Dynamically Selecting Distribution Strategies for Web Documents According to Access Pattern[⋆]

Wenyu Qu [1], Di Wu [2], Keqiu Li [3], and Hong Shen [1]

[1] Graduate School of Information Science
Japan Advanced Institute of Science and Technology
1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan

[2] Department of Computer Science and Engineering
Dalian University of Technology
No 2, Linggong Road, Ganjingzi District, Dalian, 116024, China

[3] College of Computer Science and Technology
Dalian Maritime University
No 1, Linghai Road, Dalian, 116026, China

Abstract. Web caching and replication are efficient techniques for reducing web traffic, user access latency, and server load. In this paper 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. We use both trace data and statistical data to simulate our methods. The experimental results show that 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.

[⋆] The preliminary version of this paper appeared in [8]. Corresponding author: K. Li (keqiu_01@163.com).

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 [4] 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 [9]. 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. In this paper 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 [8] 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 [3]. 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 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.

The rest of the paper is organized as follows. Section 2 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 3. Finally, we conclude our paper in Section 4.

2 Selection of Document Distribution Strategy

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

2.1 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 determine the TTL. $\alpha = 0.2$ is the default in the Squid cache [3].

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

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, in that a copy is created, it is also given a TTL. However, when 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.

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.

2.2 A Group-Based Method for Document Distribution Selection

First we introduce a method to group the documents into P groups according to their access patterns. The main factors that influence the access patterns are web resource and user behavior. According to [7], we group the documents according to the value of v_d, which is defined as follows:

$$v_d = (c_d + f_d/u_d)s_d \tag{2}$$

where c_d denotes the cost of fetching document d from the 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., when there is only one document in each group, then our method is the same as the document-based method in [11]. Therefore, from this point of view the method proposed in [11] can be viewed as a special case of our method. For the case of $P = 1$, our method can be considered 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 [11] 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 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 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 in metric m_k and r_{kaj} is the performance result in metric m_k for G_j .

Let $w = \{w_1, w_2, \cdots, w_{|M|}\}$ be the weight vector which satisfies:

$$\sum_{k=1}^{M} w_k = 1, w_k \geq 0, k = 1, 2, \cdots, |M| \tag{3}$$

We can get the following general model $R_a^* = \min_k w_k R_{ka}$. We refer to R_a^* as the total cost function for different weight vector w for an arrangement a .

Since there are a total of $|S|^{|G|}$ different arrangements, 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$|G||S|$ computations to obtain an optimal strategy arrangement for the documents in each group.

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

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 [11] is about $|D||S|$, where $|D|$ is the total number of documents. Thus, our method requires less computation than the method in [11] by $(|D| - |G|)|S|$. If we 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 our 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 $max = \max_j R_{kj}$, $min = \min_j R_{kj}$. Before defining the total performance metric result function for the case of $k = 1, 2$, we should apply a transformation $f(R_{kj}) = (R_{kj} - min)/(max - min)$ on R_{kj} so that $f(R_{kj}) \in [0, 1]$. Therefore all the performance metric results are in the interval $[0, 1]$. Otherwise it is not feasible to decide the weights for the performance metrics. For example, in the case of $ART = 150$, $TNB = 200$, $HR = 0.9$, and $BHR = 0.9$, let $w = (0.45, 0.45, 0.05, 0.05)$. In this case, we can see that HR and BHR play little role in the total cost, although the weights of them are very large. For ART and TNB, we define $R_k = \sum_{j=1}^{|D|} f(R_{kj})/|D|, k = 3, 4$.

For $HR(k = 3)$, let $R_{1j} = f_1(s_i, G_j)$ be the number of requests that hit in the replica servers and the caches for the pair (s_i, G_j). We define $R_1 = \sum_{j=1}^{|D|} R_{1j}/NR$, where NR is the total number of requests.

For $BHR(k = 4)$, let $R_{2j} = f_2(s_i, G_j)$ be the number of bytes that hit in the replica servers and the caches for the pair (s_i, G_j). We define $R_2 = \sum_{j=1}^{|D|} R_{2j}/NBR$, where NBR is the total number of requests bytes.

3 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, "Per-Group" and "Per-Document" represent the performance results of our group-based method and the existing document-based method.

3.1 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 [10]. 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 1 shows the general statistical data for the traces.

