TNC Transactions on Networks and Communications
SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
VOLUME 3, ISSUE 2
ISSN: 2054 - 7420

# Service Delivery Mechanism on Content Based Cluster Using Similarity of Services

**[1]T.N.Anitha and [2]Balakrishna.R**
*[1]Department of CSE, S.J.C. Institute of Technology, Chickballapur, India*
*[2]Department of ISE, Rajarajeswari College of engineering, Mysore road, Bangalore, India*
anithareddytn72@gmail.com; rayankibala@yahoo.com

**ABSTRACT**

Load balancing on web servers has become a major area of research due to ever increasing internet users' population and heavy load on popular website servers. Content based load balancing is proved to be a good mechanism to balance load on servers providing high quality services to the users' requesting for different category of content. On-demand creation of virtual servers has solved the complexity of load distribution and clusterization helps in grouping of same category servers. We propose a novel mechanism which works on clusterization of different grade servers intended to provide content based services to the users. Through experimental results it is found that this technique is helpful in increasing throughput and provides better quality of service to the users.

***Keywords***: Content based load balancing, Clusterization, ADC, Virtualization

## 1 Introduction

The flow of data traffic on internet is growing geometrically every year and so the load on cloud server to handle user requests. Yet users of internet expect page loading time to decrease due to the availability of high configuration systems at their end. It is very critical for popular websites to maintain high resource servers and new mechanism to keep response time to the lowest. Techniques such as virtualisation of servers, clusterization, load balancing, etc. are being studied to improvise the QoS (Quality of Service). A mix of clusterization and content based load balancing mechanism could be proved as a boon to provide high quality service to the website users.

A computer cluster is a set of connected systems, functioning in concert intimately in order that in several respects they create a single computer. The elements of a cluster are usually, but not at all times, linked to each other via rapid local area networks. Clusters are generally installed to enhance recital and/or accessibility over furnished by a single PC, whereas characteristically being very lucrative compared to single PCs of analogous speed or accessibility. Cluster Heads has the responsibility to make any interaction between its cluster members and ADC or server.

The computer comprises N uniform shared servers which offer the same documents, and a Cluster Domain Name Server (CDNS) which converts the URL-name into the IP-address of one of the servers in the cluster.

Clusters offer redundancy and distribution that make sure that website not at all goes down or loses vital operations or information. Clustered configurations permit simple scalability for parallel development, and are able to easily get a server offline for maintenance exclusive of compromise service. Particularly developed for businesses which insist high accessibility, clustered servers

**DOI:** 10.14738/tnc.32.1089
**Publication Date:** 7th April, 2015
**URL:** http://dx.doi.org/10.14738/tnc.32.1089

perform in recital for e-commerce websites, data-storage systems, internal networks, file and video distribution, high-volume blogs also other computing requirements. (Bader, David;, 1996)

## 1.1 Benefits of Dedicated Server Clustering

### 1.1.1 Redundancy and Trustworthiness

Diverse configurations in clustering could provide active or passive aspects in case one server breakdowns. Inactive choice comprises executing apps on a master committed server and containing a superfluous committed server to presume responsibilities if the master server breakdowns. In active configurations, two-server sets execute normal apps and represent from a general database with the intention that each server could occupy the other's responsibilities in cases of system breakdown.

### 1.1.2 Load Balancing

Configuring servers for utmost rapidity and recital when we have several traffic may needs dividing traffic and operations between servers for most favourable implementation. Targeted operations can be db, apps, storage systems or Web servers. Clustering permits us to perk up services radically, scale functions up or down hurriedly and identify cyber hacks prior to the reason for downtime.

### 1.1.3 High Accessibility

Clustering decreases singles points of breakdown and system susceptibility. Executing double load balancers, DBs, Web servers and superfluous network infrastructure avert downtime from break downs, cyber hacks, maintenance or natural calamities.

### 1.1.4 Data Growth

Irrespective of whether we manage a business, blog or aid or manage a data resource, a solitary committed server rapidly outgrows its processing and storage competencies. Having a clustering choice in place makes it simple to spread out without experiencing downtime which can reason for permanent harm to the status or loss of business.

