WUJNS
Wuhan University Journal of Natural Sciences

Vol.11 No.3 2006 585-590

Article ID:1007-1202(2006)03-0585-06

The Configuration Strategies on Caching for Web Servers

GUO Chengcheng, ZHANG Li, YAN Puliu

School of Electronic Information, Wuhan University, Wuhan 430072, Hubei, China

Abstract: The Web cluster has been a popular solution of network server system because of its scalability and cost effective ness. The cache configured in servers can result in increasing significantly performance. In this paper, we discuss the suitable configuration strategies for caching dynamic content by our experimental results. Considering the system itself can provide support for caching static Web page, such as computer memory cache and disk's own cache, we adopt a special pattern that only caches dynamic Web page in some experiments to enlarge cache space. The paper is introduced three different replacement algorithms in our cache proxy module to test the practical effects of caching dynamic pages under different conditions. The paper is chiefly analyzed the influences of generated time and accessed frequency on caching dynamic Web pages. The paper is also provided the detailed experiment results and main conclusions in the paper.

Key words: web servers; dynamic content caching; configuration management; replacement algorithm

CLC number: TP 393

Received date: 2005-09-06

Foundation item: Supported by the National Natural Science Foundation of China (90204008)

Biography: GUO Chengcheng (1961-), male, Professor, research direction: computer network and network communication. E-mail: netccg@whu.edu.cn

0 Introduction

The Web cluster has been a popular solution of network server system because of the scalability and of the cost effectiveness. By using the mechanism of the content-based request distribution, the cache configured in servers can result in increasing performance dramatically. This scheduling policy can improve the hit rates in the back-end's main memory caches by distributing requests based on cache affinity[1,2]. Therefore, caching is an effective way to achieve scalability and flexibility by enabling the use of a partitioned server database and specialized server nodes. Because of the dynamic content objects are used increasingly, it is becoming more important to set up and manage a dynamic page caching for Web server. Currently, most of researches for dynamic content are focused on how to implement the cache proxy and how to guarantee the cache consistency[3-5].

Unlike traditional caching in memory, Web caches are required to manage objects of variable size and frequency[6], and the characterizations of Web access patterns are found to be Zipf-distribution in popularity[7]. In order to get optimal effect, the cache replacement algorithms have to take many factors into account, e.g., cost, size, frequency[8]. greedy dual-size (GDS) algorithm[9] considers the properties of both variability in Web objects size and retrieval cost (miss penalty). Mix policy[10] takes into account the network latency, the size of the documents, their access frequencies and the time elapsed since the last reference to documents in the cache. lowest relative value (LRV) algorithm[11] uses the cost, size, and the last access time of an object to calculate a utility value. The calculation is based on extensive empirical analysis.

greedy dual-size with frequency (GDSF) algorithm[12] simply incorporates access count into GDS. Popularity-aware greedy dual-size (GDSP) algorithm[13] takes in consideration the knowledge of the skewed popularity profile of Web objects. Least normalized cost replacement for the Web with updates (LNC-R-W3-U) algorithm[14] incorporates the cost of cache consistency maintaining and replacement mechanism, it aims at minimizing response time.

In this paper, we introduce our caching management policies on dynamic pages for Web servers. There are three different replacement algorithms in our caching proxy system that is a module for Apache Web server. We compare these algorithms by practical testing, and conclude the configuration strategies based on our experiment results.

1 Design of Replacement Policy

There is obvious difference on the generated time among dynamic Web pages, but there is almost no difference in the delay of transferring one page from Web server to cache proxy because the sizes of all the dynamic pages are approximate. Therefore, the primary penalty of retrieval a dynamic page is its generated time in back-end server. So, we designed a generated-time based replacement policy (GTime) for Web page caching, the Web pages which consume more system resource will be saved to reduce system response time. Its profit function $P(p,i)$ should be calculated as:

$$P(p,i) = \frac{G(p)}{S(p)} \cdot F(p,i) \qquad (1)$$

where, p is a Web page and i is a time section, $G(p)$ is the generated time of p, $S(p)$ is the size of p, $F(p, i)$ is the access frequency of p during ith time section.

As the exiting of 'peak' phenomenon in which the access frequency is very high, it needs to divide the whole process into several time sections and the access frequency of each Web page is respectively computed for each time section. It is more reasonable to remove Web pages with the 'fall sharply' trend in access frequency.

