Content-based Workload-aware Request Distribution Policies in Cluster-based Web Servers

Lian-Feng Guo, Mei-Ling Chiang, Shang-Yi Zhuang
Department of Information Management
National Chi-Nan University
Taiwan, R.O.C
E-mail: {s3213534, joanna, s1213018}@ncnu.edu.tw

ABSTRACT

How to efficiently dispatch requests from clients to web servers and effectively utilize system resource are crucial to the performance of a web cluster. In this paper, we have proposed new Content-based Workload-aware Request Distribution (CWARD) policies on cluster-based web servers. These policies take into account content in requests, workload information, and web object characterization in dispatching requests. Basically, the more frequently web objects are accessed, the more servers will serve the requests for these web objects. The experimental results of practical implementation on Linux show that our proposed Content-based Workload-aware Request Distribution policies are efficient. The trace-driven benchmarking with real-world traces of ClarkNet and WorldCup98 demonstrates that our proposed CWARD policies can effectively improve web cluster performance. Especially, when using the proposed CWARD policies, web clusters with content-aware dispatching such as LVS-CAD and LVS-CAD/FC can achieve up to 94% better performance than the Linux Virtual Server (LVS) web cluster with content-blind dispatching.

1. INTRODUCTION

Because of the rapid growth of Internet users, more powerful web servers have to be adopted to deal with the large amount of HTTP requests. To build a more powerful web server, the cluster-based web systems have been widely adopted because of the implicit advantages of load sharing or balancing, scalability, and high availability. One of the popularly used cluster servers is Linux Virtual Server (LVS) which is composed of one request dispatching front-end server and several request handling back-end real servers.

Many researches [1, 5, 7, 9, 11] have focused on the study of content-aware request distribution in which the request dispatching server mainly bases on the content of request packets (i.e. URL information) to dispatch requests. Some researches further combine the use of the content of requests and load information, or even the workload information, to design their intelligent request dispatching mechanisms. Their research results all show that web clusters will be more efficient in handling requests. Content-based request distribution also enables the partitioning web contents, building specialized web services among web servers, or maintaining session integrity.

With the aims to achieve better cache hit rates to reduce disk access and better cluster resource utilization while maintaining load balancing among back-end real servers, we have proposed several new content-based request distribution policies named Content-aware Workload-based Request Distribution policies (CWARD). These proposed policies consider content in requests, workload information, and web object characterization in dispatching requests. Basically, the more frequently web objects are accessed, the more servers will serve the requests for these web objects.

To evaluate the effectiveness of our proposed CWARD policies, we have implemented them in two content-aware dispatching web clusters, i.e. LVS-CAD [2] and LVS-CAD/FC [11], which are based on LVS while implementing TCP Rebuilding [2] mechanism to provide content-aware request distribution.

In the experiments, the trace-driven benchmarking with real-world traces of ClarkNet [8] and WorldCup98 [8] demonstrates that our proposed CWARD policies effectively improve web cluster performance. LVS-CAD and LVS-CAD/FC with CWARD policies can achieve up to 94% better performance than LVS with content-blind dispatching.

2. BACKGROUND AND RELATED WORK

This section introduces the content-blind dispatching platform - Linux Virtual Server, two content-aware dispatching platforms – LVS-CAD and LVS-CAD/FC, and existing content-based request distribution policies.

2.1. The Content-blind Dispatching Web Cluster – LVS

Linux Virtual Server (LVS) [4] is a highly scalable and available cluster built by a set of real servers, based on Linux. The LVS consists of a front-end server for dispatching and routing requests from clients to back-end real servers according to the designated routing mechanisms and request scheduling algorithms. With the direct routing mechanism, the back-end real

server processes these packets from the front-end server and returns results directly to client.