### 1.1.5 Simple Maintenance

Server clusters permit for simple maintenance of the servers. If there is a trouble with a server, it could be detached from the cluster via either detaching the network wire or shout down the power. Once detached, the server could be repaired or reinstated. For the time being, the other servers in the cluster persist to execute processes providing as a minimum one server from the cluster relics online.

### 1.1.6 Rolling Upgrades

Server clusters make it simpler to upgrade servers or fix patches. As with any other maintenance, the servers in the cluster would go on with the essential processes, though merely one server from the cluster relics. Upgrading doesn't need downtime with a server cluster system.

# 2 Related Works

A generally hard problem in a shared setting is the recital squalor brought by an elevated load inequity and attaining lowest reply time for the clientele requests. Load balancing is hence vital for an assorted cluster, to promise a fair sharing of workload on every server in the cluster [1]. There are diverse methods for adopting load balancing in a shared assorted server setting. The taxonomy in [2] categorises the load-balancing methods into 4 groups: client-oriented, DNS-oriented, dispatcher-oriented, and server-oriented methods. Every one of these methods largely executes load

distribution algorithms that could be stagnant or dynamic and could utilize either centralized or shared control [3, 4, 5]. The Reference [6] represents that a hybrid of stagnant and dynamic approach for server choice offers a better recital. A client-oriented approach adopts the server choice on the clientele side [7]. The clientele could opt one of the servers in random but this random choice strategy could not promise load balancing and server accessibility. Alternatively the destination instigated approach needs a server to seek clientele requests [8] (from the overloaded servers). In a DNS oriented method, DNS server turns into a restricted access and confines throughput limiting performance [9]. A dispatcher oriented method acts address mapping at address point. A dispatcher oriented method might adopt either packet rewriting [10] wherein case the transparency of address rewriting is acquired [11] or the HTTP redirection that initiates high transparency compared to network load balancing, directing to weakening in performance.

## 2.1 History of Clusters

Greg Pfister has declared that clusters weren't discovered by any particular purveyor but by clientele who couldn't keep all their work on single system, or required a backup.[12] Pfister projects the date as some time in the 1960s. The official engineering base of cluster computing as a way of performing analogous exertion of any kind was debatably discovered by Gene Amdahl of IBM, who in 1967 printed what has approached to be viewed as the seminal paper on parallel processing: Amdahl's Law. (Bader, David;, 1996)

The history of near the beginning computer clusters is relatively directly attached into the history of early networks, as one of the main inspirations for the improvement of a network was to connect computing resources, forming a de facto computer cluster.

The foremost business clustering product was Datapoint Corporation's "Attached Resource Computer" (ARC) system, designed in 1977, and utilizing ARCnet as the cluster interface. Clustering as such didn't actually impression until Digital Equipment Corporation introduced their VAXcluster product in 1984 for the VAX/VMS OS (at present called as OpenVMS). The ARC and VAXcluster products not merely supported parallel computing, other than also distributed file systems and tangential tools. The thought was to give the benefits of parallel processing, whereas maintaining data dependability and exclusivity. Two other remarkable before time business clusters were the Tandem Himalayan (a circa 1994 high-ease of use product) as well as the IBM S/390 Parallel Sysplex (and circa 1994, mainly for commercial purpose). (Erguvan at el, 2009)

Within the same time framework, while computer clusters employed parallelism outer the computer on a product network, supercomputers started to utilize them within the same PC. Following the victory of the CDC 6600 in 1964, the Cray 1 was released in 1976, and launched interior parallelism by means of vector processing (Sedayao at el, 2008). Whereas in the early hours supercomputers expelled clusters and dependent on distributed memory, after a while some of the best ever supercomputers (such as K computer) dependent on cluster architectures.

## 2.2 Challenges in Cluster Computing

Load balancing consists getting the least loaded and paramount appropriate device in the network to run a work. In a local cluster setting this could simply be attained by centralized match making algorithms for example the one adopted in Condor (Erguvan at el, 2009). Centralized universal load creation pooling and drawing load-balancing decisions would not be realistic and measure well in universal cluster setting.

