

# A Survey of Cache/Proxy for Transparent Data Replication

Yingwei Jin

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

Wenyu Qu

Institute of Industrial Science, The University of Tokyo  
4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505, Japan

Keqiu Li

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

## Abstract

Web caching is an important technology for reducing Internet access latency, alleviating network traffic, and spreading server load. An important issue that affects the performance of web caching is caching architecture, in which several caches are usually federated for making caches cooperate on a large scale and effectively increase the cache population. Transparent data replication is such a caching architecture, which has shown to be easy to manage and have good system scalability. In this paper, we present an survey of state-of-the-art techniques of cache/proxy placement for transparent data replication.

## 1 Introduction

The World Wide Web has become the most successful application over the Internet since it provides simple access to a wide range of information and services. However, the exponential growth in its popularity is leading to a number of performance problems, such as network congestion, server overloading, etc. To overcome these problems, an effective method is to apply the technology of web caching by which web objects are cached at various components in the networks. It has been shown that web caching can reduce Internet access latency, alleviate network traffic, and spread server load. A survey of web caching schemes for the

Internet can be found in [37]. There are a number of factors that affects the performance of web caching, such as caching system architecture [33, 40], proxy placement [15, 18, 24], caching contents [7, 13], cache resolution/routing [11, 27], prefetching [12, 26, 30], cache placement and replacement [2, 28], cache coherency [5, 8, 17], web traffic characteristics [3, 6], and dynamic data caching [14, 35].

Cooperative caching [10] is an emerging paradigm in the design of scalable high-performance systems. Cache cooperation improves the performance of isolated caches, especially for caches with small cache populations. To make caches cooperate on a large scale and effectively increase the cache population, several caches are usually federated in caching architectures [32]. Currently, there are two common caching architectures that are used to implement a large scale cache cooperation scheme, including hierarchical caching [9] and distributed caching [29, 36]. Both hierarchical caching and distributed caching are already a fact of life in much of the Internet [1]. With hierarchical caching, caches are placed at different network levels. At the bottom level of the hierarchy there are client caches. When a request is not satisfied by a client cache, the request is redirected to the institutional cache. If the document is not present at the institutional level, the request travels to the regional cache, which in turn forward unsatisfied requests to the national cache. If the document is not present at any cache level, the national cache

contacts directly the origin server. When the document is found, either at a cache or at the origin server, it travels down the hierarchy, leaving a copy at each of the intermediate caches. Further requests for the same document travel up the caching hierarchy until the requests find the document. With distributed caching, no intermediate caches are set up, and there are only institutional caches at the edge of the network that cooperate to serve each others' misses. Since there are no intermediate caches that store and centralize all documents requested by lower level caches, institutional caches need other mechanisms to share the documents they contain. Combination of hierarchical caching and distributed caching leads to hybrid caching, in which caches may cooperate with other caches at the same level or at a higher level using distributed caching.

Transparent data replication is a newly developed caching architecture. Many research has been done based on transparent data replication since it is a promising technique for improving the system performance of a large distributed network, which can overcome the management overheads that are incurred in early studies for identifying the optimal locations for the replica before each request is served [39,41]. For transparent data replication, caches are placed transparently to both the servers and the clients. Each cache intercepts any request that passes through its associated node, and either satisfies the request by sending the requested object to the client or forwards it upstream along the path to the server until it can be satisfied. Transparent data replication has a number of advantages [24,34,38]. First, it is transparent to both clients and servers. The additional bandwidth consumption and network delay for cache miss are minimized in that no request is detoured off the regular path. Moreover, it eliminates the extra overhead of locating the objects such as sending broadcast queries and maintaining directories. In general, there are two basic approaches for transparent data replication: en-route caching [4,32] and hierarchical caching [31,32]. With a hybrid transparent data replication model, it is formulated by combining both en-route and hierarchical caching. In this model, caches are organized in a hierarchical manner as in hierarchical caching, and each request from a client is routed on the access path from the client to the server as in en-route caching.

The performance of transparent data replication

mainly depends on cache/proxy placement and cache content management (object placement, cache replacement, and cache content consistency). In this survey, we present state-of-the-art techniques of cache/proxy placement for transparent data replication, which might be of great significance for solving the data management problem in the Knowledge Grid and Semantic Grid [42,43].

The rest of this paper is organized as follows. We describe the general model for cache/proxy placement in Section 2. Sections 3 and 4 focus on cache/proxy placement for linear and tree topology, respectively. Finally, we summarize our work and concludes this paper in Section 5.