Table 1. Statistics of Trace Data

Issue	VUA	NLANR
Start Date	September 19, 1999	March 27, 2001
End Date	December 24, 1999	April 11, 2001
Duration (days)	96	16
Number of Documents	26,556	187,356
Number of Requests	1,484,356	3,037,625
Number of Creates	26,556	187,356
Number of Updates	85,327	703,945
Number of ASs	7,563	90

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 2 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.

From Figure 1 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 Figures 2 and 3.

Table 2. 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

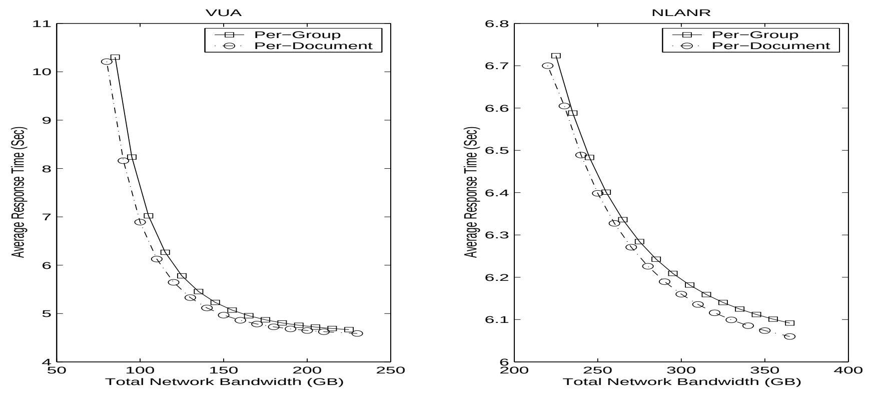

Fig. 1. Different Arrangements

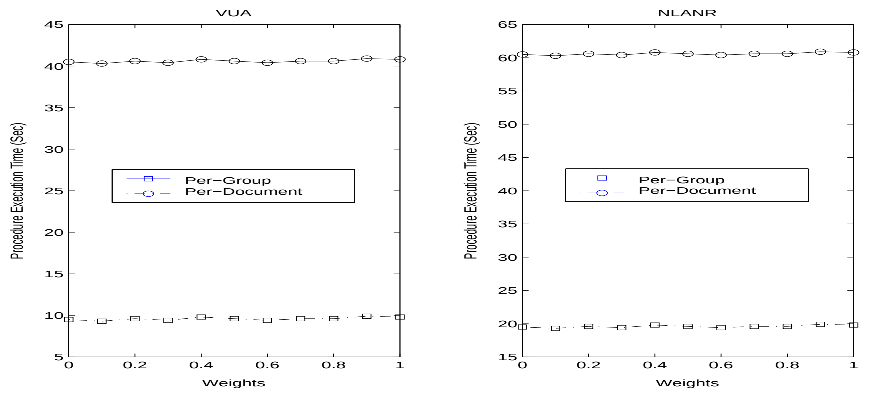

Fig. 2. Procedure Execution Time

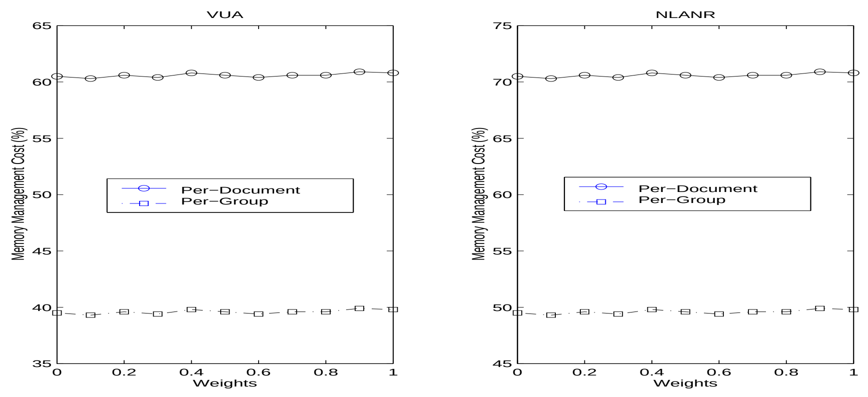

Fig. 3. Memory Management Cost

3.2 Simulation with Statistical Data

In this section we use statistical data to simulate our methods. The parameters shown in Table 3 are chosen from the open literature and are considered to be reasonable [1, 2, 5, 6]. 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 3. 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

From Figure 4 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.

