# Content-Based Load Balancing Using Web Clusters and Service Rankings

**T.N. Anitha and R. Balakrishna**

**Abstract** Web servers have gained immense popularity currently due to its nature to cater to a huge number of user requests, where any number of users can get service from Web-based applications. As it comes with global accessibility, servers hosting popular websites tend to get massive load that deteriorates in efficiency to provide quality service. Currently, most servers use load balancing techniques that distribute load among multiple virtual servers hosting a website. Generally, these load balancing techniques concentrate on balancing user request load on server and ignore the type of content requested. The reason behind it is overhead on dispatcher for analyzing content request and assigning appropriate rank. This process consumes additional time for dispatcher to allocate server and users face little delay in viewing their site; however it is helpful in the long run as heavy bandwidth consuming services such as multimedia requests can be catered efficiently by high priority servers only. The priority of servers can be defined as per their configuration and capabilities. Through experimental results on J Meter software it is proved that this concept can be helpful in providing better Quality of Service (QOS) to the website users.

**Keywords** Content-based load balancing · Clusters and Web servers · Content-wise service ranking

T.N. Anitha (✉)
Department of CSE, RajaRajeswari College of Engineering,
Mysore Road, Bangalore, India
e-mail: anithareddytn72@gmail.com

R. Balakrishna
Department of ISE, RajaRajeswari College of Engineering,
Mysore Road, Bangalore, India
e-mail: rayankibala@yahoo.com

## 1 Introduction

Highly developed Web-service delivery possesses considerable influence on organizations that are making use of it as a means to facilitate deliverance of their content to users. Similar to web pages, downloads, one-to-one communications, digital media and electronic-commerce need the latest and advanced tactics for content delivery. Simultaneously, size and dimensions of content rising widely need to convene ease of use. QOS and scalability is a gauge of the aptitude of the application to spread out to fulfill user requirements [1]. So as to convene service excellence specification sooner or later it is needed to affix more servers to convene rising needs. Cluster computing provides an influential setting to measure Web apps. Here in Web-service settings, users are able to notice merely the service and not the infrastructure needed for service deliverance. The content-based delivery services are explicit for content. Load balancing over the web offers newer prospects and efficiency of bandwidth usage [2].

Load balancing is a technique to share out workload through one or more servers. Here, even sharing of the load on many processors should get better the entire corresponding computation recital in clusters. In support of Web apps, load unprovoked situation happens often although the workload was disseminated time after time previously. As a result, vigorously and intermittently regulating workload sharing is necessary to ensure that every executive assignment at diverse sites would end their implementation at practically the same instance, reducing the inactivating time. On the other hand, active strategies to Web-services contain numerous confines [3, 4].

For getting better so that they do not tie together manifold services, there is a shortage of holdup for active and custom-made content formation and sharing and right to use. They do not shore up laterally content deliverance. There is shortage of flexibility in furnishing QOS access and practice [5]. The option of routing has too a huge influence on sharing requests as this sort of information accessible at the Web-server is fairly dissimilar. The sender is able to direct the requests in two ways; non-content-oriented and content-oriented techniques.

In this study, we talk about our idea, the limitations, and the study aims for sustaining upcoming generation streamed, interactive, and combined elevated resolution on content-oriented Web-services. We authenticated our application via content-wise service ranking (CWSR) adaptation. Additionally, we conducted a set of experiments to demonstrate the functionality of CWSR. Finally, we evaluate the content-oriented services sustained by CWSR to existing non-content-oriented service alongside the envisaged is of CWSR [6].

## 2 Related Works

The research on how to apply resource distribution of web system can be traced back to an auction since the last 20 years, there has been a noteworthy study attempt in load balancing. The fundamental supposition in most of this research is that the service scopes of the different requests are administered by an exponential supply. Contrary to the above statement, on the other hand, there is pretty tough evidence that the dimension of a web text, and for that reason its service scope, is administered in its place by heavy-tail sharing [7]. This entails that, to curtail reply time in that type of multi-base station method, “small” and “extensive” tasks must be allocated to diverse queues. In extremely variable workloads, software recital perks up significantly if comprehensive data regarding the action of every backend base station is accessible to the frontend sender. Therefore, a majority of investigators have considered and repeat in mock traces the area of web references and area conscious distribution rules that use the knowledge to widen the Web service accessibility [8]. Chuang Lin and associates largely concentrate on ventures assurance to provide the accessibility of operation services utilizing the stochastic high level Petri net of Web-server-clusters. Diverse classifications of requests are allocated diverse precedences [9]. These precedence intensities are employed and used into the load balancing algorithms to adopt precedence-oriented sending, i.e., QOS-alert load balancing. They calculated accessibility making use of MTBF and MTTR by means of the HTTP server scheduling and QOS alert load balancing to guesstimate accessibility of services [8].

