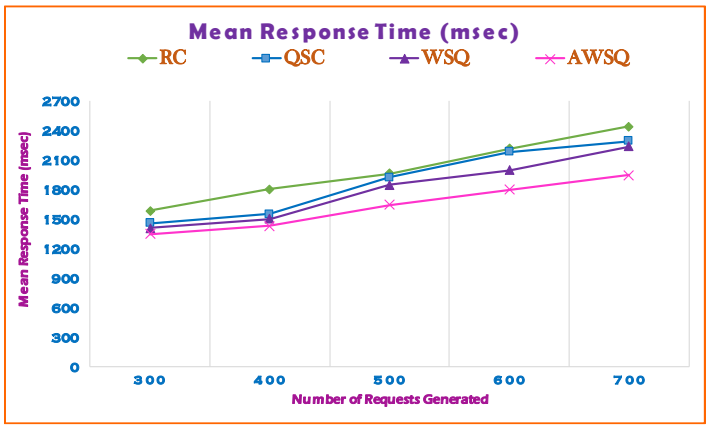

Figure 5. Mean response time of cluster with dynamic length queue

5.2.2. Throughput

The graphical results represented in Figure 6 demonstrates throughput of the discussed algorithms in comparison to proposed algorithm. For AWSQ approach the throughput starts at 8.04 requests/second for 3,00 requests and it increases to 9.96 requests/second for 7,00 requests. AWSQ approach performs better than the existing RC, QSC and WSQ approaches. For 3,00 clients request the throughput for RC approach is 6.27 requests/second, for QSC approach 6.65 requests/second and for WSQ approach 7.17 requests/second. Similarly, as the experimentation is repeated for 7,00 clients request, the throughput for RC approach reaches to 7.57 requests/second, for QSC approach the value is 8.13 requests/second and whereas for WSQ approach the value yielded is 8.76 requests/second. Thus, it is obvious that the high throughput is provided by AWSQ Approach from the above said analyses.

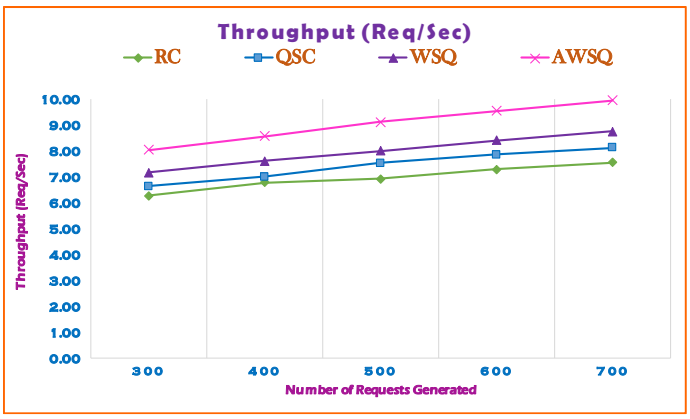

Figure 6. Throughput of cluster with dynamic length queue

5.2.3. Drop rate

In Figure 7 the drop rate of unserved request in the proposed web server system is lower in comparison to the RC, QSC and WSQ but it increases as the number of generated requests are increasing. For AWSQ approach the drop rate starts at 11.00% for 3,00 requests and it increases to 25.14% requests/second for 7,00 requests. AWSQ approach performs better than the existing RC, QSC and WSQ approaches. For 3,00 clients request the drop rate for RC approach is 24.67%, for QSC approach 17.00% and for WSQ approach 13.33% Similarly, as the experimentation is repeated for 7,00 clients request, the drop rate for RC approach reaches to 35.41% for QSC approach the value is 31.43% and whereas for WSQ approach the value yielded is 28.57%. Thus, it is obvious that the less drop rate is provided by AWSQ Approach from the above said analyses.

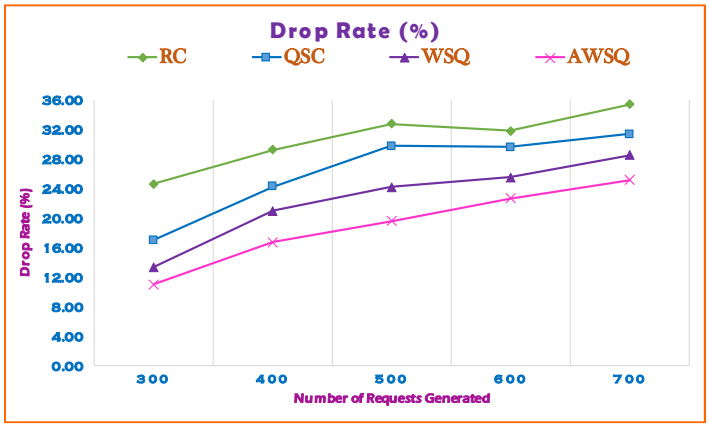

Figure 7. Drop rate of cluster with dynamic length queue

6. CONCLUSION

To expand the availability and to decrease the process of overloading of the servers in web server cluster system, a load balancing approach was used. In this research work, a competent load balancing approach, AWSQ has been formulated to accomplish load of the web servers using classification, scheduling and dispatching. To calculate the load CPU, Memory, Disk I/O and remaining capacity was used. Furthermore, a statistical analysis based on the comparison with the existing algorithms RC, QSC and WSQ the proposed approach has been designed and developed.

