Dynamically Selecting Distribution Strategies for Web Documents According to Access Pattern

Wenyu Qu[†], Keqiu Li[††], Bo Jiang[††], Hong Shen[†], and Di Wu[†††]
[†]Graduate School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan
[††]College of Computer Science and Technology
Dalian Maritime University
No 1, Linghai Road, Dalian, 116026, China
[†††]Department of Computer Science and Engineering
Dalian University of Technology
No 2, Linggong Road, Dalian, 116024, China

Summary
Web caching and replication are efficient techniques for reducing Web traffic, user access latency, and server load. In this paper, we first propose an improved GreedyDual* (GD*) cache replacement algorithm, which considers update frequency as a factor in its utility function. Second, we present a group-based method for dynamically selecting distribution strategies for web documents according to access patterns. The documents are divided into groups according to access patterns and the documents in each group are assigned to the same distribution strategy. Our group-based model combines performance metrics with the different weights assigned to each of them. Finally, we use both trace data and statistical data to simulate our methods. The experimental results show that our improved GD*algorithm can outperform the existing GD* algorithm over the performance metrics considered, and our group-based method for document distribution strategy selection can improve several performance metrics, while keeping others almost the same.

Key words:
Web caching and replication, distribution strategy, cache replacement algorithm, simulation, trace data, autonomous system (AS)

1. Introduction

In recent years, the effective distribution and maintenance of stored information has become a major concern for Internet users, as the Internet becomes increasingly congested and popular web sites suffer from overloaded conditions caused by large numbers of simultaneous accesses. When users retrieve web documents from the Internet, they often experience considerable latency.

Web caching and replication are two important approaches for enhancing the efficient delivery of web contents, reducing latencies experienced by users. A user's request for a document is directed to a nearby copy, not to the original server; thus, reducing access time, average server load, and overall network traffic. Caching [8] was originally applied to distributed file systems. Although it has been well studied, its application on the Internet gave rise to new problems, such as where to place a cache, how to make sure that cached contents are valid, how to solve replacement problems, how to handle dynamic web documents, etc. Replication was commonly applied to distributed file systems to increase availability and fault tolerance [18]. Both techniques have complementary roles in the web environment. Caching attempts to store the most commonly accessed objects as close to the clients as possible, while replication distributes a site's contents across multiple replica servers. Caching directly targets minimizing download delays, by assuming that retrieving the required object from the cache incurs less latency than getting it from the web server. Replication, on the other hand, accounts for improved end-to-end responsiveness by allowing clients to perform downloads from their closest replica server.

Although web caching and replication can enhance the delivery efficiency of web contents and reduce response time, they also bring some problems, such as maintaining consistency of documents, propagating content updates to replica servers and caches, and so on. There are many ways to distribute copies of a web document across multiple servers. One has to decide how many copies are needed, where and when to create them, and how to keep them consistent. A good distribution strategy would be an algorithm that makes these decisions. We argue that there is no distribution strategy that is optimal for all performance metrics; in most cases, we have to pay the cost of making some performance metrics worse if we hope to make one or more of the others better. For cache replacement problems, there are numerous algorithms available. The GD* algorithm is one of the most efficient [13, 16, 17]. It considers both popularity and correlation, but it does not consider update frequency in its utility function. In this paper, we first propose an improved GD*

Manuscript revised January 2006.
Corresponding author: Keqiu Li (keqiu_01@163.com).

cache replacement algorithm based on the existing GD* algorithm. Second, we present a group-based method for dynamically selecting distribution strategies for web documents according to access patterns. We divide the documents into groups according to access patterns and assign the same distribution strategy to the documents in each group. Further, we present a group-based model that combines performance metrics with the different weights assigned to each of them. Therefore, our method can generate a family of strategy arrangements that can be adapted to different network characteristics. To realize our method, we use a system model in which documents can be placed on multiple Internet hosts. Clients are grouped based on the autonomous systems (ASs) that host them. ASs are used to achieve efficient world-wide routing of IP Packets [6]. The model of an autonomous system is depicted in Fig. 1. In this model, each AS groups a set of clients that are relatively close to each other in a network-topological sense. In this paper, we consider a more general system model, in which an intermediate server is configured either as a replica server, or a cache, or neither. Finally, we use both trace data and statistical data to simulate our methods. The experimental results show that our improved GD* algorithm can outperform the existing GD* algorithm over several performance metrics, and our group-based method for document distribution strategy selection can outperform the global strategy and improve several performance metrics compared to the document-based method, while keeping the others almost the same.