## 2 Formal Model

We still represent the network by a graph denoted by  $G = (V, E)$ , where  $V = \{v_1, v_2, \dots, v_n\}$  is the set of nodes and  $E$  is the set edges. We also use  $c(v_i, v_j)$  to denote the cost incurred on the edge  $(v_i, v_j)$  and  $F_i$  to denote the the overall traffic accessing the server that has to pass through node  $v_i$ .

Obviously, placing a cahce/proxy at a node will make some requests that previously pass through this node are now satisfied at node  $v$ . This will introduce some cost saving, which is denoted by  $s(v)$ . Although placing a cache/proxy at a node will also generate some cost (e.g., maintenance cost and hardware cost), it is really difficult to combine this cost with the cost saving defined above. In addition, it is impossible to place as many proxies as possible. Therefore, the general model for the problem of cache/proxy placement can be formalized as follows:

$$\max_{|P| \leq M} \Phi(G, P) = \max_{|P| \leq M} \left\{ \sum_{v \in P} s(v) \right\}, \quad (1)$$

where  $P \subseteq V$  is a subset of nodes at each of which a cache/proxy is placed,  $M$  is the maximal number of proxies to be placed, and  $\Phi(G, P)$  is the relevant total cost gain generated. Therefore our objective is to find  $P^*$  such that the total cost gain is maximized, i.e.,  $\Phi(G, P^*) = \max_{|P| \leq M} \Phi(G, P)$ .

### 3 Cache/Proxy Placement for Linear Topology

In this section, we investigate the problem of cache/proxy for linear topology as shown in Figure 1, where  $v_0$  denotes the server.

First, we discuss the case of placing  $M$  caches/proxies. Obviously, the problem can be formulated as an optimization problem as follows:

$$\max_{|P|=M} \Phi(G, P) = \max_{|P|=M} \left\{ \sum_{i=1}^M (F_{k_i} - F_{k_{i-1}}) c(v_{k_i}, v_0) \right\}, \quad (2)$$

where  $P = \{v_{k_1}, v_{k_2}, \dots, v_{k_M}\} \subseteq V$  is a subset of nodes at each of which a cache/proxy is place,  $\Phi(G, P)$  is the relevant total cost gain generated. Therefore the objective is to find  $P^*$  such that the total cost gain is maximized, i.e.,  $\Phi(G, P^*) = \max_P \Phi(G, P)$ . We also use  $OPT(M, n)$  to denote the maximal gain for placing  $M$  caches/proxies from  $n$  nodes in the network.

Now we have the following algorithm for solving (2).

**Algorithm 4:** Cache/Proxy Placement for Linear Topology

```

1. Initialization
for  $i = 2$  to  $n$  do
   $k=1$ 
  for  $j = 2$  to  $n - 1$  do
    if  $F(j)Q(j, n) > F(k)Q(k, n)$  then
       $k = j$ 
    end if
  end for
end for
Output  $OPT(1, n) = F(k)Q(k, n)$ 

2. Filling in the array
for  $m = 2$  to  $M$  do
  for  $i = m + 1$  to  $n + 1$  do
     $k=m$ 
    for  $j = m + 1$  to  $n - 1$  do
      if  $OPT(m - 1, j) + F(j)Q(j, n) > OPT(m - 1, k) + F(k)Q(k, n)$  then
         $k = j$ 
      end if
    end for
  end for
end for
Output  $OPT(m, n) = OPT(m - 1, k) +$ 

```

$F(k)Q(k, n)$

In Algorithm 4,  $Q(i, j) = F(i) - F(j)$  and  $F(n + 1) = 0$ . It can be verified that the time complexity of Algorithm 4 is  $O(n^2M)$ , where  $n$  is the number of nodes in the network and  $M$  is the number of caches/proxies placed [18].

Based on the analysis of multimedia object caching [23], we can extend the relevant solution to solve the problem of transcoding proxy placement for linear topology. This problem can be formulated as an optimization problem as follows:

$$\max_{(v_1, v_2, \dots, v_K)} H(n, \alpha : v_1, v_2, \dots, v_k) = \sum_{j=1}^k \sum_{h=1}^l \sum_{A_{h,k} \in D(B_{h,v_j})} [(f_{A_{h,k},v_j} - f_{A_{h,k},v_{j+1}}) \cdot m(A_{h,k}, v_j) - \alpha] \quad (3)$$

An optimal solution for (3) can be easily obtained by extending Theorem 3 in [23]. Before presenting a dynamic programming-based solution for (3), we give the following definition.