2.2. THE CONTENT-AWARE DISPATCHING WEB CLUSTERS – LVS-CAD AND LVS-CAD/FC

LVS-CAD [2] (i.e. LVS with Content-Aware Dispatching) is based on LVS while implementing TCP Rebuilding [2] mechanism to provide content-aware dispatching. It develops a one-way kernel-level layer-7 front-end for examining the content of HTTP requests and analyzing workload characterization in dispatching requests from clients to servers. LVS-CAD/FC [11] (i.e. LVS with Content-Aware Dispatching and File Caching) extends LVS-CAD with web object caching mechanism, in which a small amount of server RAM is dedicated to cache the most frequently accessed web objects to reduce disk access.

TCP Rebuilding [2] proposed in our previous study is a light-weight TCP connection transfer technique that enables a web cluster to be content-aware. The TCP Rebuilding technique allows the front-end server to transfer an established connection with a client to a chosen back-end real server by rebuilding the TCP connection in the back-end real server. After the TCP connection has been built, the chosen back-end real server could respond to the request of the client directly, bypassing the front-end server. In particular, TCP Rebuilding could rebuild the TCP connection at one back-end real server using only the original HTTP request packet and no extra packets for connection transfer are required. This mechanism can be applied to build a content-aware platform for developing more complex dispatching policies. Therefore, TCP Rebuilding is adopted to construct the content-aware platform in this research.

2.3. EXISTING CONTENT-BASED REQUEST DISTRIBUTION POLICIES

To improve cache hit rates, Locality-Aware Request Distribution (LARD) [9] dispatches the requests to the same web object to the same back-end server that will most likely have the web object in RAM, unless the back-end server is overloaded.

Workload-aware Request Distribution strategy (WARD) [5] takes workload properties into account in dispatching requests. It identifies a small set of most frequent web objects, called core. The requests to core web objects are processed by any server, while the rest of the web objects, called part, are partitioned to be served by different servers.

In our previously proposed Content-based Workload-aware Request Distribution with Core Replication (CWARD/CR) [11] strategy, the web cluster uses a small amount of memory dedicated to cache a small set of most frequently accessed web objects, called core. The request to core web objects are processed by any back-end real server's RAM in a web cluster, while the rest of the web objects are partitioned to be served by different back-end real servers' RAM. Hence, a small set of most frequently accessed web objects are pre-fetched into back-end real servers' RAM to increase the performance of the whole web cluster.

Client-aware Dispatching Policy (CAP) [1] classifies requests from clients into four classes, namely normal, CPU-bound, disk-bound, and disk- and CPU-bound. When a HTTP request packet arrives, the web switch distinguishes which class it belongs to, and schedules the packet in this class with the Round-Robin manner.

3. PROPOSED CONTENT-BASED WORKLOAD-AWARE REQUEST DISTRIBUTION POLICIES

Previous studies on content-aware request distribution [7, 9] have shown that policies distributing the requests from clients based on the content of requests lead to performance improvement compared with the strategies taking into account only load balancing. Therefore, content-aware request distribution is considered in this research.

For a web cluster that consists of one request-dispatching front-end server and several request-handling back-end real servers, since all requests from clients are sent to the front-end server, if the front-end server is content-aware and is able to examine the content of HTTP request, it can easily collect the reference information of web objects and determine the most frequently accessed web objects, and then decide which back-end real server should serve the request.

According to these considerations, we use the web clusters, LVS-CAD (i.e. LVS with Content-Aware Dispatching) [2] and LVS-CAD/FC (i.e. LVS with Content-Aware Dispatching and File Caching) [11] as our research platforms to study the content-aware request distribution policies. They all use a kernel-level layer-7 front-end server that examines the content of request (i.e. URI) and analyzes workload characteristic when dispatching requests from clients.

In fact, not only content of request (i.e. URI) and workload information are important factors in designing content-aware request distribution policies, web object characterization such as file size is also helpful. Therefore, we propose new strategies related to Content-based Workload-aware Request Distribution: Frequency-based Replication (CWARD/FR), Frequency-based Sized Replication (CWARD/FSR), Partitioned Frequency-based Replication (CWARD/PFR), and Partitioned Frequency-based Sized Replication (CWARD/PFSR).

3.1. FREQUENCY-BASED REPLICATION – CWARD/FR

The basic idea of CWARD/FR is to let web objects cached in back-end real servers according to their access frequencies, such that the more frequently web objects

are accessed, the more back-end real servers can serve these frequently accessed web objects.