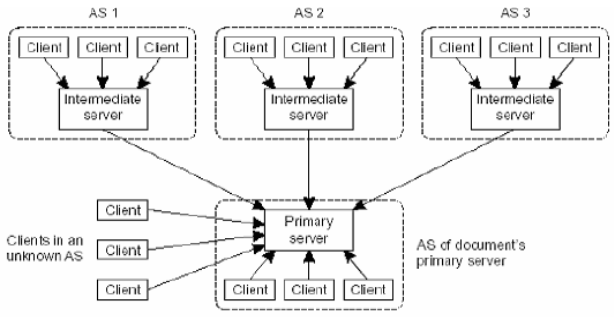

Figure 1: An Autonomous System Model

Our contributions in this paper are summarized as follows. (1) We propose an improved GD* cache replacement algorithm, which exhibits experimentally better performance than the existing GD* algorithm. (2) We present a group-based method for dynamically selecting distribution strategies for web documents according to access patterns, which can reduce network traffic and improve system performance. (3) We use trace data and statistical data to simulate our methods, and compare our results with existing methods.

The rest of the paper is organized as follows. Section 2 discusses some related work. Section 3 presents our improved GD* cache replacement algorithm. Section 4 focuses on our group-based method for dynamically selecting distribution strategies for web documents according to access patterns. The simulation experiments are described in Section 5. Finally, we conclude our paper in Section 6.

2. Related Work

A number of proposals have been made to improve the service quality of the Internet by means of caching and replication, since these are efficient ways to reduce access latency and network bandwidth. For an overview of caching and replication on the Internet, see [18,26]. Document placement and replacement are two important issues for web caching. Document placement decides where to place a new document, while document replacement concentrates on which document should be removed to make room for a new document. Removing suitable documents can improve cache hit ratio and reduce web traffic. There are many cache replacement algorithms in the literature [1, 15, 23, 24]. The demand for dynamic replication comes from the continuous increase in the large-scale web hosting market; it is evident that manual manipulation of a huge number of replicas is not feasible. There are three main challenges involved in implementing a replicated service on the Internet [18]. The first is how to assign requests to servers according to some performance criteria, in a way that is transparent to the end user. This, in turn, gives rise to two new problems: who should decide about the request redirection (location) and where the request should be directed (server selection). The second challenge is how to decide the number and placement of replica servers. The last is how to maintain content consistency. Depending upon where the redirection occurs, the various schemes for maintaining content consistency can be grouped in several main categories, such as client side redirection [4, 27], DNS redirection [7], server side direction [4], etc. Various methods for selecting among replicated servers have been extensively discussed (for an overview, see [2, 11]). Little attention has been paid to the question of where to place the replica servers in order to increase overall system performance. Research efforts in this area can be classified as follows, depending on their similarity to well known theoretical problems. (1) k-median [14, 25]: The k-median problem consists of placing k servers on the nodes in order to minimize total network cost, given that each node can hold at most one server. It has been shown that this problem is NP-hard. (2) Bin Packing [19]: Bin packing is commonly used to model load balancing problems. (3) File Allocation [3, 12]: File allocation is used to allocate the objects to sites to

optimize a performance parameter. Regarding content consistency issues, an overview of various algorithms can be found in [22].

Most researchers have concentrated on supporting a single family of strategies. For example, the TACT tool kit [28] provides support for replication based on anti-entropy schemes for a range of consistency models. The method presented in [21], which is similar to our method to some extent, allows each distributed document to have its own associated strategy. However it has the following deficiencies: (1) It applies only LRU for replacement problems. Although LRU has been widely adopted for cache replacement problems, its disadvantage is that it does not consider variable size or variable cost documents. (2) The total performance results are calculated by summing the performance results of each document. In fact there are a number of performance metrics, such as hit ratio and byte hit ratio, which cannot be managed in this way. Our method corrects these deficiencies. Furthermore, we apply our improved GD* algorithm to deal with document replacement problems.

3. An Improved GD* Cache Replacement Algorithm

In this section we introduce an improved GD* cache replacement algorithm based on the existing GD* algorithm [15]. Document placement and replacement are two important issues for web caching. Document placement decides where to place
a new document; since the cache size is limited, document replacement concentrates on which document should be removed to make room for the new document. Removing suitable documents can improve cache hit ratio and reduce web traffic. Therefore, replacement algorithms can have a great effect on system performance. There are many cache replacement algorithms such as LRU, LNC-W3, GD*, etc. As mentioned before, the GD* algorithm is one of the most efficient cache replacement algorithms. It considers both popularity and correlation. However it doesn't consider document update frequency as a factor in the utility function that is used to decide which document should be replaced. The GD* algorithm subsumes a family of algorithms, each with a different level of dependency on long-term document popularity and short-term reference correlations.