**Definition 1** Define  $H_n^*$  to be the maximum aggregate cost gain of  $H(n, \alpha : v_1, v_2, \dots, v_k)$  obtained by solving the  $n$ -optimization problem and  $I_n$  the maximum index in the optimal solution. If the optimal solution is an empty set, define  $I_n = -1$ .

Obviously, we have  $I_0 = -1$  and  $H_0^* = 0$ . Therefore, if  $I_r > 0$ ,

$$H_{I_r} = H_{I_r-1} + \sum_{h=1}^l \sum_{A_{h,k} \in D(B_{h,v_{I_r}})} [(f_{A_{h,k},v_{I_r}} - f_{A_{h,k},v_{I_r+1}}) \cdot m(A_{h,k}, v_{I_r+r}) - \alpha] \quad (4)$$

Therefore, we can check all possible locations of  $I_r$  ( $0 \leq r \leq n$ ) and select the one that maximizes  $H(r : v_1, v_2, \dots, v_k)$ . So we have

$$\begin{cases} H_0^* = 0 \\ H_r^* = \max_{1 \leq v_i \leq r} \{0, H_{v_i-1}^* + \sum_{h=1}^l \sum_{A_{h,k} \in D(B_{h,v_i})} [(f_{A_{h,k},v_i} - f_{A_{h,k},v_{i+1}}) \cdot m(A_{h,k}, v_i) - \alpha]\} \end{cases}$$

and

$$\begin{cases} I_0 = -1 \\ I_r = \begin{cases} -1 & \text{if } H_r^* = 0 \\ v & \text{if } H_r^* = H_{v-1}^* + \sum_{A_{h,k} \in D(B_{h,v})} [(f_{A_{h,k},v} - f_{A_{h,k},v+1}) \cdot m(A_{h,k}, v) - \alpha] \end{cases} \end{cases}$$



Figure 1: Cache/Proxy Placement for Linear Topology

Based on the recurrences above, (3) can be solved using dynamic programming. After computing  $H_n^*$  and  $I_n$ , we can start from  $v_r = I_n$  and obtain all the locations iteratively. It can be proved that the time complexity of the dynamic programming solution is  $O(n^2lm)$ , where  $n$  is the number of nodes,  $l$  is the number of the multimedia objects, and  $m$  the maximum number of versions for all the multimedia objects [22].

## 4 Cache/Proxy Placement for Tree Networks

In this section, we present several solution for (1) when the network topology is a tree.

Based on theorems 1 and 2 in [23], we can find an optimal solution for (2) for tree networks [20].

In [19], the authors proposed an optimal algorithm for the problem as described in (2) for tree network. In [16], the authors proposed an optimal solution for the problem as described in (2) for tree network with consideration of read and update operations. The idea is also similar to that for solving the problem of web object caching for tree networks as introduced in [23].

In [24, 25], the authors formulated this problem from another point of view. However, it can be transformed to the problem as formalized in (1). In this paper, the authors also addressed the case of multiple servers and proposed an algorithm for finding the optimal solution with the time complexity of  $O(n^3M)$ , where  $n$  is the number of nodes in the network and  $M$  is the number of caches/proxies to be placed.

In the following, we discuss the problem of transcoding proxy placement for tree network. Based on the analysis for object caching for multimedia ob-

jects in [23], this problem can be defined as follows:

$$\max_P \Phi(G, P) = \max_P \sum_{v \in P} \sum_{i=1}^l \sum_{A_{i,x} \in \theta(H_{i,v})} \left( f_{A_{i,x},v} - \sum_{z \in B_v^-(A_{i,x})} f_{A_{i,x},z} \right) m(A_{i,x}, v) - l(H_{i,v}) \quad (5)$$

where  $P \subseteq V$ .

Obviously, our objective is to compute the locations for placing transcoding proxies in a subset of nodes  $P$  that maximizes  $\Phi(G, P)$ . Similarly, we can formulate the proxy placement problem of placing  $M$  transcoding proxies by adding a constraint that  $|P| = M$ .

Based on theorems 1 and 2 in [23], the problem, i.e. 5, can be solved using dynamic programming with the following recurrences.