However, it is not enough to evaluate a Web page's future popularity only in terms of the access frequency in current time section. Therefore, we designed popularity-aware generated time based replacement policy (GTP). By predicting the access frequency and ignoring the page's size, the GTP algorithm is more adaptive for dynamic page caching. Its profit function should be calculated as:

$$P(p,i+1) = G(p) \cdot F(p,i+1) \qquad (2)$$

where, $P(p, i+1)$ is the access frequency prediction of Web page p in the next $(i+1)$ time section.

To calculate $F(p, i+1)$, We use the Adaptive Single Exponential Smoothing algorithm[15] in which the parameter of tracking signal is an key for this predicting process. Once the tracking signal is larger than a threshold, it means that the change of access frequency has burst during the current time section and there is larger warp value between prediction and practice. Therefore, this method is suitable to predict the dramatically changed 'peak' curve on line.

Let $F'(i)$ be the actual access frequency of page p in the ith time section, $F(i)$ is its predicted value, and the initial value of $F(i)$ equal to 0. The steps to compute the predicted $F(i+1)$ of Web page p as follows:

① To calculate the predicted error, $E(i)=\beta \cdot e(i)+(1-\beta) \cdot E(i-1)$, where $e(i)=F'(i)-F(i)$, $\beta=0.1$;

② To calculate the absolute value of predicted error, $M(i)=\beta \cdot |e(i)|+(1-\beta) \cdot M(i-1)$;

③ To calculate the tracking signal $T(i)=\frac{E(i)}{M(i)}$, obviously $|T(i)| \leqslant 1$;

④ To calculate the smoothing parameter, $\alpha(i) = |T(i)|$;

⑤ To predict the access frequency.

$$F(i+1) = \alpha(i) \cdot F'(i) + (1-\alpha(i)) \cdot F(i) \qquad (3)$$

In the above procedure, if the difference between $F(i)$ and $F'(i)$ is increased, $\alpha(i)$ will relatively become larger. Sequentially, it will make $F(i+1)$ change more quickly. Once $F(i+1)$ has adapted to this change, $\alpha(i)$ will become smaller to counteract those random changes.

2 Experiment Environment

The front-end dispatcher of Web servers is a computer with P-III800Mb/sHz CPU, 256 MB memory and 40 GB hard disk. It runs the content-based request distribution program on Linux OS, whose maximum distributing rate can be over 2 000 TCP connections per second.

The cluster back-end servers are 2 computers with P-IV1.7Gb/sHz CPU, 512 MB memory, and 40 GB IDE hard disk. The servers run Apache loaded DCMC module[16]. This module builds a cache space in memory to cache the requested Web pages, and provides access

proxy service. We do not take the cooperation among several DCMCs into account. Thus the simplicity and facility of DCMC can be guaranteed.

The clients are 4 computers with P-IV2. 4Gb/sHz CPU, 256 MB memory and 80 GBIDE hard disk. They run WebBench5.0 testing program on the client.

The network device is a switch of Cisco2924 (100 Mb/s). The dispatcher, servers and clients are all connected through the same switch.

In such a configuration, it can be guaranteed that no device among the network and dispatcher and clients would be the bottleneck of cluster system during the testing process, thus we can obtain the real effect of cache system in servers.

By the simulation program, we produce the static Web pages with 3 types of size: 5, 50 and 500 kB, which conform the Parato distribution, and we produce the dynamic Web pages with 3 types of generated time: 0.05, 0.5, 5 s, which conform the negative exponential distribution.

Total 200 Web pages are used in our experiments. The ratio between static Web pages and dynamic Web pages are assigned in terms of the experiment demand. The requests of clients are generated randomly. All these Web pages are evenly distributed on 2 servers according to their workload.

3 Experiment

For our experiments, the number of clients increases gradually from 1 to 40 procedures, each procedure sets up 10 threads. The testing process lasts for about 30 min, and we take 5 min as a time section to calculate the access frequency. The value is the average value of three samplings.

3.1 Experiment 1

Experiment 1 tests the cluster performance under conditions that two different scenes: all Web pages are cached and no any Web pages are cached. The testing results are shown as Fig. 1.

