A Case for Dynamic Selection of Replication and Caching Strategies

Sivasubramanian, S.; Pierre, G.E.O.; van Steen, M.R.

published in
8th Web Caching Workshop
2003

document version
Publisher's PDF, also known as Version of record

A Case for Dynamic Selection of Replication and Caching Strategies

Swaminathan Sivasubramanian Guillaume Pierre
Maarten van Steen
Dept. of Mathematics and Computer Science
Vrije Universiteit, Amsterdam, The Netherlands
{*swami,gpierre,steen*}*@cs.vu.nl*

Abstract

Replication and caching strategies are being used to reduce user perceived delay and wide area network traffic. Numerous such strategies have been proposed to manage replication while maintaining consistency among the replicas. In earlier research, we demonstrated that no single strategy can perform optimal for all documents, and proposed a system where strategies are selected on a per-document basis using trace-driven simulation techniques. In this paper, we demonstrate the need for continuous dynamic adaptation of strategies using experiments conducted on our department Web traces. We also propose two heuristics, *Simple* and *Transition*, to perform this dynamic adaptation with reduced simulation cost. In our experiments, we find that *Transition* heuristic reduces simulation cost by an order of magnitude while maintaining high accuracy in optimal strategy selection.

1 Introduction

Web users often experience slow document transfers for Web documents. To reduce access time, many systems replicate or cache documents at servers close to the clients. However, if a document is updated, replicas must be updated to prevent clients accessing a stale copy.

Many strategies have been proposed to achieve replication while maintaining consistency among replicas. A replication strategy dictates the number and location of replicas and the choice of a protocol governing the creation of replicas and consistency enforcement. Different strategies may offer various levels of performance and consistency, so a system designer should be careful when selecting a replication strategy.

We showed in earlier research that no single strategy can universally perform optimal for all documents [8]. An important gain in performance can be obtained by associating each document with the strategy that suits it best.

In this paper, we make a case for re-evaluating this document-to-strategy association from time-to-time, as changes in documents' access and update patterns are likely to affect the system performance. We do not deal with adaptation of strategies during emergency situations, such as flash crowds, rather focus on adaptation and selection of strategies for relatively stable access patterns.

We employ an approach where the best strategy for each document is periodically selected among a set of candidate strategies. The choice of "best" strategy is made by simulating the performance that each strategy would have provided in the recent past. If necessary, the strategy for the concerned document is switched dynamically.

Each server adaptation requires $d \cdot s$ simulations, where d is the number of hosted docu-

ments and s is the number of candidate strategies. In our current system, the simulation of a single strategy takes several tens of milliseconds on a 1-GHz PIII machine. Considering that we apply traditional and well-known trace-driven simulation techniques, we expect comparable performance for systems similar to ours. Each adaptation may thus lead to significant computational load if the number of objects or the number of candidate strategies is high. In particular, the latter can happen if we want to integrate parameterized strategies. Such strategies have a tunable parameter affecting their behavior, such as a "time-to-live" value or a "number of replicas."

The contributions of this paper are as follows: (i) we demonstrate the need for continuous dynamic adaptation of replication strategies for Web documents with experiments performed on our department server's Web traces; and (ii) we present techniques for the selection of an optimal replication strategy from a number of candidate strategies, while keeping the selection costs low. None of these contributions are reported in our earlier works [7, 8].

The rest of the paper is organized as follows: Section 2 describes our evaluation methodology. Section 3 demonstrates the need for continuous dynamic adaptation of replication strategies. Section 4 discusses our strategy selection heuristics and present their performance evaluation. Section 5 discusses the related work and Section 6 concludes the paper.

2 Evaluation methodology

We set up an experiment that simulates a system that is capable of switching its strategies dynamically for changes in the access and update patterns of its documents. Our experiment consists of collecting access and update traces from our department Web server, simulating different strategies and selecting the best one for a given period. With this setup, we observe the changes in the strategy adopted by the documents over the entire length of the traces. In this section, we present our simulation model in detail.

2.1 Simulation Model

We assume that documents have a single source of update called the *primary server*, which is responsible for sending updates to replicas located at *intermediate servers*. For the sake of simplicity, we consider only static documents in our evaluations, that is documents that change only due to updates by the primary server.