Modification to resources greatly affects the performance of web caching [17]. Knowing that certain kind of resources change more often than others can guide the caching policies of browsers and proxies. For example, resources that change less often may be given preference in caching, or revalidated with the origin server less frequently; therefore, document update frequency should be an important factor in deciding which document should be replaced. In this paper we propose an improved GD* cache replacement algorithm that considers document update frequency as a factor in its utility function, and our simulation results show that our improved algorithm exhibits an experimentally better performance than the existing GD* algorithm. Our improved algorithm is shown as follows:

$H_0 \longleftarrow 0$
for each request for each document d do
  if d is in the cache then
    $H(d) = H_0 + (\frac{f(d)\cdot c(d)}{s(d)\cdot u(d)})^{1/\beta}$
  else
    fetch d
    while there is not enough free cache for d do
      $H_0 = min_p\{H(p) | p \text{ is in the case}\}$
        evict the minimum p
    $H(d) = H_0 + (\frac{f(d)\cdot c(d)}{s(d)\cdot u(d)})^{1/\beta}$

where $f(d)$ refers to the request frequency of document d, $c(d)$ is the cost of fetching document d from the server, $s(d)$ is the size of document d, and $u(d)$ is the update frequency of document d.
In our algorithm, the utility value $H(d)$ for document d reflects the normalized expected cost saving if that document d stays in the cache. Obviously $H(d)$ is proportional to $1/u(d)$. We use an inflation value H_0 to age the documents in the cache. When the value H_0 catches up with $H(d)$, document d will be the candidate for eviction. On each hit or when fetching from the server, our algorithm resets the value $H(d)$ for document d. Our algorithm captures both popularity and temporal correlation. $[f(d)\cdot c(d)]/[s(d)\cdot u(d)]$ captures long-term popularity, while β control the rate of aging.

4. Document Distribution Strategy Selection

In this section, we first briefly outline the distributionstrategies used in this paper, and then we present a group-based method for dynamically selecting distribution strategies for web documents according to access patterns.

4.1 Distribution Strategies

We considered the following document distribution strategies.
1. No Replication (NoRepl): This is a basic strategy that does not use any replication at all. All clients connect to the primary server directly.
2. Verification (CV): When a cache hit occurs, the cache systematically checks the copy's consistency by sending an If-Modified-Since request to the primary server before sending the document to the client. After the primary server revalidates the request, the intermediate decides how to get the document for the client.
3. Limited verification (CLV): When a copy is created, it is given a time-to-live (TTL) that is proportional to the time elapsed since its last modification. Before the expiration of the TTL, the cache manages requests without any consistency checks and sends the copies directly to the client. After the TTL expires, the copies are removed from the cache.

In our experiments, we used the following formula to decide the TTL. $\alpha = 0.2$ is the default in the Squid cache [4].

$$T_r = (1+\alpha)T_c - T_l$$

where T_r is the expiration time, T_c is the cached time, and T_l is the last modified time. α is a parameter which can be selected by the user.

4. Delayed verification (CDV): This policy is almost identical to the CLV strategy. When a copy is created, it is also given a TTL. When t the TTL expires, the copies are not removed from the cache immediately; the cache sends an If-Modified-Since request to the primary server before sending the copies to the client. After the primary server revalidates the request, the intermediate decides how to fetch the document for the client.

From an ideal point of view, we put as many replica servers as ASes, so every client can fetch the document he needs from the replica server, which, in turn, leads to good results on some performance metrics such as hit ratio and byte hit ratio. But on the other hand, it also will make some performance metrics, such as consumed bandwidth and server load, worse.

5. SU50 (Server Update): The primary server maintains the copies at the 50 most relevant intermediate servers.

6. SU50+CLV: The primary server maintains the copies at the 50 most relevant intermediate servers; the other intermediate servers follow the CLV strategy.

Ideally, we would have as many replica servers as ASs, so every client could fetch the needed document from the replica server; this, in turn, would produce good results on some performance metrics such as hit ratio and byte hit ratio. However, on the other hand, it also would make other performance metrics, such as consumed bandwidth and server load, worse.
5. SU50 (Server Update): The primary server maintains copies at the 50 most relevant intermediate servers.
6. SU50+CLV: The primary server maintains copies at the 50 most relevant intermediate servers; the other intermediate servers follow the CLV strategy.

4.2 A Group-Based Method for Selection of Document Distribution Strategies

First we introduce a method to group the documents into P groups according to their access patterns. The main factors that influence the access pattern are web resource and user behavior. According to [17], we group the documents according to the value of v_d, which is defined as $v_d = (c_d + f_d / u_d)/s_d$, where c_d denotes the cost of fetching document d from the primary server, f_d denotes the access frequency of document d, u_d denotes the update frequency of document d, and s_d denotes the size of document d. We can see that when P is equal to the number of the documents, i.e., there is only one document in each group, our method is the same as the method in [21]. Therefore, the method proposed in [21] can be viewed as a special case of our method. For the case of $P = 1$, our method can be viewed as a global strategy method since all the documents are assigned to the same strategy.
Now we present our group-based model considering the total effect of the performance metrics from a general point of view, e.g. we define the total function for each performance metric according to its characteristics. The existing method in [21] defines the total function for each performance metric by summing the performance metrics of each document. We argue that this method does not always work well for some performance metrics such as total hit ratio.