In order to improve the competency of service, a set of Web-servers could be configured to make available Web service in a group to clientele. Load balancing is vital for a Web-service mechanism to ensure even sharing of inward requests on the Web-servers. Many approaches are there for load balancing over shared Web servers. The classification in [10, 11] divides the approaches of load balancing into four classes, namely client-oriented, DNS-oriented, dispatcher-oriented, and server-oriented approaches [12, 13]. From the above, we are able to discover that all load balancing mechanisms for shared Web servers entail recurrent message interactions among the request dispenser (DNS server or sender) and servers or clients to identify and swap over load data. The message interactions augment the network traffic in a Web-service method. A majority of the mechanisms also reveal the issue of restricted access in the routing and rerouting of requests. Generally, these mechanisms work only on load balancing without checking further consequences. Nonetheless content-based load balancing is obviously needed these days to cater to user requests more efficiently for high bandwidth usage services such as multimedia viewing or file downloading [14, 15].

## 3 Content-Wise Service Ranking

The load balancer dispatches the requests in two ways as content-based and non-content-based methods. In non-content-based the dispatcher dispatches the request to Web server without verifying the incoming request information. This method causes load imbalance and overhead on the Web servers. But in content-based method the dispatcher verifies the incoming request information, according to which a specific server providing that service is allocated. This leads to certain overhead on the dispatcher to match the requested service; to avoid this we provide ranking to the Web services. Ranks are assigned based on the highest hit rate of the service. For example, Rank1 to Multimedia Services, Rank2 to File Services, Rank3 to Text-based services, etc. Based on this, the priority of serving the requests is taken care of by the dispatcher here (Table 1).

### *3.1 Methodology for Content-Based Load Balancing in Clusters*

1. Give rank to servers as per its service;
2. Create three major clusters combining similar ranking servers;
3. Receive URL requests from user;
4. Keep incoming requests in a queue;
5. For any request, analyze type of content requested;
6. Get list of clusters and search content matching ranking cluster;
7. Get server list within the cluster;
8. Find load status of each server
9. Locate server with minimum load;
10. Verify availability of the least load server
11. If available, then forward user request to that server;
12. Else except this as down server and locate server with minimum load;
13. Go to Step 10.

**Table 1** Notation table

| Notation | Description | Notation | Description |
|---|---|---|---|
| $R$ | Ranking of server as per configuration | $n$ | Upper bound of server list |
| $C$ | Cluster of same ranking servers | LDS\|DS | List of down servers\|down server |
| UR | User request | $A$ | Availability of server |
| $Q$ | Queue | $A_{s_i}$ | Availability of $i$th server, where $i \geq 1$ and $i \leq n$ |
| $m$ | Upper bound of user request | $L\|L_{s_i}$ | Load\|load on particular server |
| $\mathrm{UR}_k$ | Particular user request in queue | $L_{\min}$ | Minimum load |
| LS\|S | List of servers\|server | $L_{\min_{s_i}}$ | Server with minimum load |

```
1.  $LS = \{S_1, S_2, .., S_n\}$
2.  $\{S_1, S_2, S_n\} \in R_1, \{S_1, S_2, S_n\} \in R_2, \{S_1, S_2, S_n\} \in R_3$
3.  $C_1 = \{LS(R_1)\}, C_2 = \{LS(R_2)\}, C_3 = \{LS(R_3)\}$
4.  $DS \leftarrow \emptyset,\ L_{min} \leftarrow 1000$
5.  $Q = \{UR_1, UR_2, .., UR_m\}$
6.  for all $UR \in Q$ do
7.            $R_x \equiv SR(UR)$
8.            $C_x = R_x$
9.            for all $S \in C_x$ and $i \le n$ do
10.                     $L_{s_i} \leftarrow \sum UR \in S_i$
11.                     if $L_{s_i} \le L_{min}$ then
12.                              $L_{min} \leftarrow L_{s_i}$
13.                              $L_{min_{s_i}} \leftarrow s_i$
15                      end if
16  end for
17  $A_{s_i} \leftarrow \text{INetAddress.isAvailable}\left(L_{min_{s_i}}\right)$
18  if $A_{s_i} \equiv$ true then
19     $UR_k \rightarrow L_{min_{s_i}}$
20  end if
21  else
22  $DS \leftarrow L_{min_{s_i}}$
23  $LDS \leftarrow LDS \cup \{DS\}$
24  $LS \leftarrow LS - LDS$
25  Repeat Step 9
26  end else
27   end for
```