In our experiments, we use a list of 30 strategies to choose from: (i) **NR**: No replication, (ii) **CLV[p]** (Cache with Limited Validation): Intermediate servers cache documents for a given time (TTL) after which it is removed from the cache. Different strategies are derived with TTL value fixed at $p = 5\%$, 10%, 15%, and 20% of the age of document. (iii) **SI** (Server Invalidation): Intermediate servers cache the document for an unlimited time. The primary server invalidate the copies when a document is updated. (iv) **SU[x]** (Server Updates): The primary server for a document maintains copies at the x most popular intermediate servers (the top x servers sorted based on the total number of clients handled by them) for the document. When the document is updated, the primary server pushes update to the intermediate servers. Different strategies are derived for $x = 5, 10, 25, ..., 50$. (v) **Hybrid[x]** (SU[x] + CLV[10]): The primary server maintains copies at the x most popular servers, where $x = 10, 15, 20, 25, 30, 40, 50$ and the other intermediate servers follow CLV strategy, with TTL fixed at $p = 10\%$ of the age of the document.

In our simulations, we group clients based on the Autonomous Systems (AS) to which they belong. We measure the available network bandwidth between the AS belonging to the primary server and other ASes as follows: We record the time t taken by our server to serve a document of b bytes to a client from an AS. We approximate the bandwidth between primary server and the AS to which the client belongs to as b/t.

We redirect the requests of a client to the replica located in the client's AS. If there is no such replica available, then the requests are redirected to the primary server. We did not implement more sophisticated redirection policies due to the lack

of inter-AS network measurements.

2.2 Adaptation Mechanisms

As shown in [8], one can optimally assign a strategy to each object using a cost function. This function is designed to capture the inherent tradeoff between the performance gain by replication to performance loss due to consistency enforcement. In our experiments, we use a cost function that takes three parameters: (i) access latency, l; (ii) number of stale documents returned, c; and (iii) network overhead, b, that is the bandwidth used by the primary server for maintaining consistency and serving clients from ASes without replicas. The cost function for a strategy s during a given period of time t is: $cost(t,s) = w_1 * l + w_2 * c + w_3 * b$, where w_1, w_2 and w_3 are constants determining the relative weight of each metric.

The performance of a strategy s, during a given period of time t, is represented by the value of the cost function $cost(t,s)$. This value is obtained by simulation of strategy s with past traces. The primary server periodically evaluates the performance of candidate strategies for each document (for the previous period) and selects the best as the one that had the smallest cost.

The Web trace used in our experiments covers the requests and updates made to the documents hosted in our Web server from June 2002 to March 2003. Numerical details about the trace are given in Table 1. We perform our evaluations only for objects that receive more than 100 requests a week, reducing the total number of objects to be evaluated in the order of thousands. We adopt the no replication (NR) strategy for the rest. We fix the adaptation period of the server to one day.

3 The Need for Dynamic Adaptation

In this section, we demonstrate the need for continuous dynamic adaptation of replication strategies for Web documents. To do so, we measure the proportion of documents that change their strategy over the duration of the traces, determined by Adaptation Percentage (AP). AP is defined as the ratio of the total number of adaptations observed and the total number of possible adaptations. For example, in a 7 day period, if a document makes 2 adaptations out of the maximum possible 6 adaptations, then its AP is 33%. This metric shows the rate at which an object switches its strategy dynamically. A higher value of AP implies that the objects dynamically switch their strategy more often thereby showing a clear need for continuous adaptation.

Table 1: Trace Characeteristics

Number of days	273
Number of GET requests	78,049,912
Number of objects	2,185,896
Number of updates	156,721
Number of unique clients	1,252,779
Number of different ASes	2853

We performed a complete evaluation of the documents receiving at least 100 requests/week in our traces and plotted the average AP over all these documents (aggregated every week). The results are given in Figure 1. As seen in the figure, AP varies from 5% to 50%. This figure shows that the objects switch their strategy often, though the rate of adaptation varies over a period of time. This clearly demonstrates the need for continuous dynamic adaptation of strategies, if the adopted strategy for an object needs to remain optimal.

We further evaluated the need for continuous dynamic adaptation by comparing the performance of a system that would associate a strategy to each document once and never adapt (*Non-adaptive*), to one which would periodically adapt to the current-best strategy associations (*Adaptive*). Results are given in Table 2. It can be seen that the *Adaptive* selection method outperforms its *Non-adaptive* counterpart, according to all metrics simultaneously.