Let $S = \{s_j, j = 1,2,\cdots,|S|\}$ be the set of distribution strategies, $G = \{G_j, j = 1,2,\cdots,|G|\}$ be the set of the groups, $M = \{m_j, j = 1,2,\cdots,|M|\}$ be the set of performance metrics such as total turnaround time, hit ratio, total consumed bandwidth, etc. A pair of arrangement (*strategy, group*) means that a strategy is assigned to the documents in a group. We denote the set of all the possible arrangements as A. We can define a function f_k for each metric m_k on a pair $a \in A$ by $R_{ka} = \sum_{j=1}^{|G|} r_{kaj} = \sum_{j=1}^{|G|} f_k(a, G_j)$, where R_{ka} is the

aggregated performance result on metric m_k and r_{kaj} is the performance result on metric m_k for G_j.

Let $\omega = (\omega_1, \omega_2, \cdots, \omega_{|M|})$ be the weight vector which satisfies $\sum_{k-1}^{|M|} \omega_k = 1, \omega_k \geq 0, k = 1,2,\cdots,|M|$. We can get the following general model defined as $R_a^* = \min_k \sum_{k=1}^{|M|} \omega_k R_{ka}$, where R_a^* is the total cost function for different weight vector ω for an arrangement a.

Since there are total of $|S|^{|G|}$ different arrangements, therefore it is not computationally feasible to achieve the optimal arrangements by the brute-force assignment method. The following result shows that it requires at most $|G| \cdot |S|$ computations to obtain an optimal strategy arrangement for the documents in each group.

$$
\begin{aligned}
R^* &= \min_{a \in A} \sum_{k=1}^{|M|} \omega_k R_{ka} \\
&= \min_{a \in A} \sum_{k=1}^{|M|} \omega_k \left(\sum_{j=1}^{|G|} r_{kaj} \right) \\
&= \min_{a \in A} \sum_{k=1}^{|M|} \sum_{j=1}^{|G|} \omega_k r_{kaj} \\
&= \min_{a \in A} \sum_{j=1}^{|G|} \sum_{k=1}^{|M|} \omega_k r_{kaj} \\
&\geq \min_{a \in A} \sum_{j=1}^{|G|} \left(\min_j \sum_{k=1}^{|M|} \omega_k R_{ka} \right)
\end{aligned}
$$

From the above reasoning, we can obtain the total optimal arrangement by computing the optimal arrangement for each group. Therefore, the computation is the sum of that for obtaining the optimal arrangement for the documents in each group, whereas the computation workload for the method in [21] is about $|D| \cdot |S|$. Therefore, our method requires less computation than the method in [21] by $(|D| - |G|) \cdot |S|$. Suppose that there are 100 documents and we divide the documents into 10 groups, we can see that the computation can be reduced by 90%.

In the experiments we mainly considered the following performance metrics: (1) Average Response Time per request (ART): the average time for satisfying a request. (2) Total Network Bandwidth (TNB): the total additional time it takes to transfer actual content, expressed in bytes per milli-second. (3) Hit Ratio (HR): the ratio of the requests satisfied from the caches over the total requests. (4) Byte Hit Ratio (BTR): the ratio of the number of bytes satisfied from the caches over the total number of bytes.

For the case of $k = 1,2$, suppose $\alpha = \max_j R_{kj}$ and $\beta = \min_j R_{kj}$ before defining the function for calculating the total performance result for these two cases, we should apply a transformation $f(R_{kj}) = (R_{kj} - \beta)/(\alpha - \beta)$ such that $f(R_{kj}) \in [0,1]$. Therefore all the performance results are in the interval of $[0,1]$. Otherwise it is not feasible to the weights for the performance metrics. For example, in the case of ART=150, TNB=200, HR=0.9, and NHR=0.9, and $\omega = (0.45,0.45,0.05,0.05)$, we can see that HR and THR play little role on the total cost although their weights are very large. For the case of $k = 3,4$, we define $R_k = \sum_{j=1}^{|D|} f(R_{kj})/|D|$.

5. Simulation

In this section we use trace data and statistical data to simulate the methods proposed in previous sections. In the simulation model, we assume that the primary server has the privilege of updating the documents whose copies are distributed or stored in the replica servers and the caches. A replica server always holds the document; a cache may or may not hold it. In the following figures, “I-GD*” and “GD*” represent the performance results of our improved GD* algorithm and the existing GD* algorithm, respectively. “Per-Group” and “Per-Document” represent the performance results of our group-based method and the existing document-based method.

5.1 Simulation with Trace Data

In this section we apply trace data to simulate our results. The trace-based simulation method is similar to that introduced in [20]. In our experiments, we collected traces from two web servers created by the Vrije Universiteit Amsterdam in the Netherlands (VUA) and the National Laboratory for Applied Network Research (NLANR). Table ¥ref{statistics} shows the general statistical data for the traces.