As shown in Fig. 1, even if all static Web pages are cached, it gets almost the same performance as that without caching. The reason is that the file management system in the kernel of OS can provide effectively the caching to the static Web pages. Therefore, it is reasonable to presume a worse performance if only caching part of the static Web pages by DCMC, which is a program in application layer.

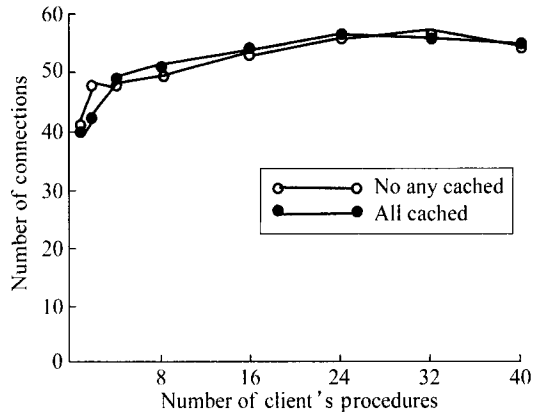

Fig. 1 Comparison of non-caching and all-caching

3.2 Experiment 2

Experiment 2 tests the performance of Web servers on caching both static and dynamic Web pages. We also compare the results of 3 replacement policies, GDSF, GTime and GTP. Among the requests, 70% are for static Web pages and 30% for dynamic ones in this experiment.

Figures 2 and 3 show the results of 3 replacement algorithms under conditions that cache size respectively is 30% or 60% of the total size of all pages.

Figure 2 shows that, when requests are primarily for static Web pages, GDSF and GTime get better effect than GTP. The reason may be that GTP does not consider the size of cached Web pages, and the size of Web pages is important to keep the hit ratio for the static Web pages. Figure 3 shows that the influence of Web page's size becomes smaller with the increase of cache space, and the difference are not obvious. Another noticeable phenomenon is that, as Fig. 3 shows, although the cache space doubles, the effects of caching improves lesser. This reason is that, when the requests are mainly for static Web pages, the extra cache for static Web pages is not as efficient as system's own cache. The result of experiment 1 also shows this point.

3.3 Experiment 3

Experiment 3 tests the performance of Web servers on only caching dynamic Web pages. We also compare the results of 3 replacement algorithms, GDSF, GTime and GTP.

The process of the experiment is similar to experiment 2. Figure 4 shows the effect of the 3 algorithms when cache size is 30% of total dynamic pages' size, which is about 9% of all Web pages. Figure 5 shows the