Since CWARD/FR takes workload properties into account, CWARD/FR policy first determines the number of back-end real servers that can serve a specific web object according to equation 1. In the equation 1, let α_i denote the number of times the specific web object *i* was accessed. This access information can be easily collected by the front-end server since all requests from clients are sent to it. Besides, the total amount of back-end real servers is assumed to be γ. The number of the back-end real servers that can serve this web object *i* is λ_i. The total amount of web objects served in web cluster is *n*. In our experiment described in Section 4, the value of access times is normalized by the $\log_{10}$ function.

$$\frac{\alpha_i}{\max_{i=1}^{n}\{\alpha_i\}} * \gamma = \lambda_i \qquad (1)$$

After the number of back-end real servers that can serve a specific web object *i* is determined, then the front-end server uses the Round-Robin manner to assign a set of back-end real servers for serving each web object *i*.

The proposed CWARD/FR policy is illustrated in Figure 1. The requests for the same web object will be distributed to one of the set of back-end real servers that most likely have the web object cached in the main memory according to the designated request scheduling algorithm.

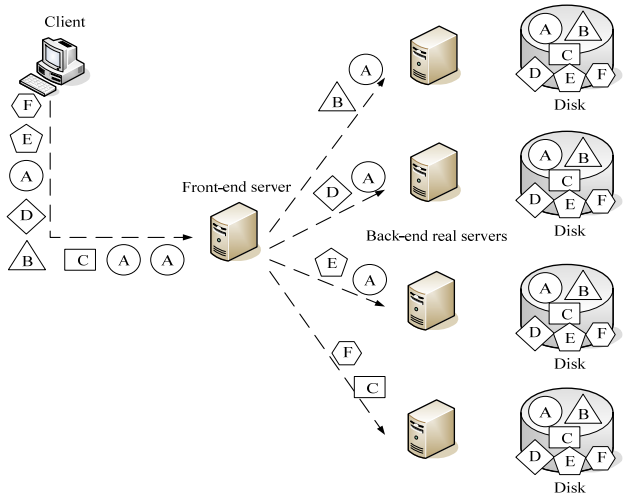

Figure 1: CWARD/FR Policy

When the web object caching is used in the web cluster such as LVS-CAD/FC, the proposed CWARD/FR policy with web object caching can work in the same way as the basic CWARD/FR does. The most frequently accessed web objects will be cached/pre-fetched in the most back-end real servers; whereas, the lesser frequently accessed web objects will be cached/pre-fetched in the lesser amount of back-end real servers.

Figure 2 illustrates the principle of CWARD/FR strategy with web object caching in a web cluster that contains a front-end server, four back-end real servers, and six web objects (A, B, C, D, E, F) in the incoming request stream. In Figure 2, γ is assumed to be four and λ can be one to four. For example, λ equal to four means that the web object should be cached/pre-fetched into four back-end real servers' RAM.

