Optimization of Cluster Web Server Scheduling from Site Access Statistics

Nartpong Ampornaramveth, Surasak Sanguanpong
Faculty of Computer Engineering, Kasetsart University,
Bangkhen Bangkok, Thailand
E-Mail: {g4165217, nguan}@.ku.ac.th

Abstract

In order to satisfy high volume of site accesses, cluster of web servers have been widely deployed by many popular web sites. This paper reviews a number of parallel web server architectures and proposes a method for calculating the server scheduling weights for efficient cluster performance by taking into consideration the web site's access statistics. Conventionally, equal weights are used for homogeneous cluster of web servers which have identical hardware configuration. However, for many web sites, the cluster is constructed with new servers being added as the site slowly gains popularity. As hardware architecture changes over time, these sites tend to end up with a heterogeneous cluster with servers of different hardware configuration.

In order to obtain the optimal overall performance, the scheduling weights for such a heterogeneous cluster must be carefully calculated to maximize the performance of each server. Traditionally, the server performance is measured using benchmarks. In this paper, CPU benchmark and Disk I/O benchmark are used as base parameters for weights calculation. A typical web site consists of both static pages, which are I/O-bounded, and dynamic pages, which are CPU-bounded. We will make use of statistical data from site access log to identify the ratio of static load to dynamic load. Finally the server performance is obtained from the summation of CPU benchmark and Disk I/O benchmark, adjusted with respect to this ratio.

Experiments were carried out on a cluster of three heterogeneous servers. Results showed that our method offer significant amount of cluster performance gains in comparison with the use of CPU or Disk I/O benchmark alone.

Keywords: Cluster, Web Server, Load Balancing

1. Introduction

As web site gains popularity and its traffic increases, the original single web server will no longer be able to handle the increased requests. In order to cope with these excessive requests, one has to upgrade the existing hardware to achieve higher performance or utilize cluster web technology. The cluster model requires addition of mid-range server which has the advantage of being more cost-effective and more robust against hardware failure. It is also easily scalable to meet increase traffic by adding additional servers when required. Moreover, the total capacity of the cluster servers outperforms that of a single server machine. Google, one of the best-known search engines, is said to deploy more than 10,000 web servers for its query processing [1].

The deployment of cluster web servers is rather complex as all of them serve same url and contents. The requests have to be evenly distributed to each web server depending on its processing capacity. There should also be minimal impact in the event of any server outage.

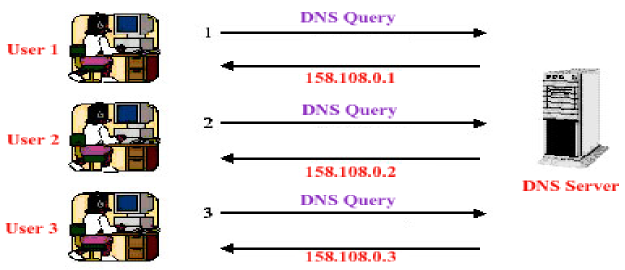

Figure 1: DNS Round Robin Method

2. Cluster Web Server Technology

Various scheduling algorithm has evolved from the past to the present and they are as followed:

2.1 DNS Round Robin

In general, a url is mapped to an IP address in the DNS server, eg. www.ku.ac.th has ip address 158.108.2.71. In order to have more web servers serving the same url, we can add ip addresses to the same domain name in the DNS server, eg. www.ku.ac.th now has ip addresses 158.108.2.71, 158.108.2.72, 158.108.2.73**.** Web servers are selected in a sequential order, from first web server to the last web server and then back to the first web server again (see Figure 1). This scheduling algorithm does not take individual server performance into consideration thus not fully utilizing the resources of the higher performance server. Another drawback is unavailability of web site to certain users in the event of a server failure as immediate removal of failed server's IP address from DNS is impossible.