- Suppose that  $v \in V$  and  $C(v) = \{v_1, v_2, \dots, v_{t_1}\}$ . If  $t_1 = 0$ , then  $P_v^* = \phi$ ; otherwise,  $P_v^* = \cup_{i=1}^{t_1} P_{v,v_i}^*$ .
- Suppose that  $u \in D(v)$  and  $C(u) = \{u_1, u_2, \dots, u_{t_2}\}$ . If  $t_2 = 0$ , then

$$P_{v,u}^* = \begin{cases} \{u\} & \text{if } \pi(u, v) \geq 0 \\ \{\phi\} & \text{Otherwise} \end{cases}$$

$$\text{where } \pi(u, v) = \sum_{i=1}^l \sum_{A_{i,x} \in \theta(H_{i,u})} f_{A_{i,x},u}(c(v, u) + w(H_{i,u}, A_{i,x})).$$

Otherwise,

$$P_{v,u}^* = \begin{cases} \cup_{i=1}^{t_2} P_{v,u_i}^* & \text{if } \rho(u, v) \geq 0 \\ P_u^* \cup \{u\} & \text{Otherwise} \end{cases}$$

$$\text{where } \rho(u, v) = G(T_{v,u}, \cup_{i=1}^{t_2} P_{v,u_i}^*) - G(T_{v,u}, P_u^* \cup \{u\}).$$

It is easy to see that the time complexity of the dynamic programming-based algorithm is  $O(n^2lm)$ , where  $n$  is the number of nodes,  $l$  is the number of the multimedia objects, and  $m$  the maximum number of versions for all the multimedia objects [21].

## 5 Conclusions

Web caching is recognized as one of the effective techniques for reducing the user access latency and alleviating the network traffic. Many caching architectures have been proposed in the literature to cater different network environments. In this paper, we presented an overview of state-of-the-art techniques of cache/proxy placement for transparent data replication that has been shown to be an effective caching architecture.

## References

- [1] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm. *World Wide Web Caching: The Application-Level View of the Internet*. IEEE Communication Magazine, 170-178, 1997.
- [2] A. Balamash and M. Krunz. *An Overview of Web Caching Replacement Algorithms*. IEEE Communications Surveys, 6(2), 44-56, 2004.
- [3] P. Barford, A. Bestavros, A. Bradley, and M. Crovella. *Changes in Web Client Access Patterns Characteristics and Caching Implications*. World Wide Web, 2(1-2), 15-28, 1999.
- [4] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura. *Self-Organizing Wide-Area Network Caches*. Proc. of IEEE INFOCOM'98, 600-608, 1998.
- [5] M. Bhide, P. Deolasee, A. Katkar, and A. Panchbudhe. *Adaptive Push-Pull: Disseminating Dynamic Web Data*. IEEE TC, 51(6), 652-667, 2002.
- [6] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. *Web Caching and Zipf-like Distributions: Evidence and Implications*. Proc. IEEE INFOCOM'99, 126-134, 1999.
- [7] R. Caceres, F. Douglass, A. Feldmann, G. Glass, and M. Rabinovich. *Web Proxy Caching: The Devil is in the Details*. ACM PER, 26(3), 11-15, 1998.
- [8] P. Cao and C. Liu. *Maintaining Strong Cache Consistency in the World Wide Web*. IEEE TC, 47(4), 445-457, 1998.
- [9] A. Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz, and K. Worrell. *A Hierarchical Internet Object Cache*. Proc. of USENIX1996, 22-26, 1996.
- [10] M. D. Dahlin, R. Y. Wang, T. E. Anderson, and D. A. Patterson. *Cooperative Caching: Using Remote Client Memory to Improve File System Performance*. Proc. of OSDI'94, 267-280, 1994.
- [11] L. Fan, P. Cao, J. Almeida and A. Z. Broder. *Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol*. Proc. of SIGMETRICS'99, 178-187, 1999.
- [12] L. Fan, Q. Jacobson, P. Cao, and W. Lin. *Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance*. IEEE/ACM TN, 5(3), 281-293, 2000.
- [13] A. Feldmann, R. Caceres, F. Douglass, G. Glass, and M. Rabinovich. *Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments*. Proc. of IEEE INFOCOM'99, 107-116, 1999.
- [14] A. Iyengar and J. Challenger. *Improving Web Server Performance by Caching Dynamic Data*. Proc. of the USENIX Symposium on Internet Technologies and Systems, 1997.
- [15] X. Jia, D. Li, X. Hu, and D. Du. *Optimal Placement of Web Proxies for Replicated Web Servers in the Internet*. The Computer Journal, 44(5), 329-339, 2001.
- [16] X. Jia, D. Li, X. Hu, W. Wu, and D. Du. *Placement of Web-Server Proxies with Consideration of Read and Update Cost on the Internet*. The Computer Journal, 46(4), 378-390, 2003.
- [17] R. Ladin, B. Liskov, L. Shriram, and S. Ghemawat. *Providing Availability Using Lazy Replication*. ACM TOCS, 10(4), 360-391, 1992.