Dividing the universal cluster setting with a little interrelate topologies is the major for forming flexible cluster computing methods. Well-harmonized shared load balancing algorithms and protocols throughout clusters is essential to create most favorable resource distribution and utilization in an overall cluster setting. Even with such a system, getting the preeminent machine to carry out all jobs in global scale is not advantageous. One should observe the swapping among local and global optimization taking into consideration the price of stirring a job to distant sites for implementation. Meager cross-cluster implementation decisions may cause network overcrowding, and making systems inactive for long hours whereas making global appointment decisions and relocation of jobs. (Franco Milicchio, Wolfgang Alexander Gehrke, 2007)

Job scheduling comprises getting most appropriate profession to execute from amongst jobs belonging to numerous users and groups. In a local cluster setting this is accomplished by fair-share and main concern on the basis of setting up plans. Expanding this idea to divisions and projects extend throughout numerous geographic areas is the major for worldwide job scheduling optimization. Utilizing these global scheduling plans one should be competent to manage and implement project resource main concerns with challenging projects. This is needed for “good” throughput in global level. We refer better throughput since good consumption of resources for vital assignments as described by the user community. (Erguvan at el, 2009)

The algorithms of load-balancing which select the implementation place must also think about network stack, accessibility of user information, and safe implementation setting with same user qualifications all over the sites. These issues could be resolved with shared file mechanisms and computing setting services for example AFS, DCE/DFS, and Kerberos validation methods. (Sedayao at el, 2008)

Sharing of resources could be made in the vicinity at every site and through a centralized group for the universal system. As every site shares its own resources, it might permit users from other sites to distribute its surplus resources but provides foremost preference to the native users. When making use of a variety of the two strategies it might be feasible to have some of the resources assured for native users whereas remaining resources distributed among all other sites. The system has to be competent to implement these resource distribution strategies as the global distribution and operation snapshot is accessible at all stages. (Sedayao at el, 2008)