2.2 Application Load Balancing

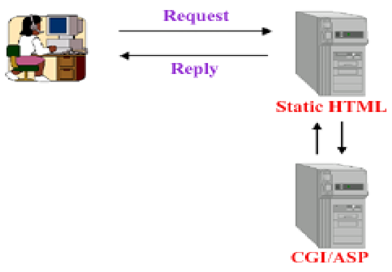

Figure 2: Application Load Balancing Method

In general, a web site has both static and dynamic pages. Static pages refer to the pages that have same contents in response to requests from all users. Dynamic pages have contents that can be different from request to request such as the results of a search engine or changes with time such as live stock prices. Application load balancing works in 2 steps (see Figure 2) user will connect to the main server which serves static pages first and if the request involves dynamic content, then the main server will send a CGI/ASP request to the dedicated web servers designated for dynamic contents. These server(s) generally have higher performance.

This method is better than having single web server serving both static and dynamic pages as it results in faster response. Application load balancing is being used in pWEB [2], NonStop WebCharger 1.0A+ (Tandem)

2.3 HTTP Redirect

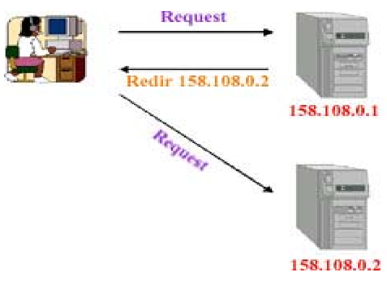

Figure 3: HTTP Redirection Method

The two methods mentioned above do not take individual existing server loading into consideration thus resulting in unbalanced loading on the web servers. In http redirect, web page requests are submitted to a load balancer which monitors the loading in each web servers and redirect the request to the server with the least loading. The main responsibility of the load balancer is to ensure all web servers perform at near capacity.

Figure 3 shows an example where browser requests are sent to load balancer (158.108.0.1) and gets redirected automatically to 158.108.0.2 which has the least load among the other servers. Http redirection is being used in Scalable Web Server Architectures (SWEB) [3], ClusterCATS 1.1 (Bright Tiger Technologies). The disadvantage of http redirection is that the browser has to submit two requests in order to retrieve a single page. The system performance is thus not satisfactory due to this overhead.

2.4 Using Virtual IP Address

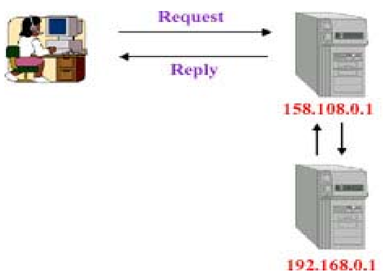

Figure 4: Using Virtual IP Address with Reply thru Load Balancer

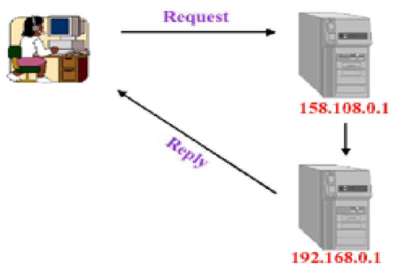

Figure 5: Using Virtual IP Address with Direct Reply

Virtual IP address is the most commonly used technique as only the load balancer owns a global IP address on the Internet. IP addresses of all the web servers are private numbers which are inaccessible from the internet (eg. 192.168.*.* or 10.*.*.*). The only way to access the site is through the load balancer's IP address thus confines security management to the load balancer only.

In this setup, web page requests are sent to the load balancer which has valid internet IP address and then directed to the web servers. The reply can be sent back to the client either via load balancer or direct (bypass load balancer).

For reply sent via load balancer, the bottleneck will be at the load balancer as it has to handle both incoming and outgoing traffic (see Figure 4)

For reply sent directly to the client (see Figure 5), the loading on the load balancer is greatly reduced as it only needs to service incoming requests from the client. The incoming request is significantly small in size compare to outgoing reply. Example of incoming request will be a one-line "get filename" statement whereas outgoing reply will be contents like html file or gif image. This setup enables more requests to be serviced but it is more difficult to setup.

The disadvantage of virtual IP address is that there exists a single point of failure which is the load balancer. The use of backup load balancer has been developed for use in such situation [4].