The above algorithm explains the complete mechanism involved in internal processing of dispatch action. The very first step here is to obtain a list of all available servers and keep the names in temporary storage. Then as per the specific service of systems allot ranks to them such that multimedia service server gets Rank1 and Text Service server gets Rank3. Next, combine all the same rank servers in one cluster and form multiple clusters, so that for each rank there would be individual clusters. Now, down servers list can be initialized with null and min load server with 1000, assuming maximum load would be less than 1000 on any server. All the user requests can be placed in a queue with a queue limit and would be replaced with new ones after certain intervals. Now, for each user request dispatcher has to find the type of content requested so that the appropriate rank cluster can be found. Once cluster is found, the server inside has to be checked for minimum load. If minimum load server appears to be down then it should be kept in the list of down serves and again minimum load server should be searched, except servers listed in down server list. After it is found, the user request is forwarded to the respective server.

## 4 Experimental Results

For the experimental results generation J Meter tool is used, which is one of the most popular tools used for testing web applications. The speciality of this tool is it can generate any number of user requests considered as load for the web

**Table 2** Content-based load balancing system sample data

| Id | Sample time (Ms) | Label | Response code | Status | Thread name | Service | Reached | Bytes | Latency |
|---|---|---|---|---|---|---|---|---|---|
| 1411839684493 | 1347 | User requests | 200 | Ok | Users 1–10 | Text | True | 9461 | 1344 |
| 1411839683821 | 1321 | User requests | 200 | Ok | Users 1–5 | Text | True | 9461 | 2038 |
| 1411839683820 | 1339 | User requests | 200 | Ok | Users 1–4 | Text | True | 9461 | 2039 |
| 1411839684658 | 1325 | User requests | 200 | Ok | Users 1–11 | Text | True | 9461 | 1324 |
| 1411839684826 | 1242 | User requests | 200 | Ok | Users 1–12 | Text | True | 9461 | 1240 |
| 1411839685009 | 1242 | User requests | 200 | Ok | Users 1–13 | Text | True | 9461 | 1238 |
| 1411839685170 | 1230 | User requests | 200 | Ok | Users 1–14 | Text | True | 9461 | 1228 |
| 1411839685347 | 1184 | User requests | 200 | Ok | Users 1–15 | Text | True | 9461 | 1183 |
| 1411839685497 | 1399 | User requests | 200 | Ok | Users 1–16 | Text | True | 9459 | 1390 |
| 1411839685665 | 1303 | User requests | 200 | Ok | Users 1–17 | Text | True | 9459 | 1296 |

**Table 3** Non-content-based load balancing system sample data

| ID | Sample time (ms) | Label | Response code | Status | Thread name | Service | Reached | Status | Latency |
|---|---|---|---|---|---|---|---|---|---|
| 1411839691853 | 1180 | User requests | 200 | Ok | Users 1–54 | Text | True | 9459 | 1172 |
| 1411839692027 | 1188 | User requests | 200 | Ok | Users 1–55 | Text | True | 9459 | 1182 |
| 1411839692194 | 1172 | User requests | 200 | Ok | Users 1–56 | Text | True | 9459 | 1165 |
| 1411839692362 | 1154 | User requests | 200 | Ok | Users 1–57 | Text | True | 9459 | 1149 |
| 1411839692529 | 1162 | User requests | 200 | Ok | Users 1–58 | Text | True | 9459 | 1152 |
| 1411839692687 | 1146 | User requests | 200 | Ok | Users 1–59 | Text | True | 9459 | 1140 |
| 1411839692864 | 1208 | User requests | 200 | Ok | Users 1–60 | Text | True | 9461 | 1207 |
| 1411839693021 | 1228 | User requests | 200 | Ok | Users 1–61 | Text | True | 9461 | 1227 |
| 1411839693188 | 1186 | User requests | 200 | Ok | Users 1–62 | Text | True | 9461 | 1185 |
| 1411839693355 | 1173 | User requests | 200 | Ok | Users 1–63 | Text | True | 9461 | 1172 |

application. In order to work with J Meter there are minimum prerequisites that have to be done such as settings and making decision about parameters that have to be evaluated.