## 2.3 Architecture

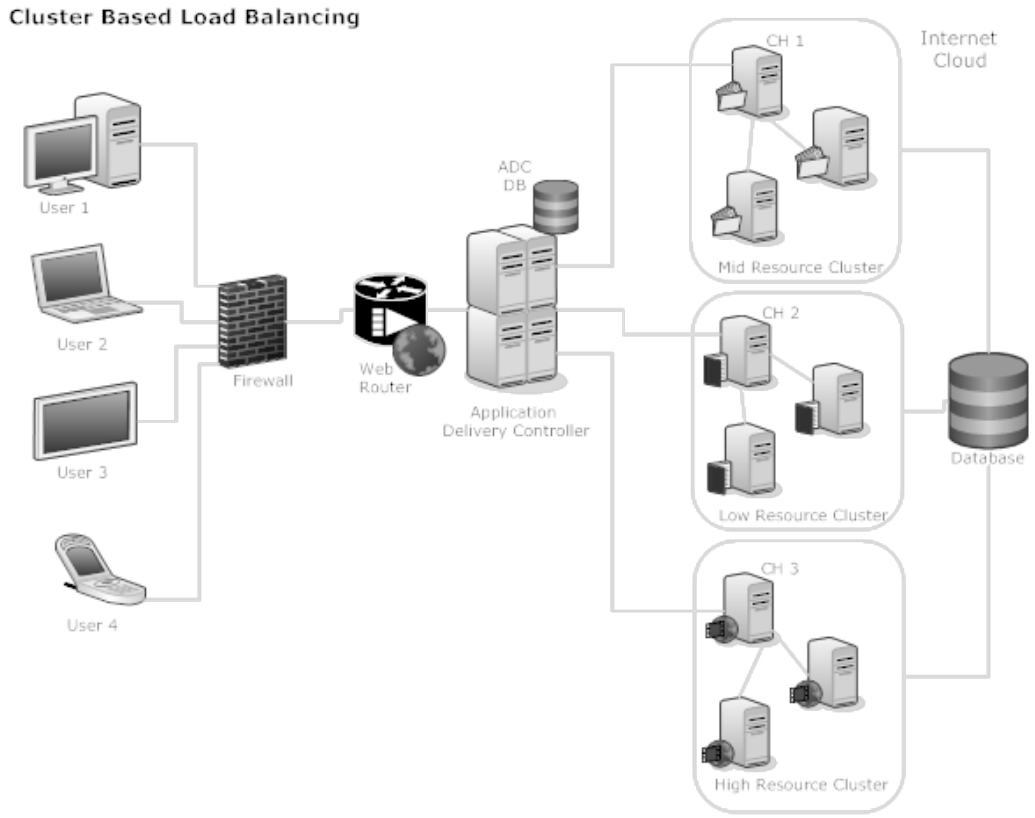


**Figure 1: Cluster Based Load Balancing**

# 3 Methodology

In the proposed mechanism, Multiple Virtual Servers (VS) are derived from Physical Servers (which is having capacity to serve .1 million user requests at an instance). VSs belonging to same set of configuration forms cluster. A cluster can keep maximum 382 VMs. User sends a URL request on browser to access a web application. After filtering through firewall it reaches to switch or router within LAN. URL Request is sent through ISP (Internet Service Provider) to the ADC Server. ADC analyses the requested content in User Request. ADC retrieves the list of Clusters belonging to that grade. ADC dispatches the request to Cluster Head (CH) of lowest load cluster. CH retrieves list of VMs within it and checks for lowest load VM. CH pings and checks availability of that VM, if available, then forwards User Request to the VM, else, searches for second lowest load VM which is available.

## 3.1 ADC

ADC or Application Delivery Controllers are high end load balancer devices used for distributing load among available virtual servers to enhance performance of any applications running these servers. These devices work on mechanism of receiving, analyzing and dispatching user requests to servers.

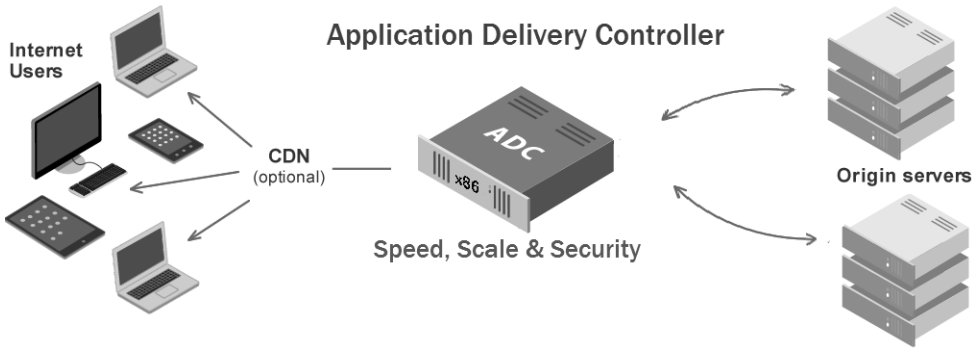


**Figure 2: Application Delivery Controller**

## 3.2 Cluster Formation

We propose a mechanism where clusters are formed dynamically as a result of excess amount of virtual servers in any cluster. We take a scenario where we assume a particular Physical Server (PS) which can handle at most .1 million User Request (UR) at a time.

$$Capacity(PS) \cong 100000\ UR$$

In our design, Clusters are formed out of same configuration VMs regardless of its distance and other parameter. The capacity of cluster to hold number of VMs is 382.

$$C_i = \{VM_1, VM_2, .., VM_{382}\}$$

Any excess of VM in a Cluster can lead to another cluster formation with exceeding number of VMs. Formation of Cluster is on-demand and it dynamically forms or dissolve whenever VM joins or leaves the group.

$$C_i\{VM_1, VM_2, .., VM_{382+1}\} = C_i\{VM_1, VM_2, .., VM_{382}\}, C_{i+1}\{VM_1\}$$

From each physical server multiple virtual servers are derived which resides in clusters for different configuration sets so that efficient services can be provided to all sort of user requests.

$$PS \ni \{C_1, C_2, .., C_n\}$$

Now as per the theorem we may say that total no. of requests served by all clusters belonging to one Physical Server could be equal to or less than Capacity of PS i.e. 0.1 million.

$$\sum UR \in (C_1 + C_2 + .. + C_n) <= (Capacity(PS) = .1m)$$

## 3.3 Service based Clusters

The configuration of Virtual Servers can be divided in 3 major categories i.e. high resource, medium resource and low resource VMs. Clusters are formed by joining similar configuration VMs and is completely on-demand. As no. of requests grow, new VMs are dynamically created and kept inside cluster. There could be 'n' no. of clusters as it depends upon which type of clusters are more. For e.g. clusters of low configuration set of VMs would take much less resource than clusters of high configuration VMs.