3. Server Scheduling Algorithm

The scheduling algorithm for the selection of the web servers can be categorized into 4 types:

3.1 Round Robin Scheduling

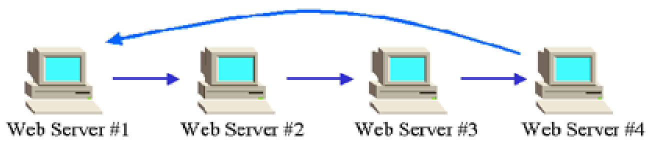

Figure 6: Round Robin Server Scheduling

Web servers are selected one by one, in a sequential order, from first web server to the last web server and then the first web server again as shown in Figure 6.

3.2 Weighted Round Robin Scheduling

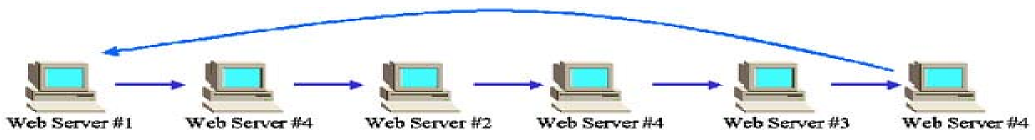

Figure 7: Weighted Round Robin Server Scheduling

If the web servers have different performance (heterogeneous cluster), the weights can be assigned to each server by its performance factor. By assigning higher weight to servers with higher performance, the load can be better balanced. For example, if the test results indicate that web servers #1-3 can process 150 requests/second and web server #4 can process 450 requests/second. The weights should be 1,1,1,3 for servers #1,2,3,4 respectively. This means that, in six requests, web servers #1,2,3 will each get 1 request, but web server #4 will get 3 requests.

3.3 Least Connection Assignment

As all web requests are being handled through load balancer, the load balancer is able to determine the number of connections to each web server which correlates to the loading on each of them. The least connection web server will be selected since it will have the least loading.

3.4 Weighted Least Connection Assignment

This scheduling algorithm uses the same basis as weighted round robin as in 3.2 in that higher weights are assigned to servers with higher performance for better load balancing. However, servers are selected using the least connection instead of round robin fashion.

4. Determine Weighted Values from Site Access Statistics

From the use of weights as in 3.2 and 3.4, we can see that the web servers can perform at their peak through the selection of optimal weighting factors, which can be obtained only by benchmarking the server performance in serving the real site's dynamic and static contents, under the real user access pattern. However, due to the fact that contents can be updated over the time, and because it is inconvenient to measure servers' performance every time the update occurs, it is more desirable to use some estimates of server performance in calculating the weights.

Since the raw CPU processing speed has direct connection to the server performance, the simplest form of weights can be obtained from CPU speed:

$$W(i) = Pc(i) \qquad (1)$$

where W(i) is the weight of web server i and Pc(i) is CPU speed of the web server. As static web pages are I/O-bounded and dynamic web pages are CPU-bounded, one may suggest the weight value also include disk I/O value as followed:

$$W(i) = \alpha Pc(i)+(1-\alpha)Pd(i) \qquad , 0<=\alpha<=1 \qquad (2)$$

where Pd(i) is disk I/O of web server i and α is the ratio determining the contribution of Pc and Pd to the server weight. Web Server that serves static web pages usually has a low α value as disk I/O plays a more important role in determining the response of web access. On the contrary, web server with more dynamic web pages should have higher α value. The proper value of α can be obtained from user access pattern. In our approach, this value is derived from the site access statistics:

$$\alpha = Nd / (Ns+Nd) \quad (3)$$

where Nd is number of access to dynamic web pages and Ns is number of access to static web pages during a specified time period.

5. The Experiment

To determine web server performance with respect to the weighting values determined from equation (2) and (3), we have conducted tests using a load balancer, cluster of web servers, and client machines as shown below:

Load Balancer Specifications
1) PII 450MHz, SDRAM 256MB (Linux 2.4.10, Linux IP Virtual Server 0.8.1)
Cluster of Web Servers
1) AMD Athlon 1.3 GHz, SDRAM 256MB, IDE 7200rpm x 1, Linux 2.4.10, Apache 1.3.20
2) Intel Celeron 366MHz, SDRAM 256MB, SCSI 10000rpm x 1, Linux 2.4.10, Apache 1.3.20
3) Intel Dual PIII 933 MHz, SDRAM 256MB, IDE 7200rpm x 4 (Linux RAID0), Linux 2.4.10, Apache 1.3.20
Client Machines (for submitting HTTP requests)
1) AMD Athlon XP 1500+, DDR SDRAM 256MB, Windows XP
2) Intel PIII 600 MHz, SDRAM 256MB, Windows XP
3) Intel PIII 550 MHz , SDRAM 256MB, Windows XP
4) Intel PIII 450 MHz, SDRAM 256MB, Windows XP