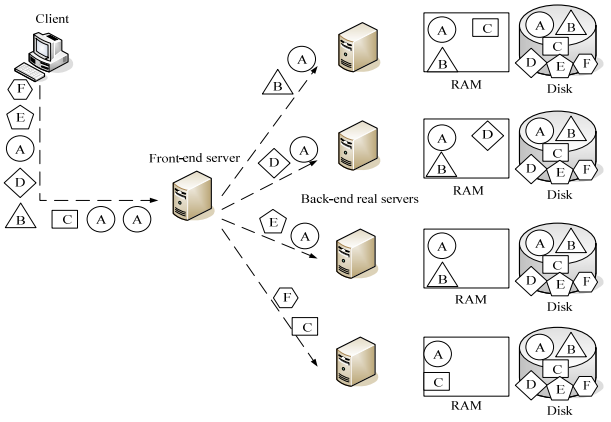

Figure 2: CWARD/FR Policy with Web Object Caching

The CWARD/FR policy with web object caching is detailed as follows. First of all, because frequently accessed web objects must be pre-fetched into back-end real servers' RAM, λ in the equation 1 is calculated to decide the amount of back-end real servers by which the web object can be served. For example, if the web object A in Figure 2 is the hottest web object, A is pre-fetched into λ back-end real servers' RAM. When the front-end server receives a sequence of the requests, it will examine the content of the HTTP request. If the requested content belongs to the web object pre-fetched, e.g., the web object A in Figure 2, the front-end server will modify the URI in the content of the HTTP request to appropriate path for routing the HTTP request to the designated back-end real server's cache according to the traditional Round-Robin strategy. Secondly, if the requested content is not pre-fetched, e.g., the web objects E or F in Figure 2, the front-end server routes the HTTP request to the back-end real server according to the designated request scheduling algorithm.

3.2. FREQUENCY-BASED SIZED REPLICATION – CWARD/FSR

According to the traffic characterization of a modern web site [6], large web objects (up to 64 KB) make up only 0.3% of the working set but consume 53.9% of required storage space. In addition, these large web objects occupy only 0.1% of all client requests. Thus, full replication of these web objects is not cost-effective. It is better to cache the frequently accessed and smaller sized web objects.

According to this consideration, we propose another Content-based Workload-aware Request Distribution

policy named Frequency-based Sized Replication (CWARD/FSR). Similar to the CWARD/FR strategy, the basic idea is to let web objects cached in back-end real servers according to their access frequencies. The difference is that the most frequently accessed and smaller sized web objects are replicated in the most of back-end real servers; whereas, the lesser frequently accessed and larger sized web objects are replicated in the lesser amount of back-ends.

Therefore, CWARD/FSR is an extension of CWARD/FR. In addition to the access times, CWARD/FSR also takes the file size of a web object into account in determining the number of back-end real servers to serve the web object according to equation 2. In equation 2, β_i denotes the corresponding file size of the specific web object *i*. In our experiment described in Section 4.3, the value of access times divided by file size is also normalized by the $\log_{10}$ function.

$$\frac{\frac{\alpha_i}{\beta_i}}{\max\left\{\frac{\alpha_i}{\beta_i}\right\}} * \gamma = \lambda_i \quad (i = 1, ..., n) \qquad (2)$$

3.3. PARTITIONED FREQUENCY-BASED REPLICATION – CWARD/PFR

If some frequently accessed web objects are extremely large size, most of the back-end real servers would cache these web objects under the proposed CWARD/FR policy described in Section 3.1, which would occupy the large amount of back-end real servers' caches. Thus, replication of caching these large-sized web objects is not cost-effective.

To address this problem, we propose another Content-based Workload-aware Request Distribution policy named Partitioned Frequency-based Replication (CWARD/PFR). The CWARD/PFR strategy is designed to combine the advantages of CWARD/FR and LARD policies. Basically, it partitions the working set into two object groups: the group with large-sized web objects and the group with small-sized web objects. The threshold to determine whether the size of a web object is large is set to the average size of web objects in our experiment.

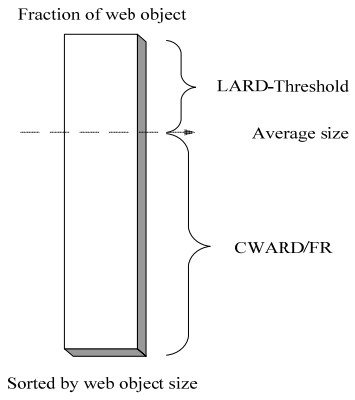

Figure 3: CWARD/PFR Policy

Figure 3 shows how the CWARD/PFR works. At first, web objects are sorted by their sizes. The front-end server examines the requests from clients and determines which group the requests belong to. Then the requests belonging to the group with small-sized web objects will be distributed to the chosen back-end real server according to CWARD/FR policy. Otherwise, LARD [9] policy will be used.