### 3.4 Algorithm Used

**Table 1: Notations**

| Notation | Description |
|---|---|
| PS \| VS | Physical Server \| Virtual Server |
| G | Grade of Server as per Resources |
| CL | Cluster of Same Grade Servers |
| Req | URL Request |
| CT | Content Type in Request |
| ADC | Application Delivery Controllers |
| $Req_k$ | Specific URL Request in ADC |
| SL \| S | Servers List \| Server |
| n | Upper Bound of Server List |
| LDS \| DS | List of Down Servers \| Down Server |
| A | Availability of Server |
| $A_{s_i}$ | Availability of $i^{th}$ Server, where i>=1 and i<=n |
| L \| $L_{s_i}$ | Load \| Load on Particular Server |
| $L_{min}$ | Minimum Load |
| $L_{min_{s_i}}$ | Server with Minimum Load |

1. $PS = \{VS_1, VS_2, \ldots, VS_n\}$
2. $\{VS_1, VS_2, VS_n\} \in G_1, \{VS_1, VS_2, VS_n\} \in G_2, \{VS_1, VS_2, VS_n\} \in G_3$
3. $CL_1 = \{SL(G_1) \leq 382\}, CL_2 = \{SL(G_2) \leq 382\}, CL_3 = \{SL(G_3) \leq 382\}$
4. $DS \leftarrow \emptyset,\ L_{min} \leftarrow 1000$
5. $PS \leftarrow \sum Req \leq .1million$
6. $ADC = \{Req_1,\ Req_2, .., Req_m\}$
7. ***for all*** $Req \in ADC$ ***do***
8. $\quad GetGrade(CT(Req)) \equiv G_x$
9. $\quad Req \rightarrow minLoad\big(CL(G_x)\big).CH$
10. $\quad$ ***for all*** $S \in CL_x$ ***and*** $i \leq n$ ***do***
11. $\quad\quad L_{s_i} \leftarrow \sum Req\ \in S_i$
12. $\quad\quad$ ***if*** $L_{s_i} \leq L_{min}$ ***then***
13. $\quad\quad\quad L_{min} \leftarrow L_{s_i}$
14. $\quad\quad\quad L_{min_{s_i}} \leftarrow s_i$
15. $\quad\quad$ ***end if***
16. $\quad$ ***end for***
17. $\quad A_{s_i} \leftarrow INetAddress.isAvailable\left(L_{min_{s_i}}\right)$
18. $\quad$ ***if*** $A_{s_i} \equiv true$ ***then***
19. $\quad\quad Req_k \rightarrow L_{min_{s_i}}$
20. $\quad$ ***end if***
21. $\quad$ ***else***

22. $DS \leftarrow L_{min_{s_i}}$
23. $LDS \leftarrow LDS \cup \{DS\}$
24. $LS \leftarrow LS - LDS$
25. ***Repeat*** $Step\ 10$
26. ***end else***
27. ***end for***

### 3.5 Random Walk Algorithm

**Table 2: Random Walk Algorithm**

| Notation | Stands For |
|---|---|
| $G$ | Graph |
| $V$ | Vertices |
| $E$ | Edges |
| $VS$ | Virtual Server |
| $t$ | Time |
| $P$ | Probability |
| $\mu$ | Allocation of Server |

Random walk algorithm works on probability distribution of virtual servers to be visited. It can be integrated in various kinds of P2P and web applications where multiple servers exist and each server need to be visited randomly. We propose this mechanism to choose a Cluster or Virtual Server from it to serve User Requests. Let us assume to have a Graph which is a set of Vertices and Edges which are interconnected. So, it can equated as

$$G = (V, E).$$

Beginning of the random walk happens with virtual server $VS_0$ that is either already defined to be picked first or kept in that position at the time of setup. Now as the random walk goes further to Virtual Sever $VS_1$ at time $t$, it reaches to neighbouring Server $VS_{t+1}$ at time $t + 1$ which is selected at random with a definite probability allocation $\mu$. Assume $\mu_t$ represent the allocation of virtual server$VS_t$, so that