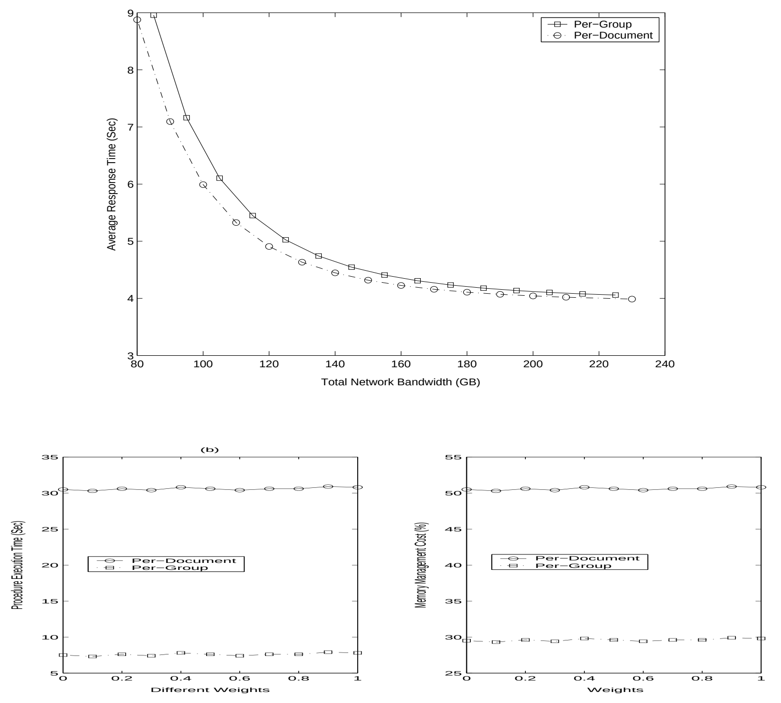

Fig. 4. Performance Results for Per-Group Strategy

4 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 presented a method for dynamically selecting web replication strategies according to the access patterns. We also used both web trace and statistical data to simulate our method. 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. Aggarwal, C., Wolf, J. L. and Yu, P. S. (1999) *Caching on the World Wide Web.* IEEE Transaction on Knowledge and Data Engineering, 35, 94-107.
2. Barford, P. and Crovella, M. (1998) *Generating representative web workloads for network and server performance evaluation.* Proc. of ACM SIGMETRICS'98, Madison, WI, June, pp. 151-160.
3. Bates, T., Gerich, E., Joncheray, L., Jouanigot, J. M., Karrenberg, D., Terpstra, M. and Yu, J. (1995) *Representation of IP routing policies in a routing registry.* Technical Report, Zvon-RFC 1786, May.
4. Bestavros, A. (1997) *WWW traffic reduction and load balancing through server-based caching.* IEEE Concurrency: Special Issue on Parallel and Distributed Technology, 15, 56-67.
5. Breslau, L., Cao, P., Fan, L., Phillips, G. and Shenker, S. (1999) *Web caching and zipf-like distributions: evidence and implications.* Proc. of IEEE INFOCOM'99, March, pp. 126-134.
6. Calvert, K. L., Doar, M. B. and Zegura, E. W. (1997) *Modelling internet topology.* IEEE Comm. Magazine, 35, 160-163.
7. Krishnamurthy, B. and Rexford, J. (2001) *Web Protocols And Practice.* Addison-Wesley, Boston.
8. Li, K. and Shen, H. (2004) *Dynamically Selecting Distribution Strategies for Web Documents According to Access Pattern,* Proc. of the Fifth International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 04), pp. 554-557.
9. Loukopoulos, T., Ahmad, I. and Papadias, D. (2002) *An overview of data replication on the Internet.* Proc. of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN'02), Makati City, Metro Manila, Philippines, 22-24 May, pp, 31-37.
10. Pierre, G. and Makpangou, M. (1998) *CSaperlipopette!: a distributed web caching systems evaluation tool.* Proc. of 1998 Middleware Conference, The Lake District, England, 15-18 September, pp. 389-405.
11. Pierre, G. and Steen, M. (2002) *Dynamically selecting optimal distribution strategies for web documents.* IEEE Transactions on Computers, 51, 637-651.