Table 1: Statistics of Trace Data

Issue	VU Amstetdam	VU Amstetdam
Start Date	Sept 19, 1999	March 27, 2001

End date	Dec 24, 1999	April 11, 2001
Durations (days)	96	16
# of Documents	26556	187,356
# of Requests	1,484,356	3,037,625
# of Creates	26556	187,356
# of Updates	85,327	703,845
# of Ases	7563	90

5.1.1 Experiment for Improved Replacement Algorithm

In this section, we simulate our improved GD* cache replacement algorithm across a wide range of cache sizes, from 0.04 percent to 12.0 percent. Here, the cache size is the percent of the total unique file size. In our experiments, we considered the following performance metrics: average response time, request response ratio, which is defined as the ratio of the access latency of the target object to its size; hit ratio and byte hit ratio. From figs 2-5, we can easily see that our algorithm achieved improved results for all the performance metrics considered. Specifically, the mean improvements of the average response time over GD* for VAU and NLANR are 8.1 percent and 6.2 percent, respectively.

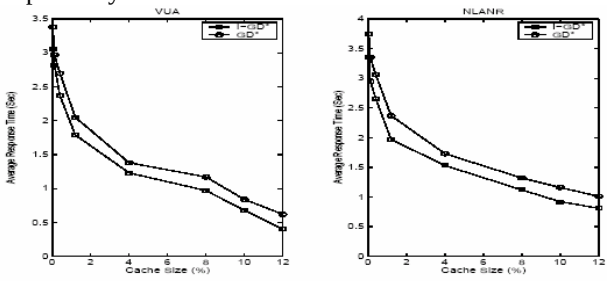

Figure 2: Average Response Time

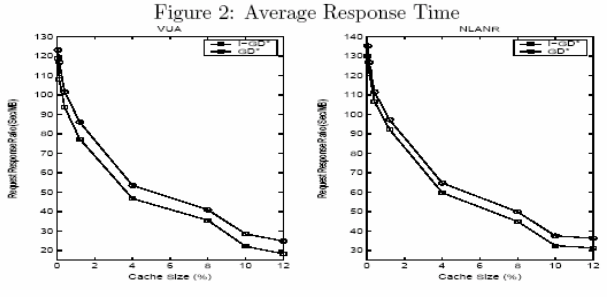

Figure 3: Request Response Ratio

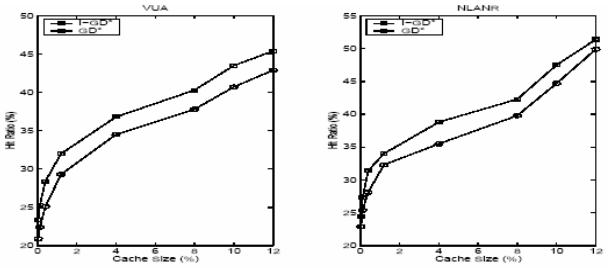

Figure 4: Hit Ratio

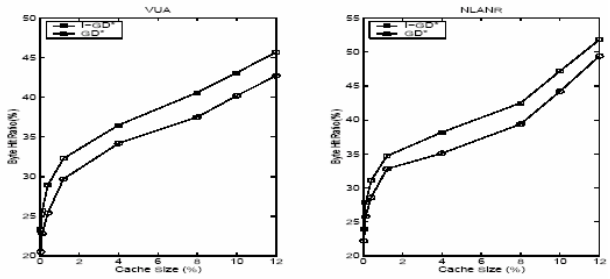

Figure 5: Byte Hit Ratio

5.1.2 Experiment for A Global Strategy

In this experiment we assigned the same distribution strategy to each document. The results of this experiment are shown in Table 2. Note that when NoRepl is used, there is a higher average response time and a higher consumption of total network bandwidth, since the strategy does not consider caching and replication, but both byte hit ratio and hit ratio are 100% since it accesses the primary server directly. Distribution strategies for the replica servers can bring down the average response time, but they also lead to an increased consumption of total network bandwidth. It is difficult to develop a distribution strategy that can optimize all the performance metrics. Generally, you can make one or more of the performance metrics better, but at the cost of making some others worse (see Table 2). Most strategies are relatively good with respect to one or more metrics, but no strategy is optimal for all.

Table 2: Performance Results for Global Strategies

Distribution Strategy	VUA				NLANR			
	TNB(GB)	ART(Sec)	BHR(%)	HR(%)	TNB(GB)	ART(Sec)	BHR(%)	HR(%)
NoRepl	176.9	11.8	100	100	308.5	9.55	100	100
CV	168.6	12.3	99.4	99.2	292.1	9.80	99.7	99.4
CLV	161.3	9.01	88.3	88.5	274.6.3	7.99	87.4	87.6
CDV	164.5	8.75	89.9	89.8	283.4	7.89	88.3	88.5
SU50	165.7	6.96	92.5	92.5	287.7	6.86	90.8	91.1
SU50+CDV	160.9	5.92	95.7	95.4	298.6	6.32	95.0	94.8

5.1.3 Experiment for Per-Group Strategy

In this section we describe our experiment for assigning the same distribution strategy to the documents in each group. The simulation results shown in Table 3 were obtained when the number of groups was 100 and 200 for VUA and NLANR, respectively. We simulated a

case in which there are two performance metrics, ART and TNB.