The experimental investigation and simulation analysis of the proposed algorithm yielded better optimal results in heterogeneous environment considering fixed and dynamic length queues. On comparison with the existing approaches the proposed one has the less drop rate. Therefore, the high throughput and processing of more number of requests can be done effectively. Subsequently for overload conditions, the approach leads to minimized mean response time.

REFERENCES

[1] K. Ramana and M. Ponnavaikko, "Web Cluster Load Balancing Techniques: A Survey," *International Journal of Applied Engineering Research*, vol/issue: 10(19), pp. 39983-39998, 2015.

[2] T. Sasidhar, *et al.*, "Load Balancing Techniques for Efficient Traffic Management in Cloud Environment," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 6(3), pp. 963-973, 2016.

[3] D. L. Eager, *et al.*, "Adaptive load sharing in homogeneous distributed systems," *IEEE transactions on software engineering*, vol. 5, pp. 662-675, 1986.

[4] H. Takagi, "Analysis of throughput and delay for single-and multi-hop packet radio networks," *Performance Evaluation*, vol/issue: 5(3), pp. 206-207, 1985.

[5] C. Emiliano and C. Michele, "A client-aware dispatching algorithm for web clusters providing multiple services," *Proceedings of the 10th International Conference on World Wide Web*, 2001.

[6] E. Casalicchio, *et al.*, "Content-aware dispatching algorithms for cluster-based web servers," *Cluster Computing*, vol/issue: 5(1), pp. 65-74, 2002.

[7] A. Kamra, *et al.*, "Yaksha: A self-tuning controller for managing the performance of 3-tiered web sites," *Quality of Service, 2004. IWQOS 2004. Twelfth IEEE International Workshop on. IEEE*, 2004.

[8] B. Schroeder and M. H. Balter, "Web servers under overload: How scheduling can help," *ACM Transactions on Internet Technology (TOIT)*, vol/issue: 6(1), pp. 20-52, 2006.

[9] M. Andreolini, *et al.*, "A distributed architecture for gracefully degradable Web-based services," *Network Computing and Applications, 2006. NCA 2006. Fifth IEEE International Symposium on. IEEE*, 2006.

[10] M. Aron, *et al.*, "Scalable content-aware request distribution in cluster-based network servers," *Proceedings of the 2000 Annual USENIX technical Conference. No. LABOS-CONF-2005-025*, 2000.

[11] T. Schroeder, *et al.*, "Scalable web server clustering technologies," *IEEE network*, vol/issue: 14(3), pp. 38-45, 2000.

[12] M. Andreolini, *et al.*, "Scalability of content-aware server switches for cluster-based Web information systems," *Proceedings of the 12th International World Wide Web Conference*, 2003.

[13] T. L. Pao and J. B. Chen, "The scalability of heterogeneous dispatcher-based web server load balancing architecture," *Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT'06. Seventh International Conference on. IEEE*, 2006.

[14] L. Zhang, *et al.*, "A content-based dynamic load-balancing algorithm for heterogeneous web server cluster," *Computer Science and Information Systems*, vol/issue: 7(1), pp. 153-162, 2010.

[15] H. Singh and S. Kumar, "WSQ: Web Server Queueing Algorithm for Dynamic Load Balancing," *Wireless Personal Communications*, vol/issue: 80(1), pp. 229-245, 2015.

[16] T. T. Kwan, *et al.*, "NCSA's world wide web server: Design and performance," *Computer*, vol/issue: 28(11), pp. 68-74, 1995.

[17] M. H. Balter, *et al.*, "On choosing a task assignment policy for a distributed server system," *Journal of Parallel and Distributed Computing*, vol/issue: 59(2), pp. 204-228, 1999.

[18] W. Zhang, "Linux virtual server for scalable network services," *Ottawa Linux Symposium*, vol. 2000, 2000.

[19] S. Y. Park, *et al.*, "Efficient inter-backend prefetch algorithms in cluster-based web servers," *HPC Asia*, 2001.

[20] G. Teodoro, *et al.*, "Load balancing on stateful clustered web servers," *Computer Architecture and High-Performance Computing, 2003. Proceedings. 15th Symposium on. IEEE*, 2003.

[21] X. Zhang, *et al.*, "An extension-based dynamic load balancing model of heterogeneous server cluster," *Granular Computing, 2007. GRC 2007. IEEE International Conference on. IEEE*, 2007.

[22] A. Tiwari and P. Kanungo, "Dynamic load balancing algorithm for scalable heterogeneous web server cluster with content awareness," *Trendz in Information Sciences & Computing (TISC), 2010. IEEE*, 2010.

[23] S. Sharifian, *et al.*, "A predictive and probabilistic load-balancing algorithm for cluster-based web servers," *Applied soft computing*, vol/issue: 11(1), pp. 970-981, 2011.

[24] S. Potluri and K. S. Rao, "Quality of Service based Task Scheduling Algorithms in Cloud Computing," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 7(2), pp. 1088, 2017.

[25] Khan Z., *et al.*, "Effective Load Balance Scheduling Schemes for Heterogeneous Distributed System," *International Journal of Electrical and Computer Engineering (IJECE)*, vol/issue: 7(5), pp. 2757-2765, 2017.