## 4.1 Settings

Before getting the results, a few configurations need to be done which are listed below:

1. Install server, database, and web application in the PCs.
2. Set rank of PCs (for example, Rank1 for multimedia service and Rank3 for text service).
3. Create cluster after grouping PCs of the same rank.
4. Creation of test project in J Meter.
5. Set the URL of application and parameters.
6. Set the number of user threads as 300 and ramp time as 60 s.
7. Add listeners on HTTP request results so that test results can be recorded.

Run the test and save the result file. Tables 2 and 3 demonstrate the sample data from J Meter which shows sample time taken for each user request in content-based proposed system.

Tables 2 and 3 demonstrate the sample data for content-based and non-content-based systems are generated from J Meter.

Figures 1 and 2 show content-based system takes 1185 or 1.2 ms and non-content-based system takes 1117 or 1.1 ms to serve the requests.

Figure 3 indicates there is a 0.1 ms latency comparison between non-content and content-based request processing. Figure 4 shows that content-based system has 26 % additional throughput compared to non-content-based system.

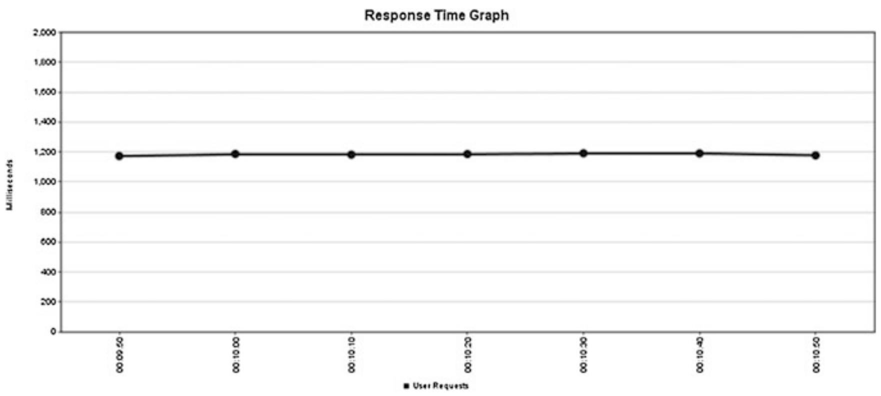


**Fig. 1** Content-based response time

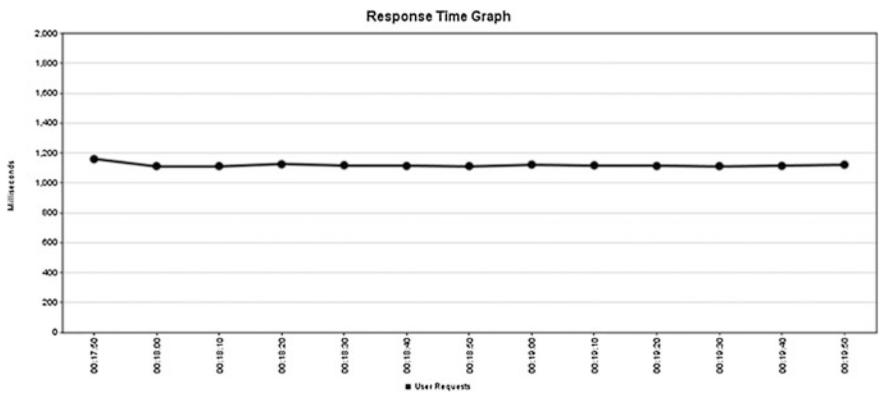


**Fig. 2** Non-content-based response time

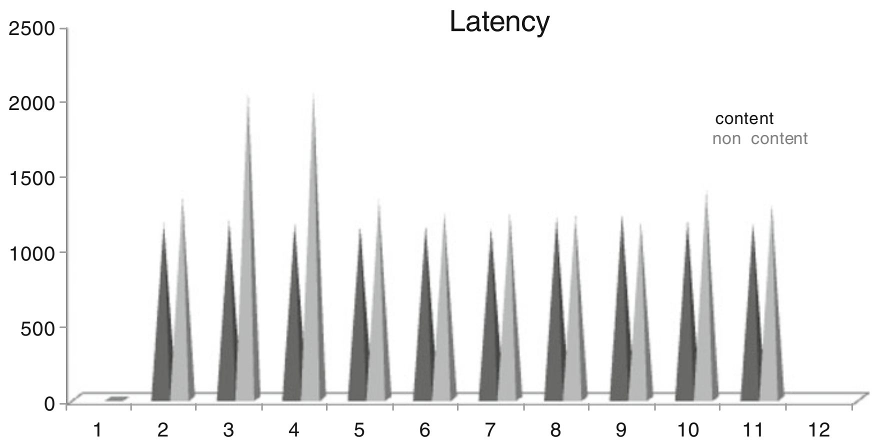
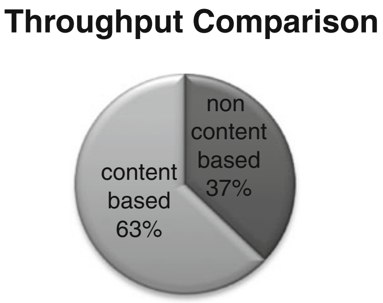


**Fig. 3** Latency of content and non-content-based load balancing systems

**Fig. 4** Throughput comparison



## 5 Conclusion

The above comparisons of sample requests show that non-content-based dispatcher obviously takes 0.1–0.2 ms time compared to proposed content aware dispatcher. This test was done for text-based service but can be treated as generic for any kind of service as current non-content-based dispatcher cannot make differentiating while redirecting requests at the time of load balancing. The difference comes when actual service is used by user such as viewing multimedia content. Here, if proper server is not allocated there are chances that user may fail to get quality service and this may result in huge latency and bad throughput.