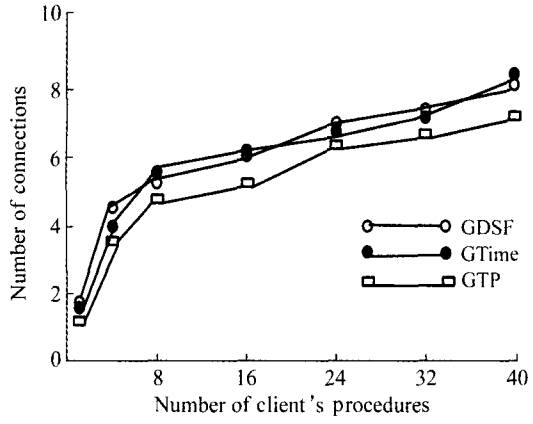

Fig. 2 Effects of caching when cache size is 30% of total pages size

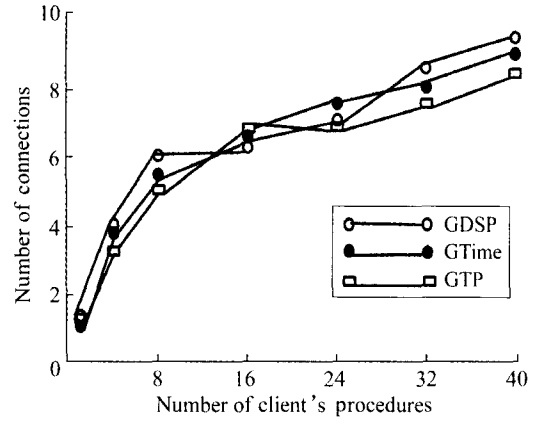

Fig. 3 Effects of caching when cache size is 60% of total pages size

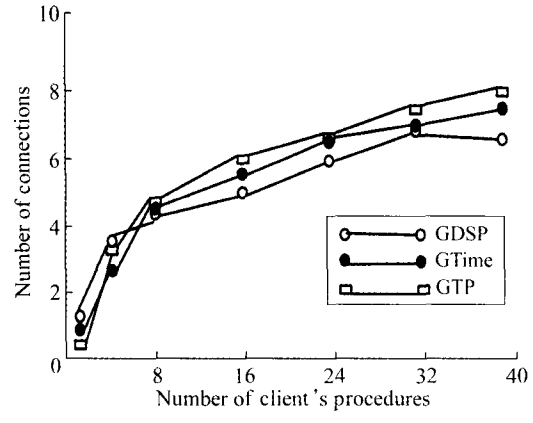

Fig. 4 Effects of caching when cache size is 30% of total dynamic pages size

effect when cache size is 50% of total dynamic pages' size, which is about 15% of all Web pages.

Figure 4 and Figure 5 show that, when DCMC module only caches dynamic Web pages, both GTP and Gtime are better than GDSF. It means that the generated time of dynamic page is the most important for the replacement algorithm. However, GTP is just a little better than GTime and in Fig. 4 it is more obvious. Figure 6 shows the statistics of the hit rate (H) in Fig. 4's experiment. Because the requests are mainly for static pages and DCMC is configured not to cache static pages, the hit rate is a little low. Figure 6 shows that GTP's hit rate is two times higher than the other 2 algorithms. But because the proportion of dynamic pages in requests is small, the performance improvement in Fig. 4 is not so great.

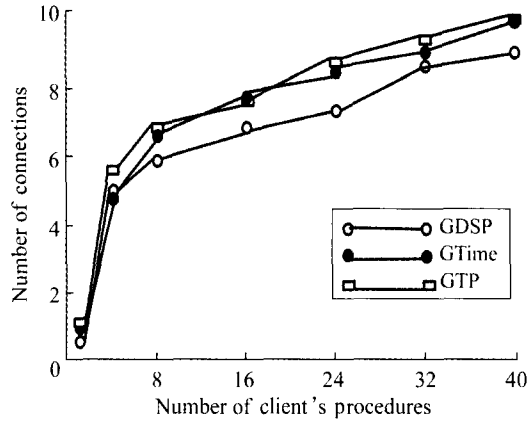

Fig. 5 Effects of caching when cache size is 50% of total dynamic pages size

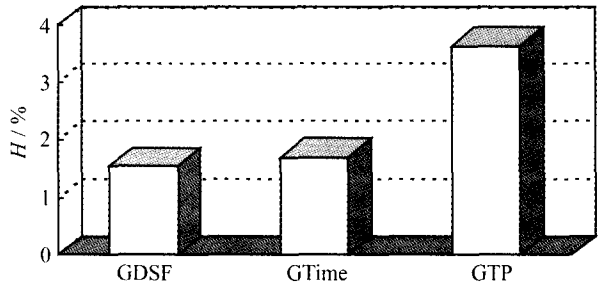

Fig. 6 Hit rate of caching in figure 4's experiment

By comparing Fig. 4 with 2, as well as Fig. 5 with 3, they get almost the same effect, while only caching dynamic pages uses much less memory resources than caching both static and dynamic Web pages. When the cache space is the same, therefore, the Web servers may get better system performance if it only caches dynamic Web pages, that is, it will get faster average response time.

3.4 Experiment 4

Experiment 4 tests the performance of Web servers with multi-copy of Web pages. For the experiment, we also use the generated 200 documents, in which the proportion of static pages is 70%. The cache size of DCMC is configured respectively as 30% of the total pages' size and as 30% of all dynamic pages' size. Unlike former

experiments, the 200 documents are placed in each server to form the complete replication of each page. Because the number of pages doubles in a single server, the cache size relatively reduces half of former. The dispatcher of the Web servers distributes the requests in round Robin manner.

Figure 7 and Figure 8 show respectively the effect when both static and dynamic pages are cached together and dynamic pages are cached only.