For web objects belonging to the small-sized group, the most frequently accessed web objects will be cached/pre-fetched in the most back-end real servers, whereas the lesser frequently accessed web objects will be cached/pre-fetched in the lesser amount of back-end real servers, according to CWARD/FR policy. This is intended to achieve load balancing of a web cluster because the replication of small-sized web objects will not occupy the large amount of back-end real servers' caches. However, the group with large-sized web objects is cached/pre-fetched in back-end real servers' caches according to LARD with Threshold (LARD-Threshold) strategy, which is the variation of LARD [9]. The LARD-Threshold strategy uses a threshold to decide whether the web objects belonging to the group with large-sized web objects should be cached/pre-fetched at one back-end real server's cache.

3.4. PARTITIONED FREQUENCY-BASED SIZED REPLICATION – CWARD/PFSR

To take the file sizes of web objects into account and combine the advantages of CWARD/FSR and CWARD/PFR, we propose another Content-based Workload-aware Request Distribution policy named Partitioned Frequency-based Sized Replication (CWARD/PFSR). Similar to the CWARD/PFR strategy, web objects are partitioned into the group with large-sized web objects and the group with small-sized web objects. The difference is that requests from the clients belonging to the group with small-sized web objects are distributed to the chosen back-end real servers according to CWARD/PFR policy and the requests belonging to the group with large-sized web objects are distributed according to LARD-Threshold policy as showed in Figure 4.

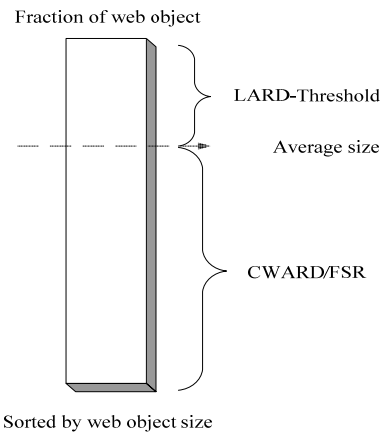

Figure 4: CWARD/PFSR Policy

4. PERFORMANCE EVALUATION

In this section, we present performance evaluation of our proposed Content-based Workload-aware Request Distribution policies under two content-aware dispatching platforms: LVS-CAD and LVS-CAD/FC clusters. LVS is used as the performance comparison.

4.1. EXPERIMENTAL ENVIRONMENT

The experimental environment consists of one front-end server, eight back-end real servers, and ten clients connected through a single 24-port Fast-Ethernet switch. The hardware and software environment are shown in Table 1. Each back-end real server is configured with 256MB and 512MB RAM in the experiments with ClarkNet trace [8] and WorldCup98 trace [8], respectively. The request routing mechanism is set to be direct routing [4] and the request scheduling algorithm used is set to be Weighted Round Robin [4].

Table1: Hardware/Software Environment

Item	Front-end	Back-end	Client
Processor(MHz)	Intel P4 3.4G	Intel P4 3.4G	Intel P4 2.4G
Memory (MB)	DDR 256/512	DDR 256/512	DDR 256
NIC (Mbps)	Intel Pro 100/1000	Intel Pro 100/1000	Reltek RTL8139
OS	Red Hat Linux 8.0	Red Hat Linux 8.0	Red Hat Linux 8.0
Kernel	2.4.18	2.4.18	2.4.18-14
IPVS	1.0.4	X	X
Web Server	X	Apache 2.0.40	X
Benchmark	X	X	http_load
Number of PCs	1	8	10

4.2. ACCESS LOG

We use publicly obtainable traces from the Internet Traffic Archive [8]: ClarkNet and WorldCup98. The working sets used in the research are derived from these two logs. Because the WorldCup98 trace contains a huge amount of requests, so we use only six hours' requests on the day July 12, 1998 from AM 09:00 to PM 03:00.

Because the degree of locality of reference in ClarkNet trace is high, so when caching 18% of most frequently accessed web objects, it could achieve 90% of cache hit ratio. The degree of locality of reference is also high and file size is quite large in WorldCup98 trace. When caching 20% of most frequently accessed web objects, it could achieve 81% of cache hit ratio and occupy 61% of file size.