Test Environment

Load Balancer and Web Servers are setup using Virtual IP address configuration. All servers and client machines are connected to a 100Mbit/s Switch with 100Mbit/s network interface cards (NIC).

In our experiment, the values of Pc and Pd in equation (2) are derived from:

Pc: CPU time used in compiling Linux kernel 2.4.10 on each web server

Pd: Raw random disk access performance as reported by "Bonnie" benchmark [5] using 500MB of simulated data

Overall performance of web server is measured using Zdnet Webbench 4.01 program [6]. A set of 4 client machines are used for submitting simulated http requests. Static web pages test data consists of 6,000 files with the total size of 70MB which are provided as part of the benchmark program. Dynamic web pages contents are from the simcgi program bundled with the test program.

6. Test Results

Pc values of web servers are obtained from the inverse of time spent in compiling Linux kernel 2.4.10 (make –j 10) and the results are shown in Table 1.

	Server 1 (Athlon 1.3GHz)	Server 2 (Celeron 366MHz)	Server 3 (Dual PIII 933MHz)
Ts: Time (Second)	177	631	130
Pc: (1/Ts) normalized to 100	38	11	51

Table 1: Time required for compiling Linux kernel and Pc

Pd values are measured using Bonnie program which returns the number of random seek/sec in accessing 500MB data and the results are shown in Table 2.

	Server 1 (IDE 7200rpm)	Server 2 (SCSI 10000rpm)	Server 3 (IDE 7200rpm x 4)
Bonnie Results (seek/sec)	112	348	405
Pd: normalized to 100	13	40	47

Table 2: Bonnie Benchmark Result and Pd

In our experiment, we measured the performance of this cluster web server under six different server scheduling methods

RR: Round Robin Scheduling
LC: Least Connection Scheduling
WRR-Disk: Weighted Round Robin Scheduling with $\alpha=0$
WLC-Disk: Weighted Least Connection Scheduling with $\alpha=0$
WRR-CPU: Weighted Round Robin Scheduling with $\alpha=1$
WLC-CPU: Weighted Least Connection Scheduling with $\alpha=1$

The stand alone server performances are also included.

In the first experiment with 6000 files (70MB total size) of static web content, we found relatively small differences among individual server performances as shown in figure 8. In this figure, the number of concurrent http client threads on all the four client machines is shown on the horizontal axis. Despite the gaps in hardware specification, each individual server can deliver nearly 500 req/sec. This is due to the fact that all the 70MB data can be easily stored in disk cache area of the server’s 256MB memory, the performance is then determined solely by the available 100Mbps network bandwidth. All three servers perform equally well from the bandwidth point of view. Thus, the overall cluster performance of LC and RR methods with no weight are better than the weighted ones.