## References

1. Sachin Kumar C, Singhal N. A priority based dynamic load balancing approach in a grid based distributed computing network. Int J Comput Appl. (0975–8887) 2012;49:5.
2. Mehta MA. A hybrid dynamic load balancing algorithm for distributed systems. J Comput. 2014;9:8 (Academy Publishers).
3. Ciardo G, Riska A, Smirni E. EquiLoad: a load balancing policy for clustered web servers. Performance Evaluation, Elsevier; 2001. p. 101–24.
4. Aversa L, Bestavros A. Load balancing a cluster of web servers using distributed packet rewriting. IEEE; 2003.
5. Gilly K, Juit C, Pulgjaner R. An up-to-date survey in web load balancing. Springer; 2011. P. 699–706.
6. Hussain J, Mishra DK. Performance evaluation of diffusion method for load balancing in distributed environment. Int J Adv Res Comput Communication Eng. 2013;2:12.
7. Menascè DA. QOS issues in web services. IEEE Internet Computing (November–December 2002).
8. Conti M, Gregori E, Lapenn W. A content delivery policies in replicated web services: client-side versus server-side. Cluster Comput. 2005;8:47–60 (Springer Science+Business Media, Inc. Manufactured in The Netherland).
9. Li J, Lin C. Availability analysis of web-server clusters with QOS-aware load balancing. In: International Symposium on Computational Intelligence and Design, IEEE; 2010.
10. Deshmukh AP, Pamu K. Applying load balancing: a dynamic approach. Int J Adv Res Comput Sci Softw Eng. 2012;2:6.
11. Zou S, Sha J. Analysis and algorithm of load balancing strategy of the web server cluster system. In: ICCIP 2012. Springer 2012.
12. Ayyasamy S, Sivanandam SN. A cluster based replication architecture for load balancing in peer-to-peer content distribution. Int J Comput Netw Commun (IJCNC). 2010;2:5.
13. Roussopoulos M, Baker M. Practical load balancing for content requests in peer-to-peer network. Berlin: Springer; 2006.
14. Dümmler J, Rauber T, Rünger G. Semi-dynamic scheduling of parallel tasks for heterogeneous clusters. In: 10th international symposium on parallel and distributed computing; 2011.
15. Tian C, Jiang H, Iyengar A, Liu X, Wu Z, Chen J, Liu W. Improving application placement for cluster-based web applications. IEEE Trans Netw Serv Manag. 2011;8:2.