4.3. EXPERIMENTAL RESULTS

This purpose of experiments in this section is to evaluate the effect of the proposed CWARD policies. In Figures 5 and 6, the CWARD/CR (core5%/part4%) means that LVS-CAD and LVS-CAD/FC clusters use CWARD/CR (i.e. CWARD with Core Replication) policy [11], in which the core set (i.e. the most frequently accessed web objects) has 5% of working set web objects, and the part set (i.e. the less frequently accessed web objects) in each back-end real server has 4% of working set web objects. In this experiment, LVS-CAD/FC's web object cache is set to occupy 15.63% of system RAM and the same amount of web objects are pre-fetched into each back-end real server's RAM before the web cluster is evaluated.

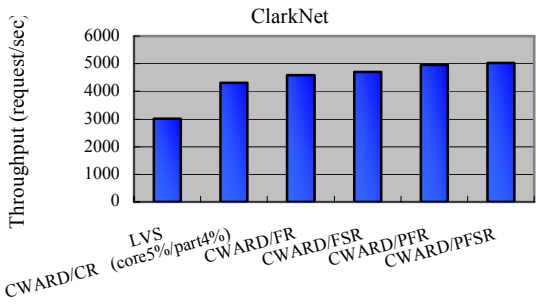

Figure 5: Comparison of Various Policies under LVS-CAD/FC (ClarkNet Trace)

The performance gain of our proposed CWARD policies is obviously large. Figure 5 shows that the performance of LVS-CAD/FC with various CWARD policies is 43-66% better than LVS. In Figure 6, though web objects are not pre-fetched into each back-end real server's RAM, LVS-CAD still performs 1-7.6% better than the LVS.

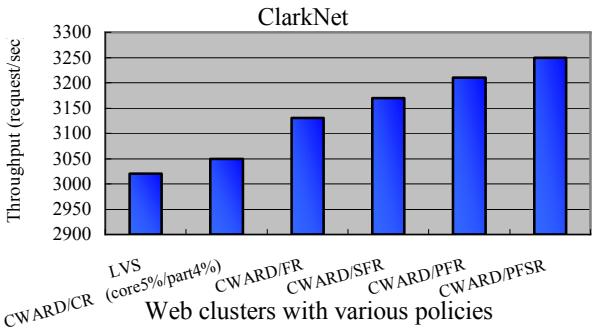

Figure 6: Comparison of Various Policies under LVS-CAD (ClarkNet Trace)

In Figures 7 and 8, the WorldCup98 trace is used in the experiment. Because file size is quite large in WorldCup98 trace, when caching 20% of the most frequently accessed web objects, it could achieve 81% cache hit ratio and occupy 61% file size. Because most frequently accessed and large-sized web objects could occupy the huge amount of back-end real servers' cache, so CWARD/PFR and CWARD/PFSR are used in evaluating the web cluster performance. The CWARD/PCR means that the working set is partitioned into two groups. The group with large-sized web objects

uses the LARD-Threshold policy and the group with small-sized web objects uses the CWARD/CR (core3%/part1%) in which the core set has 3% of working set web objects, and the part set in each back-end real server has 1% of working set web objects.

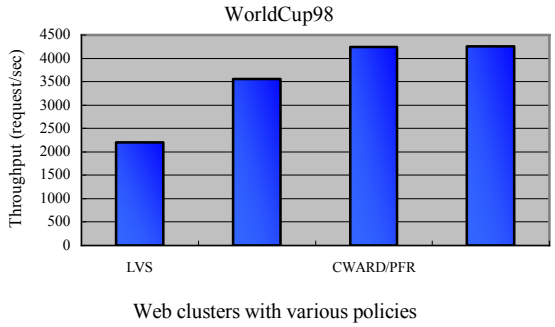

Figure 7: Comparison of Various Policies under LVS-CAD/FC (WorldCup98 Trace)

In Figure 7, LVS-CAD/FC's web object cache is set to occupy 60% of system RAM and web objects of approximately 18% of total web object size are pre-fetched into each back-end real server's RAM before the web cluster is evaluated. The performance gain of our proposed CWARD policies is quite large. As shown in Figure 7, the performance of our LVS-CAD/FC with various CWARD policies is 61-94% better than the LVS.