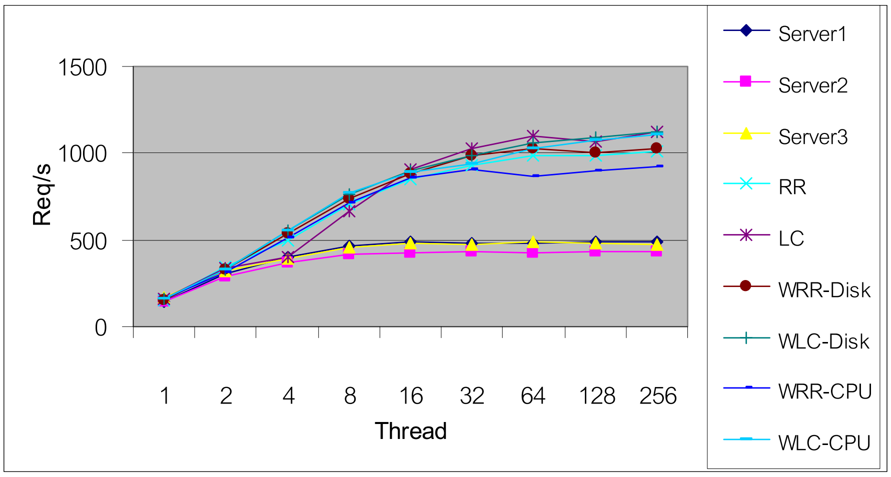

Figure 8: Experiment with 6000 files (70MB total size) of static web content

In order to avoid file caching, we increased the data size to 42,000 files of static content (490MB) and repeated the tests. The results are shown in figure 9. It can be seen that disk performance has considerable impact on the server performance here. With the lowest Pd value, server1 is the worst performer and we obtain the best cluster result when using weighted least connection scheduling whose weights are calculated from disk performance: WLC-Disk.

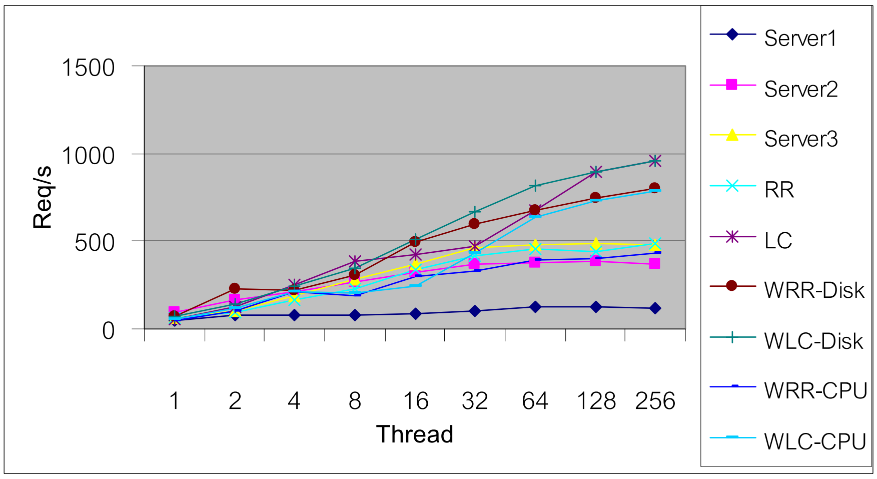

Figure 9: Experiment with 42000 files (490MB total size) of static web content

For dynamic web content, we use WebBench's "simcgi" test suite in the third experiment and got the results in figure 10. Good results are obtained when weights are biased by the CPU speed (Pc) and the best performer in this experiment is WLC-CPU.

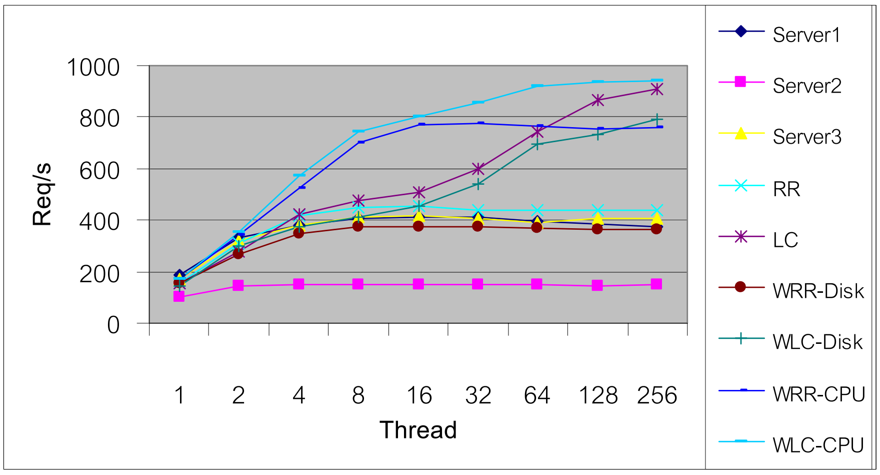

Figure 10: Experiment with dynamic web content

To imitate the real world web site with mixed dynamic and static contents, we configure the client to request for static content 66.6% of the time, and dynamic content 33.3% of the time. In this experiment, we also add two more tests using weights calculated by equations (2) and (3) with α=1/3 (33.3%) namely: WRR-Mix and WLC-Mix for round robin and least connection scheduling respectively.