Table 3: Performance Results for Per-Group Strategy

$w = (w_1, w_2)$	VUA		NLANR	
	TNB(GB)	ART(Sec)	NB(GB)	TT(hours)
(0.9,0.1)	95.3	8.82	162.2	5.37
(0.8,0.2)	110.2	6.95	175.7	5.68
(0.7,0.3)	126.5	6.24	196.7	5.83
(0.6,0.4)	136.5	5.86	212.5	6.04
(0.5,0.5)	150.7	5.57	256.5	6.27
(0.4,0.6)	167.4	5.33	283.5	6.62
(0.3,0.7)	178.2	5.20	314.5	6.89
(0.2,0.8)	191.7	5.11	346.8	7.05
(0.1,0.9)	205.6	5.05.1	379.4	7.24

From Figure 6 we can see that the results of our method approximate those of the existing method when we group the documents into 117 and 211 groups for VUA and NLANR, respectively. From our experiments, we conclude that there is almost no improvement in result as the number of groups increases. However, our method can significantly improve both the procedure execution time and the memory management cost, as can be seen in figs 7 and 8.

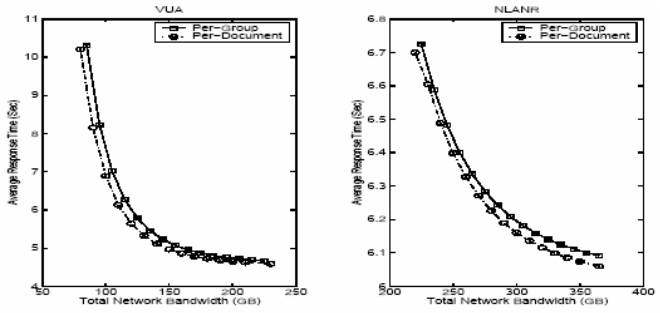

Figure 6: Different Arrangements

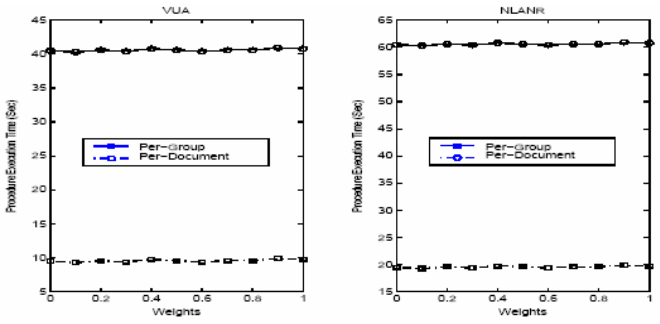

Figure 7: Procedure Execution Time

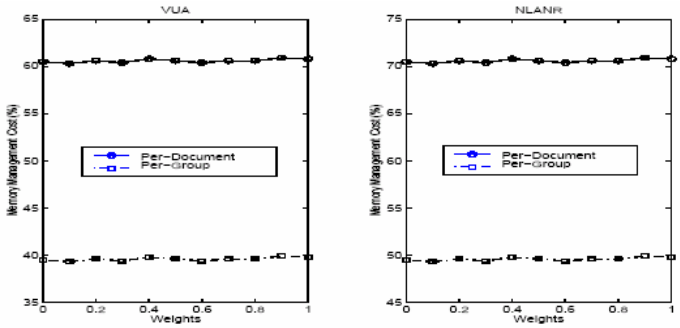

Figure 8: Memory Management Cost

5.2 Simulation with Statistical Data

In this section we use statistical data to simulate our methods. The parameters shown in Table ¥ref{parameter} are chosen from the open literature and are considered to be reasonable [1,5, 9, 10]. We have conducted experiments for many topologies with different parameters and the performance of our results was found to be insensitive to topology changes. Here, we list only the experimental results for one topology, due to space limitations.

Table 4: Parameters Used in Simulation

Parameter	Value
Number of Nodes	200
Number of Web Objects	5000
Number of Requests	500000
Number of Updates	10000
Web Object Size Distribution	Pareto Distribution $p(x) = \frac{ab^a}{a-1}$ $(a = 1.1, b = 8596)$
Web Object Access Frequency	Zipf-Like Distribution $\frac{1}{i^\alpha}$ $(i = 0.7)$
Delay of Links	Exponential Distribution $p(x) = \theta^{-1}e^{-x/\theta}$ $(\theta = 0.06\ Sec)$
Average Request Rate Per Node	$U(1, 9)$ requests per second

5.2.1 Improved Replacement Algorithm Performance Results

In Fig. 9, we can easily see that our improved GD* algorithm outperform the existing GD* algorithm over all the performance metrics considered.