In Figure 8, though web objects are not pre-fetched into each back-end real server's RAM, the performance of our LVS-CAD with various CWARD policies is still 27-32% better than the LVS.

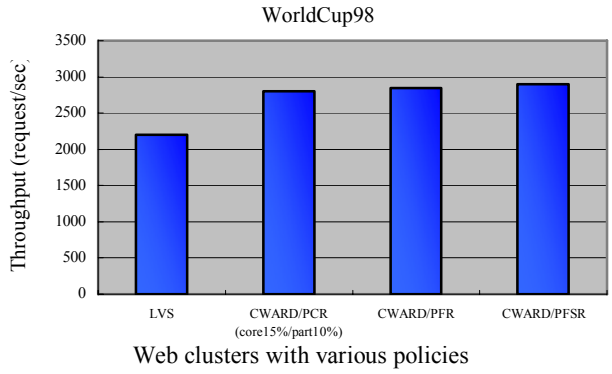

Figure 8: Comparison of Various Policies under LVS-CAD (WorldCup98 Trace)

5. CONCLUSION

With the aims to achieve better load balancing among servers, efficient memory usage, and better cache hit rates to reduce disk I/O in a web cluster, we have proposed new Content-based Workload-aware Request Distribution policies which consider content in requests, workload information, and web object characteristic in distributing requests of web objects to back-end real servers. The basic idea is to let web objects cached in back-end real servers according to their access frequencies and file sizes, such that the more frequently accessed and smaller sized web objects can be served among the more back-end real servers. Basically, the more frequently web objects are accessed, the more back-end real servers will serve the requests for these web objects.

We have implemented the proposed policies in two content-aware dispatching web clusters on Linux. Experimental results show that our LVS-CAD/FC with the proposed CWARD policies are efficient and can achieve up to 94% better performance than the layer-4 LVS web cluster. Our LVS-CAD with the proposed CWARD policies can still outperform LVS by 32%.

Based on this research and our content-aware dispatching platforms, several issues could be further explored, such as support of quality of service, adaptive content-aware dispatching algorithms, and efficiently support of dynamic web contents.

REFERENCES

[1] Emiliano Casalicchio and Michele Colajanni, "A Client-Aware Dispatching Algorithm for Web Clusters Providing Multiple Services," Proc. of 10th Int'l World Wide Web Conf., Hong Kong, pp. 535-544, May 1-5, 2001.

[2] H. H. Liu and Mei-Ling Chiang, "TCP Rebuilding for Content-aware Request Dispatching in Web Clusters," Journal of Internet Technology, Vol. 6, No. 2, pp. 231-240, April 2005.

[3] Http_load, http://www.acme.com/software/http_load/.

[4] Linux Virtual Server Website, http://www.linuxvirtualserver.org/.

[5] Ludmila Cherkasova and Magnus Karlsson, "Scalable Web Server Cluster Design with Workload-Aware Request Distribution Strategy WARD," In Proceedings of the 3rd International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems, San Jose, CA, pp. 212-221, June 2001.

[6] M. Arlitt and T. Jin, "Workload Characterization of the 1998 World Cup Web site," Hewlett-Packard Technical Report, February 1999.

[7] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel, "Scalable content-aware request distribution in cluster-based network servers," In Proceedings of the USENIX 2000 Annual Technical Conference.

[8] The Internet Traffic Archive Website, http://ita.ee.lbl.gov/.

[9] V. S. Pai, et al, "Locality-Aware Request Distribution in Cluster-based Network Servers," Eighth International Conference on Architectural Support for Programming Languages and Operating Systems, San Jose, CA, Oct. 1998.

[10] Valeria Cardellini, Emiliano Casalicchio, Michele Colajanni, and Philip S. Yu, "The State of the Art in Locally Distributed Web-Server Systems," ACM Computing Surveys, Vol. 34, No. 2, pp. 263-311, June 2002.

[11] Yu-Chen Lin, Mei-Ling Chiang, and Lian-Feng Gu, "System Support for Workload-aware Content-based Request Distribution in Web Clusters," Journal of Internet Technology, Vol. 7, No. 3, 2006.