The server weights, as calculated from equation (2), are as shown in table 3.

	Server 1 (Athlon 1.3GHz, IDE 7200rpm)	Server 2 (Celeron 366MHz, SCSI 10000rpm)	Server 3 (Dual PIII 933MHz, IDE 7200rpm x 4)
Pc	38	11	51
Pd	13	40	47
W: α=1/3	21	30	49

Table 3: The new server weights

The result in figure 11 validates the benefit of our weight calculation method. For the same scheduling method, better result was obtain when weights are calculated by equation (2) (-Mix) than using Pc or Pd alone (–CPU and –Disk).

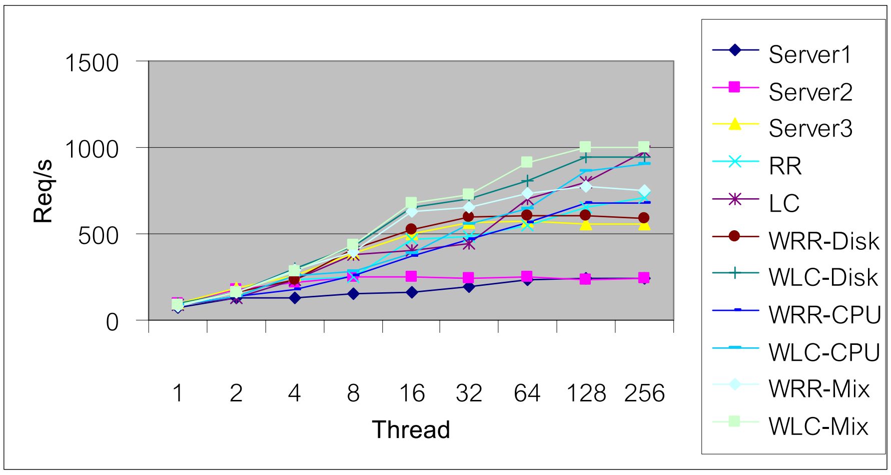

Figure 11: Experiment with 66.6% of static content and 33.3% of dynamic content

7. Conclusion

This paper has presented a summary of cluster web server architectures and revealed the need of proper weights for scheduling of the heterogeneous web cluster. A method for calculating weights from the ratio of dynamic and static web access statistics has been proposed. The method poses the advantage of requiring no prior knowledge of the web site content. The applicability and benefit of this method was validated by experiments. However, because the server CPU and disk performances have relatively slight effect for small static content files, the method can still be improved by taking into consideration the networking performance, and file size in the future.

Acknowledgements
This work is supported by 100th Anniversary Shell Scholarship Grant.

References

[1] Jim Reese, The Technology Behind Google , Atlanta Linux Showcase 2000 , available at http://technetcast.ddj.com/tnc_play_stream.html?stream_id=420, Oct 2000
[2] Edward Walker, "pWEB - A Parallel Web Server Harness", http://www.ihpc.nus.edu.sg/STAFF/edward/pweb.html, April, 1997.
[3] SWEB: Towards a Scalable WWW Server on MultiComputers by D. Andresen, T. Yang, V. Holmedahl and O. Ibarra. This paper gives the system organization and scheduling algorithms, and initial experimental results. in Proccedings of the 10th International Parallel Processing Symposium (IPPS'96) , Hawaii, April, 1996. Talk slides . The journal version in Journal of Parallel and Distributed Computing, 1997.
[4] Wensong Zhang, Shiyao Jin, Quanyuan Wu, "Creating Linux Virtual Servers", *On May 20, 1999 LinuxExpo'99*, 1999 available at http://www.linux-vs.org
[5] Tim Bray, "Benchmark which measures the performance of Unix file system operations", http://www.textuality.com/bonnie/, 1996
[6] ZDNet eTesting Labs, "Measure web server software performance" , http://www.etestinglabs.com/benchmarks/webbench/webbench.asp, 2002