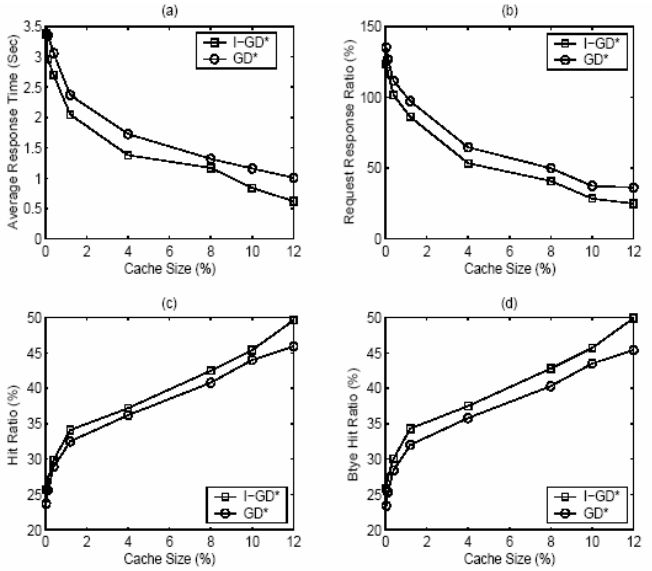

Figure 9: Performance Results for Improved Replacement Algorithm

5.2.2 Per-Group Strategy

From Fig. 10 we can see that the results of our method approximate those of the existing method when we group the documents into 89 groups. However, our method can improve both the procedure execution time and the memory management cost.

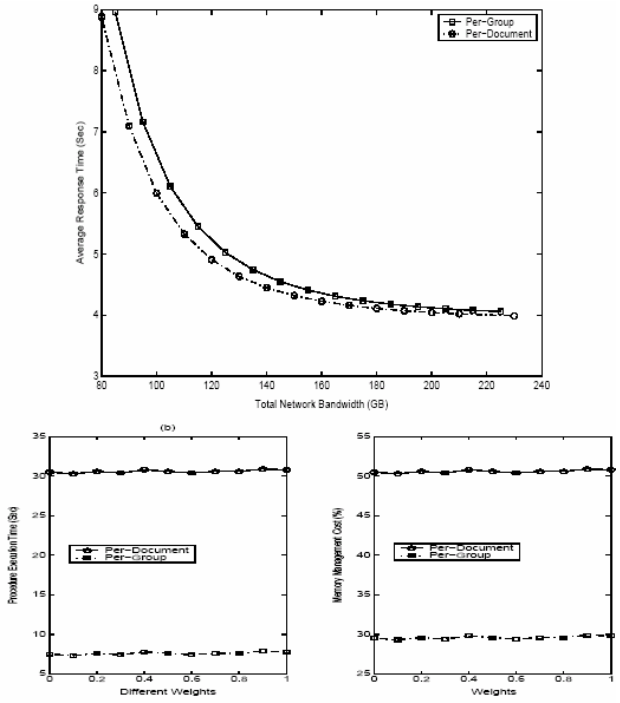

Figure 10: Performance Results for Per-Group Strategy

6. Concluding Remarks

Since web caching and replication are efficient ways to reduce web traffic and latency for users, more and more researchers have been paying a lot of attention to this topic. In this paper, we proposed an improved GD* cache replacement algorithm and resented a method for dynamically selecting web replication strategies according to the access patterns. We also used both web race and statistical data to simulate our method and our algorithm. However, we can see that there will be performance problems when more strategies are considered. In the future, this work should be extended to the replication of other types of objects, since we considered only static objects in this paper. The application of our method to dynamical web documents also should be studied. Such studies should lead to a more general solution to web caching and replication problems.

References

1. Y. Amir, A. Peterson and D. Shaw. Seamlessly selecting the best copy from Internet-wide replicated web servers. Proc. Of the 12th Int. Symposium on Distributed Computing, 1998.

2. B. Awerbuch, Y. Bartal and A. Fiat. Optimally-competitive distributed file allocation. Proc. Of the 25th Annual ACM STOC, pp. 164-173, 1993.

3. M. Baentsch, L. Baum, G. Molter, S. Rothkugel and P. Sturm. Enhancing the web infrastructure-from caching to replication. IEEE Internet Computing, pp. 18-27, 1997.

4. T. Bates, E. Gerich, L. Joncheray, J. M. Jouanigot, D. Karrenberg, M. Terpstra and J. Yu. Representation of IP routing policies in a routing registry. RFC 1786, 1995.

5. M. Beck and T. Moore. The Internet-2 distributed storage infrastructure project: an architecture for Internet content channels. Proc. Of the 3rd Int. WWW caching Workshop, June, 1998.

6. A. Bestavros. WWW traffic reduction and load balancing through server-based caching. IEEE Concurrency: Special Issue on Parallel and Distributed Technology. Vol. 15, pp. 56-67, 1997.

7. A. Bestavros, M. Crovilla, J. Liu and D. Martin. Distributed packet rewriting and its application to scalable server architectures. Proc. Of the 6th Int. Conference on Network Protocols (ICNP'98), 1998.

8. M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham and P. Shenoy. Adaptive push-pull: disseminating dynamic web data. IEEE Transactions on Computers, pp. 652-667, Vol 51, No. 6, June 2002.