$$\mu_t(m) = P(VS_t = m)\ \forall\ m \in V.$$

Let

$$P = \left(P_{m,n}\right), m, n \in V$$

Depict the Random Walk's evolution matrix, where $P_{m,n}$ is the possibility that random walk switches from virtual server $m$ to Virtual server $n$ in single go. $P_{m,n} = 0$ if virtual servers $m, n$ are not neighbours. The modus operandi of random walk is

$$\mu_{t+1} = \mu_t P = \mu_0 P^{t+1}.$$

## 4 Experimental Results

Various results were obtained duing experiments condected using JMeter Software on hetrogeneous systems. In order to have 2 levels of request dispatching, one system was made to act as ADC which was responsible for analyzing content requested and finding minimum load cluster of that grade and forwarding request to it's cluster head. Secound level of dispatching happens through cluster head which gets status of all its member servers and after checking availability forards request to it. To

analyze effect of load balancing different amount of request loads were produced on servers using JMeter.

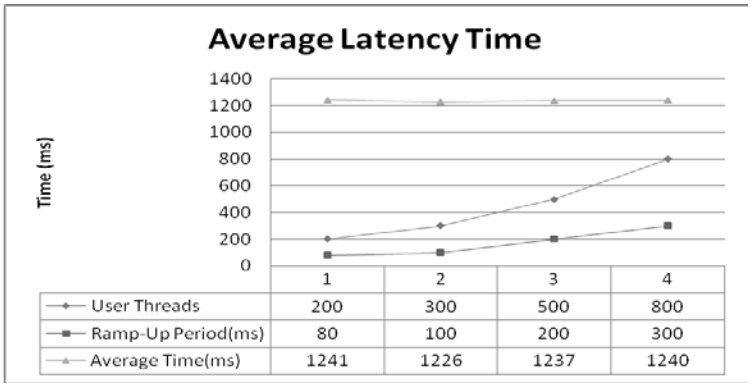


**Figure 3: Average Latency Time**

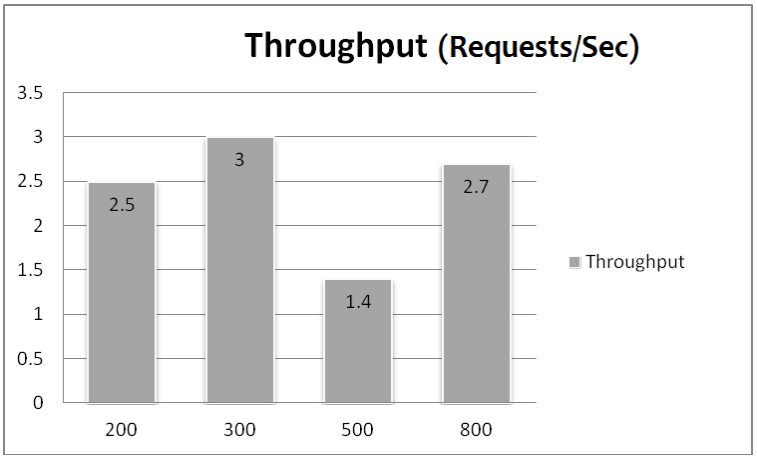


**Figure 4: Throughput (Requests/Sec)**

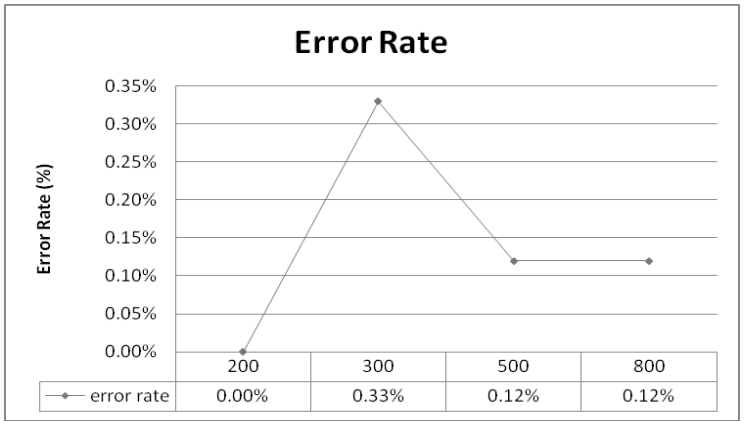


**Figure 5: Error Rate**

Fig 3 states the latency in serving use requests. It shows latency of 1241 milliseconds in handling each user request when load amount to 200 user requests, 1226ms for 300 user requests, 1237ms for 500 user requests and 1240ms for 800 user requests. It shows there is no particular effect of load due to increasing no. of user requests as the processing time remains approximately similar. Figure 4 represents throughput per second which is quite tend to vary with the processing speed of systems over time. It shows handling of average of 2.5, 3, 1.4 and 2.7 requests per second with error rate (ref. Figure 5) of 0%, .33%, .12% and .12% for 200, 300, 500 and 800 user requests respectively.