From these experiments, it can be seen that documents need to continuously adapt strategies to

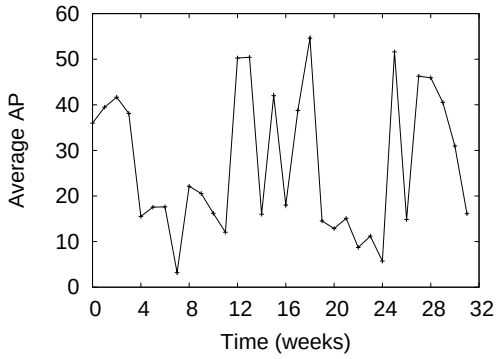

Figure 1: Time plot of AP of documents (aggregated every week) for the entire length of traces

Table 2: Performance of *Non-adaptive* and *Adaptive* selection methods given in terms of (i) Total Client latency (TCL) for all requests, (ii) number of stale documents delivered to clients (NS) and (iii) network usage bandwidth (BW)

Strategy	TCL(hrs)	NS	BW (GB)
Non-adaptive	37.5	11	13.1
Adaptive	31.3	3	12.7

maintain optimal performance, even for a simple Web server like ours. We believe that continuous dynamic adaptation will be even more beneficial for bigger replicated Web services handling more varying access patterns.

4 Strategy Selection Heuristics

In this section, we propose two selection heuristics, *Simple* and *Transition*, that aim to reduce the simulation cost by reducing the number of candidate strategies evaluated during the selection process.

4.1 Performance Evaluation Metrics

To evaluate the performance of strategy selection heuristics, we compared the selection accuracy and computational gain in comparison to the full evaluation method (evaluating all candidate strategies every period). We evaluate the performance of selection heuristics with the following metrics: (i) **Speedup**: This is defined as the ratio of total number of candidate strategies evaluated by the full evaluation method to that of the selection heuristic; (ii) **Accuracy**: This is defined as the percentage of times when the heuristic has selected the same strategy as the full evaluation strategy; and (iii) Average Worst Case Ratio (**AWCR**): This metric indicates how bad a strategy is when the heuristic has made a non-optimal selection. The AWCR of a heuristic is computed as the average of WCRs for all non-optimal selections in an adaptation period, where WCR is defined as follows:
$WCR(t) = \frac{|(cost(selected,t) - cost(best,t))|}{|(cost(worst,t) - cost(best,t))|}$
AWCR can vary from 0 to 1. The greater its value, the worse is its choice of strategy.

4.2 Selection Heuristics

The basic assumption behind the *Simple* heuristic is that a significant change in the request or update rate of a document can indicate the need for a strategy re-evaluation. In such cases, a full evaluation is performed. Otherwise, the current strategy is retained. Our evaluations showed that this heuristic yields poor accuracy with a very little gain in speedup. We conclude that more sophisticated heuristic is necessary to give a better speedup without much loss in accuracy.

The *Transition* heuristic tries to predict the likely strategy transitions and evaluates only the most promising ones. It aims to gain speedup by evaluating only this subset of strategies instead of the entire set. Obviously, the size and constituents of the selected subset of strategies to evaluate determines the speedup and accuracy of the heuristic. The smaller the size of this subset, the higher the speedup. On the other hand, the heuristic will find the optimal strategy only if it belongs to the evaluated subset of strategies.

Table 3: Performance of *Transition* for different values of *y*

y%	Accuracy	Speedup	AWCR
5%	97.8%	8	0.01
10%	95.5%	12	0.03
15%	93.4%	13	0.06
20%	92.2%	15	0.08
25%	88%	15.5	0.10

Table 4: Length of phase 1 vs. Accuracy

No. of days	Accuracy	AWCR
5	90.7%	0.09
7	95.5%	0.03
9	96%	0.03
14	96%	0.03

The *Transition* heuristic works in two phases. In the first phase, full evaluations are performed on the traces to build the *transition graph*. This graph captures the history of transitions between different strategies. It is a weighted directed graph, whose nodes represent the candidate strategies and weights are the number of observed transitions between strategies. No speedup is gained in this phase as full evaluations are performed.

The second phase of the heuristic uses the transition graph built during the first phase to determine the likely subset of strategies that need to be evaluated to perform strategy selection. This subset is determined as the set of target strategies whose estimated transition probability is greater than *y*%. The accuracy of this heuristic depends on two factors: (i) the value of *y* and (ii) the duration of phase 1.

Effect of *y*: We evaluated the accuracy and speedup of this heuristic for different values of *y* with a transition graph built from a week-long trace. The results are presented in Table 3. As could be expected, the speedup of the heuristic increases when *y* increases, since less strategies are evaluated. At the same time, the accuracy decreases and AWCR increases. When $y = 10$, we obtain speedup of 12, with only a little loss in accuracy. Thus, *Transition* strategy drastically reduces the simulation cost while making very good strategy selections.

Effect of the length of Phase 1: We evaluated the accuracy of the heuristic for different durations of phase 1 with *y* fixed at 10%. Results are given in Table 4. As seen from the table, a transition graph built out of just 5-day traces already leads to a an accuracy of 90.7%. Increasing the size of transition graph leads to better accuracy, however the gain stabilizes around 7 days. This corresponds to 6 full evaluations performed over thousands of documents. These data are representative enough to account for most future transition patterns.