9. V. Cardellini, M.Colajanni and P. Yu. High performance web-server systems. Proc. Of the 13th Int. Symposium on Computer and Information Sciences (ISCI'1998), pp. 288-293, 1998.

10. A. Chankhunthod, P. B. Danzig, C. Neerdaels, M. F.

Schwartz and K. J. Worrell. A hierarchical Internet object cache. Proceedings of the 1996 Usenix Technical Conference (San Diego, CA), pp.153-163, 1996.
11. P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham and P. Shenoy. "Adaptive push-pull: disseminating dynamic web data". Proc. 10th Int'l WWW Conf., pp 265-274, May 2001.
12. L. W. Dowdy and D. v. Foster. Comparative models of the file assignment problem. ACM Computing Surveys, Vol. 14 (2), 1982.
13. S. Irani. A competitive analysis of paging. Available at: http://www.ics.uci.edu/~irani/.
14. S. Jamin, C. Jin, T. Kurc, D. Raz and Y. Shavitt. Constrained mirror placement on the Internet. Proc. Of the IEEE INFOCOM 2001 Conference, 2001.
15. S. Jin and A. Bestavros. Greeddual* web caching algorithm exploiting the two sources of temporal locality in web request streams. Computer Comm. Vol. 4, No. 2, pp. 174-183, 2001.
16. S. Jin and A. Bestavros. Sources and characteristics of web temporal locality. Proc. of IEEE/ACM Symp. Modeling, Analysis and Simulation of Computer and Telecommunication Systems, San Francisco, CA, 29 August - 1 September.17.
17. B. Krishnamurthy and J. Rexford. Web Protocols And Practice. 2001.
18. T. Loukopoulos, I. Ahmad and D. Papadias. An overview of data replication on the Internet. Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks. pp, 31-37, 2002.
19. B. Narengran, S. Rangarajan and S. Yajnik. Data distribution algorithms for load balanced fault-tolerant web access. Proc. Of the 16th Symposium on Reliable Distributed Systems (SRDS'97), pp. 22-24, 1997.
20. G. Pierre and M. Makpangou. Saperlipopette!: a distributed web caching systems evaluation tool. Proc. 1998 Middleware Conf., pp. 389-405, Setp. 1998.
21. G. Pierre and M. Steen. Dynamically selecting optimal distribution strategies for web documents. IEEE Transactions on Computers, pp. 637-651, Vol. 51, No. 6, June 2002.
22. Y. Saito. Optimistic replication algorithms. Technical
23. P. Scheuermann, J. Shim, R. and Vingralek. A case for delay-conscious caching of web documents. Computer Network and ISDN Systems, Vol. 29, pp.997-1005, 1997.
24. J. Shim, P. Scheuermann, and R. Vingralek. Proxy cache algorithms: design, implementation, and performance. IEEE Transaction on Knowledge and Data Engineering, Vol. 11, pp. 549-562, 1999.
25. A. Vigneron, L.Gao, M.Golin, G. Italiano and B. Li. An algorithm for finding a k-median in a directed tree. Information processing letters, pp. 81-88, 2000.
27. J. Wang. A survey of web caching schemes for the Internet. ACM SIGCOMM Computer Comm. Rev., Vol. 29, pp. 36-46, 2000.
27. C. Yoshikawa, B.Chun, P. Eastham, A. Vahdat, T. Anderson and D.Culler. Using smart clients to build scalable services. Proc. Of the 1997 USENIX Annual Techinical Conference, pp. 6-10, 1997.
28. H. Yu and A. Vahdat. Design and evaluation of a Continuous consistency model for replicated services. Proc. Fourth Symposium Operating System Design and Implementation, 2000.

Dr. Wenyu Qu received his Bachelor degree and Master degree in the Department of Applied Mathematics and the Department of , Dalian University of Technology, Dalian, China in 1994 and 1997 respectively, and obtained the doctor degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Japan. She is currently a Postdoctoral Researcher in the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Japan. His research interests include mobile agent-based technology, network routing and resource management, and fault-tolerant computing.

Dr. Keqiu Li received his Bachelor degree and Master degree in the Department of Applied Mathematics, Dalian University of Technology, Dalian, China in 1994 and 1997 respectively. He obtained the doctor degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, Japan. He is currently a professor in the College of Computer Science and Technology, Dalian Maritime University, China. His research interests include Internet technology, distributed computing, multimedia application, and network security.

Dr. Bo Jiang is currently a professor in the College of Computer Science and Technology, Dalian maritime University, China. His research interests include software engineering and networking.

Dr. Hong Shen is currently a full Professor in the Graduate
School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa. He has published over 140 technical papers on algorithms, parallel and distributed computing, interconnection networks, parallel databases and data mining, multimedia systems, and networking.

Dr. Di Wu is currently a associate professor in the Department of Computer Science and engineering, Dalian University of Technology, China. His research interests include software engineering and networking.