- [18] B. Li, X. Deng, M. J. Golin, and K. Sohraby. *On the Optimal Placement of Web Proxies in the Internet: The Linear Topology*. Proc. of HPN'98, 21-25, 1998.
- [19] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby. *On the Optimal Placement of Web Proxies in the Internet*. Proc. of IEEE INFOCOM'1999, 1282-1290, 1999.
- [20] K. Li and H. Shen. *Optimal Methods for Proxy Placement in Coordinated En-Route Web Caching*. IEICE Trans. on Communications, E88-B(4), 1458-1466, 2005.
- [21] K. Li and H. Shen: *Optimal Proxy Placement for Coordinated En-Route Transcoding Proxy Caching*. IEICE Trans. on Information and Systems, E87-D(12), 2689-2696, 2004.
- [22] K. Li and H. Shen: *Proxy Placement Problem for Coordinated En-Route Transcoding Proxy Caching*. IJCSSE, 19(5), 95-103, 2004.
- [23] K. Li, T. Nanya, B. Jiang, and W. Qu. *State-of-Art Techniques for Object Caching over the Internet*. Proc. of IMSCCS'06, 199-206, 2006.
- [24] P. Krishnan, D. Raz, and Y. Shavitt. *The Cache Location Problem*. IEEE/ACM TN, 8(5), 568-582, 2000.
- [25] P. Krishnan, D. Raz, and Y. Shavitt. *Transparent En-Route Cache Location in Regular Network Topologies*. Proc. of DIMACS, 1998.
- [26] T. M. Kroeger, D. D. E. Long, and J. C. Mogul. *Exploring the Bounds of Web Latency Reduction from Caching and Prefetching*. Proc. of USITS'97, 13-22, 1997.
- [27] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd and V. Jacobson. *Adaptive Web Caching: Towards a New Caching Architecture*. Computer Network and ISDN Systems, 30(22-23), 2169-2177, 1998.
- [28] S. Podlipnig and L. Boszormenyi. *A Survey of Web cache Replacement Strategies*. ACM Computing Surveys, 35(4), 374-398, 2003.
- [29] D. Povey and J. Harrison. *A Distributed Internet Cache*. Proc. of the 20th Australian Computer science Conference, 1997.
- [30] V. N. Padmanabhan and J. C. Mogul. *Using Predictive Prefetching to Improve World Wide Web Latency*. Proc. of the ACM SIGCOMM '96 Conference, 1996.
- [31] M. Rabinovich and O. Spatscheck. *Web Caching and Replication*. Addison-Wesley, 2002.
- [32] P. Rodriguez and S. Sibal. *Spread: Scalable Platform for Reliable and Efficient Distribution*. Computer Networks, 33, 33-49, 2000.
- [33] P. Rodriguez, C. Spanner, and E. W. Biersack. *Analysis of Web Caching Architectures: Hierarchical and Distributed Caching*. IEEE/ACM TN, 9(4), 404-418, 2001.
- [34] X. Tang and S. T. Chanson. *Coordinated En-Route Web Caching*. IEEE TC, 51(6), 595-607, 2002.
- [35] X. Tang and S. T. Chanson. *The Minimal Cost Distribution Tree Problem for Recursive Expiration-Based Consistency Management*. IEEE TPDS, 15(3), 214-227, 2004.
- [36] X. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay. *Design Considerations for Distributed Caching on the Internet*. Proc. of ICDCS, 273-284, 1999.
- [37] J. Wang. *A Survey of Web Caching Schemes for the Internet*. ACM CCR, 29(5), 36-46, 1999.
- [38] D. Wessels and K. Claffy. *ICP and the Squid Web Cache*. IEEE JSAC, 16(3), 345-357, 1998.
- [39] O. Wolfson and A. Milo. *The Multicast Policy and its Relationship to Replicated Data Placement*. ACM TODS, 16(1), 181-205, 1991.
- [40] L. Xiao, X. Zhang, A. Andrzejak, and S. Chen. *Building a Large and Efficient Hybrid Peer-to-Peer Internet Caching System*. IEEE TKDE, 16(6), 754-769, 2004.
- [41] J. Xu, B. Li, and D. L. Li. *Placement Problems for Transparent Data Replication Proxy Services*. IEEE JASC, 20(7), 1383-1398, 2002.
- [42] H. Zhuge. *China's E-Science Knowledge Grid Environment*. IEEE Intelligent Systems, 19(1), 13-17, 2004.
- [43] H. Zhuge. *The Knowledge Grid*. World Scientific Publishing Co., Singapore, 2004.