An important issue in using *Transition* is to determine how often to rebuild the transition graph. In our experiments, we did not observe a degradation in accuracy over our 9 months traces with a transition graph built from one week. Hence, we feel that this transition graph needs to be rebuilt only rarely.

One of the shortcomings of this heuristic is its inability to handle emergencies like flash crowds as the pre-built transition graph does not cover such drastic changes in access patterns. Such scenarios might call for fast detection of the onset of emergencies (also determining the access patterns of emergency) and triggering a strategy selection mechanism that evaluates only a small set of candidate strategies that can perform well in the given scenario.

5 Related work

A large number of proposals have been made in the past to improve the quality of Web services. Cache consistency protocols such as Alex [3] and TTL policies aim to improve the scalability of the Web. Invalidation strategies have been proposed to maintain strong consistency at relatively low cost in terms of delay and traffic [2]. Several replication strategies have been proposed in the past.

Radar uses a dynamic replication protocol that allows dynamic creation/deletion of replicas based on the clients' access patterns [9]. In [6], the authors propose replication protocols that determine the number and location of replicas to reduce the access latency while taking the server's storage constraints into account. All these systems adopt a single strategy or single family of strategies for all documents, possibly with a tunable parameter. However, our earlier work advocated the simultaneous use of multiple strategies.

A number of systems select strategies on a per-document basis. In [1], the authors propose a protocol that dynamically adapts between variants of push and pull strategies on a per-document basis. Similarly, [4] proposes an adaptive lease protocol that switches dynamically between push and pull strategies. Finally, in [5], the author propose a protocol that chooses between different consistency mechanisms, invalidation or (update) propagation, on a per-document basis, based on the document's past access and update patterns. These protocols perform a per-document strategy selection similar to ours but they are inherently limited to a small set of single family of strategies (e.g., push or pull, invalidation or propagation) and cannot incorporate different families of strategies as done in our system.

6 Conclusions and Future Work

The need for dynamically selecting a strategy on a per-document basis was shown in our earlier research. In this paper, we demonstrate the need for continuous dynamic adaptation of strategies with experiments performed on the traces of our department Web server. We find that continuous adaptation is beneficial even for seemingly relative stable access patterns. As a second contribution of this paper, we proposed two heuristics, *Simple* and *Transition* to perform this continuous adaptation with reduced simulation cost. In our experiments, we find that the *Transition* heuristic performs better than its Simple counterpart, both in terms of accuracy and speedup. We conclude that, in our traces, evaluating strategies based on their past transition patterns is a good solution that yields high speedup with high accuracy.

Another way of reducing simulation overhead would be to perform *object clustering*, i.e., to group objects with similar access and update patterns. We are also investigating schemes to handle emergency situations such as flash crowds in our system.

References

[1] M. Bhide, P. Deolasee, A. Katkar, A. Panchbudhe, K. Ramamritham, and P. Shenoy, *Adaptive Push-Pull: Disseminating Dynamic Web Data* , IEEE Transactions on Computers **51** (2002), no. 6, 652–668.

[2] P. Cao and C. Liu, *Maintaining Strong Cache Consistency in the World Wide Web*, IEEE Transactions on Computers **47** (1998), no. 4, 445–457.

[3] V. Cate, *Alex – A Global File System*, File Systems Workshop (Berkeley, CA), USENIX, USENIX, May 1992, pp. 1–11.

[4] V. Duvvuri, P. Shenoy, and R. Tewari, *Adaptive Leases: A Strong Consistency Mechanism for the World Wide Web*, 19th INFOCOM Conference, IEEE Computer Society Press, March 2000, pp. 834–843.

[5] Z. Fei, *A Novel Approach to Managing Consistency in Content Distribution Networks*, 6th Web Caching Workshop, June 2001, pp. 71–86.

[6] J. Kangasharju, James Roberts, and K.W. Ross, *Object Replication Strategies in Content Distribution Networks*, 6th Web Caching Workshop, June 2001.

[7] G. Pierre, I. Kuz, M. van Steen, and A.S. Tanenbaum, *Differentiated Strategies for Replicating Web Documents*, Computer Communications **24** (2001), no. 2, 232–240.

[8] G. Pierre, M. van Steen, and A.S. Tanenbaum, *Dynamically Selecting Optimal Distribution Strategies for Web Documents*, IEEE Transactions on Computers **51** (2002), no. 6, 637–651.

[9] M. Rabinovich and A. Aggarwal, *Radar: A Scalable Architecture for a Global Web Hosting Service*, Computer Networks **31** (1999), no. 11–16, 1545–1561.