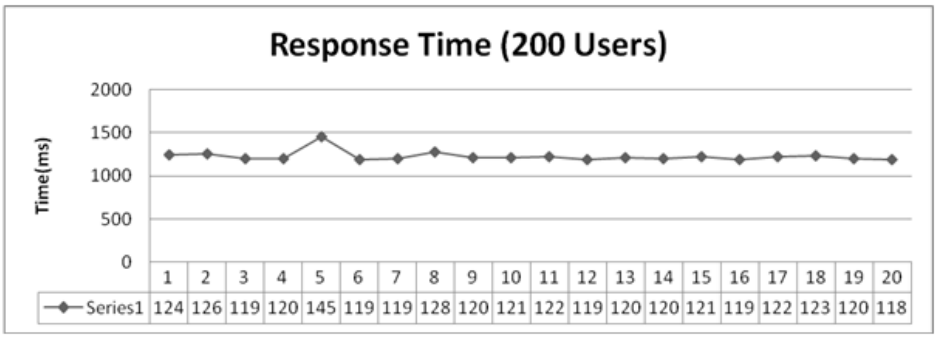


**Figure 6: Response Time (200 Users)**

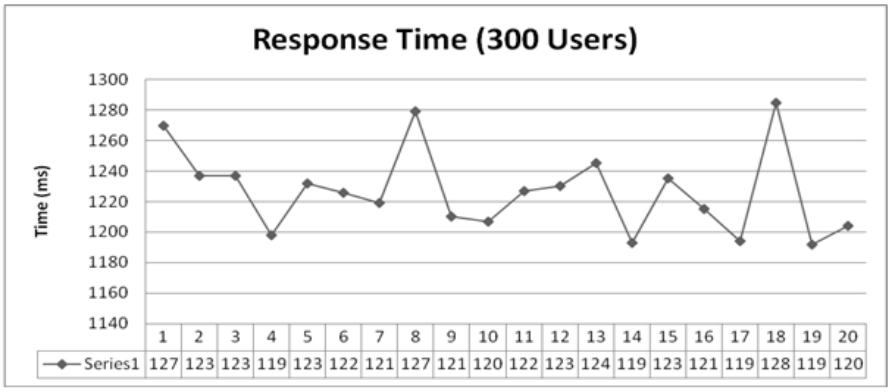


**Figure 7: Response Time (300 Users)**

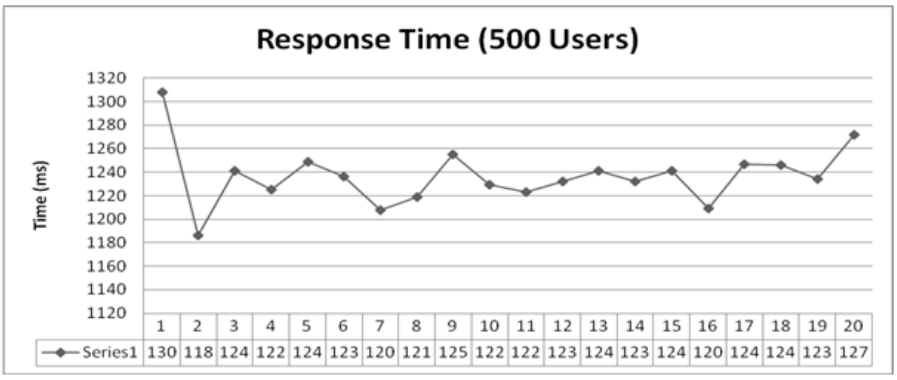


**Figure 8: Response Time (500 Users)**

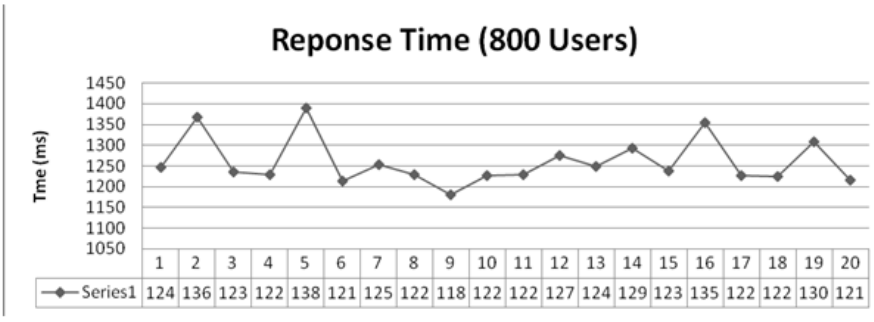


**Figure 9: Response Rate (800 Users)**

# 5 Conclusion

Content based load balancing has achieved immense weightage in the area of reasearch on load balancing on web and cloud servers. Clusterization techniques were found to be very much helpful in organizing virtual servers as per the ranking alloted based on their resource configurations. Experimental results on hetrogeneous systems has indicated this mechanism is helpful to achieve a good throughput with less error rate while doing load balancing on clustered servers. As there are 2 levels of dispatching required so initial server allocation takes little extra time but as this difference comes in milliseconds and can be ignored as it acts as preliminary to provide better QoS and satisfactory usage of service as per the content request.

Future works can be carried out to make this system more efficient and without any error rates by using additional improved mechanisms.

## REFERENCES

[1] W. Tang, and M. Mutka, "Load distribution via static scheduling and client redirection for replicated web servers," Proceedings of the First International Workshop on Scalable Web Services (in conjunction ICPP 2000), Toronto, Canada, pp. 127-133, 2000.

[2] V. Cardellini, and M. Colajanni, "Dynamic load balancing on web-server systems," IEEE Internet Compute 3, pp. 28-39, 1999.