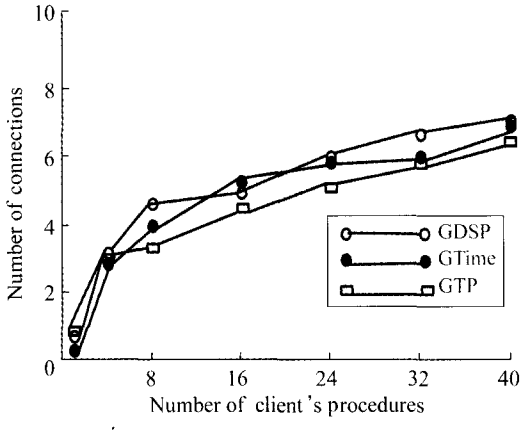

Fig. 7 Static and dynamic pages are cached together when cache size is 30% of total pages

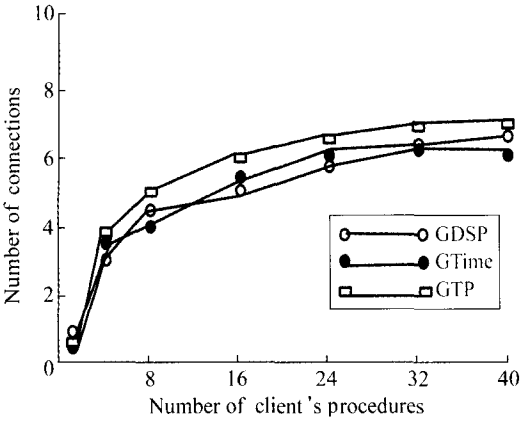

Fig. 8 Only dynamic pages are cached when cache size is 30% of total dynamic pages

By comparing Fig. 7 with 2, as well as Fig. 8 with 4, the complete replication of each page does not always get better effect (even get lower effect) than that without copy. This is because, although the replication on multiple servers can enhance the parallel process ability, the cache size in single server is reduced correspondingly, and the delay of route dealing in dispatcher is increased. These prevent the improvement of the whole system's responding performance.

3.5 Experiment 5

Experiment 5 tests the performance of Web servers when all the dynamic pages are cached in DCMC. The distribution of Web pages, experiment steps and conditions are the same as Experiment 2. The cache size of each server is configured equal to the size of all the dynamic Web pages on it. We test the system's responding performance when the requested ratio of static pages and dynamic pages are 7 : 3, 5 : 5, 3 : 7 respectively. Figure 9 shows the testing results.

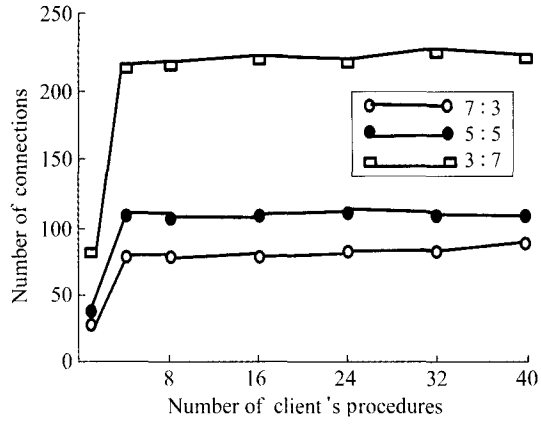

Fig. 9 Effects when caching all dynamic Web pages

Under condition that the ratio is 7 : 3, comparing the curve in Fig. 9 with figure 5's effect, although cache size only doubles, it leads the number of connections per second to increases about 8 times higher. As shown in Fig. 9, with the proportion of dynamic Web pages in requests increasing, the DCMC can produce super-linear increase on response capability. This means that the performance of Web servers can be improved dramatically by caching all the dynamic Web pages in back-end servers.

4 Conclusion

By the practical applications and the experiments above, our conclusions are as follows:

① The configuration of cache in cluster's back-end nodes is the first important thing we should consider. Because operating system in server can provide cache for static Web pages, just adding cache for dynamic Web pages in server is enough.

② Caching dynamic Web pages as many as possible, and caching all the dynamic Web pages can lead to the best result. As the price of memory is lower and lower, it's feasible to cache all dynamic Web pages in those Web sites mainly providing service on static Web pages.

③ Different cache replacement policies can gain different effect on performance of a cluster's, the generated

cost of a Web page is the most important points to build a high speed cache for Web servers, which includes many dynamic content objects. When the cache size is small, it is also a better choice to predict the access frequency of a Web page.