[3] T.L. Casavant, and J.G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computer systems," IEEE Trans. Software Eng. 14 (2), pp.141-153, 1988.

[4] J. Cao, G. Bennett, and K. Zhang, "Direct execution simulation of load balancing algorithms with real workload distribution," J. System. Software 54, pp. 227-237, 2000.

[5] Y. Wang, and R. Morris, "Load sharing in distributed systems," IEEE Trans. Compute. C-34 (3), pp. 204-217, 1985.

[6] M.J. Zaki, W. Li, and S. Parthasarthy. "Customized Dynamic Load Balancing for a Network of Workstations," Journal of Parallel and Distributed Computing 43, pp. 156-162, 1997.

[7] C. Yoshikawa, B. Chun, P. Eastham, A. Vahdat, and T. Anderson, "Using smart clients to build scalable services," Proceedings of USENIX, pp. 105-117, 1997.

[8] D. Eager, E. Lazowska, and J. Zahorjan, "A comparison of receiverinitiated and sender-initiated dynamic load sharing," in Perform. Eval.Vol. 1, pp. 53-68, 1986.

[9] Cisco Distributed Director, http://www.cisco.com/warp/public/cc/pd/cxsr/dd/index.shtml.

[10] A Bestarvros,M Crovella, JLiu,and D. Martin, "Distributed packet rewritting and its applications to scalable web server architectures," Proceeding of the Sixth International Conference on Network Protocols, Austin, TX, pp. 290-297, 1998.

[11] D. Dias, W. Kish, R. Mukherjee, and R. Tewari, "A scalable and highly available web-server," Proceedings of the 41st International Computer Conference (COMPCON'96), IEEE Computer Society, San Jose, CA, pp. 85-92, 1996.