④ Replication distribution and cache space configuration should be considered together. Improving the ability on parallel process should not be at the cost of lower cache hit rate. It is worth of only replicating suitably these Web pages whom are accessed too continually.

References

[1] Hunt G, Nahum E, Tracey J. Enabling Content-Based Load Distribution for Scalable Services [R/OL] // Technical Report, IBM T J, Watson Research Center [2002-03-11]. *http:// www. research. ibm. com/ people/ n/nahum/ publications/ibm-tr97-cluster. pdf.*

[2] Pai V S, Aron M, Banga G, *et al*. Locality-Aware Request Distribution in Cluster-Based Network Servers [C/OL]// *Proceedings of the ACM Eight International Conferences on Architectural Support for Programming Languages and Operating System* [2002-03-20]. *http://citeseer. ist. psu. edu/article/ pai98localityaware. html.*

[3] Garg P K, Eshghi K, Gschwind T, *et al*. Enabling Network Caching of Dynamic Web Objects [C] // *Proceedings of the 12th International Conference on Computer Performance Evaluation, Modelling Techniques and Tools*. Berlin: Springer Heidelberg, 2002:329-338.

[4] Amiri K, Park S, Tewari R, *et al*. DBProxy: A Dynamic Data Cache for Web Applications [C]// *Proceedings of the 19th International Conference on Data Engineering (ICDE03)*. Bangalore: IEEE Computer Society, 2003:821-83.

[5] Challenger J R, Dantzig P, Iyengar A, *et al*. Efficiently Serving Dynamic Data at highly Accessed Web Site [J]. *IEEE/ ACM Transactions on Networking*, 2004, **12**:233-246.

[6] Arlitt M F, Williamson C. Internet Web Server: Workload Characterization and Implication [J]. *IEEE/ACM Transactions on Networking*, 1997, **5**:631-644.

[7] Barford P, Bestavros A, Bradley A, *et al*. Changes in Web Client Access Patterns: Characteristics and Caching Implications [J]. *WWW Journal*, 1999, **1**:3-16.

[8] Lin Yangwang, Zhang Dajiang, Qian Hualin. A Novel Replacement Algorithm for Web Caching [J]. *Journal of Software*, 2001, **11**:1710-1715 (Ch).

[9] Cao P, Irani S. Cost-Aware WWW Proxy Caching Algorithm [C/OL] // *Proceedings of USENIX Symposium on Internet Technology and System* [2003-07-08]. *http://citeseer. ist. psu. edu/cao97greedydualsize. html.*

[10] Niclausse N, liu Z, Nain P. A New Efficient Caching Policy for the World Wide Web [C/OL]// *Proceeding of the Workshop on Internet Server Performance (WISP '98)* [2003-07-08]. *http://www-sop. inria. fr/mistral/ personnel/ Nicolas. Niclausse/articles/wisp98/.*

[11] Rizzo L, Vicisano L. Repalcement Policies for a Proxy Cache [J]. *IEEE/ACM Transactions on Networking*, 1999, **2**: 158-170.

[12] Martin A, Ludmila C, John D, *et al*. Evaluting Content Management Techniques for Web Proxy Caches [C/OL]// *Proceedings of the 2nd Workshop on Internet Server Performance (WISP99)* [2004-01-22]. *http://citeseer. ist. psu. edu/arlitt99evaluating. html.*

[13] Jin Shudong, Bestavros A. Popularity-Aware Greedy Dual-Size Web Proxy Caching Algorithms [C] // *Proceedings of the 20th International Conference on Distributed Computing Systems*. New York: IEEE Press, 2000:254-261.

[14] Shim J, Scheuermann P, Vingralek R. Proxy Cache Algorithms: Design, Implementation and Performance [J]. *IEEE Transactions on Knowledge and Data Engineering*, 1999, **4**: 549-562.

[15] Wang Yongling. Calculation of Prediction [M]. Beijing: Science Press, 1986 (Ch).

[16] Liu Dan, Guo Chengcheng, Zhang li. Design and Implementation of a Dynamic Content Cache Module for Web Server [J]. *Wuhan University Journal of Natural Sciences*, 2004, **9**(5):828-834.

□