Cooperative Determination on Cache Replacement Candidates for Transcoding Proxy Caching*

Keqiu Li[1], Hong Shen[1,2], and Francis Y.L. Chin[3]

[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 Technology,
University of Science and Technology of China,
Hefei, Anhui 230026, China
[3] Department of Computer Science and Information Systems,
University of Hong Kong, Pokfulam Road, Hong Kong

Abstract. Transcoding proxy caching is an important technology for improving the services over Internet, especially in the environment of mobile computing systems. In this paper, we address cooperative determination on cache replacement candidates for transcoding proxies. An original model which determines cache replacement candidates on all candidate nodes in a coordinated fashion with the objective of minimizing the total cost loss is proposed. We formulate this problem as an optimization problem and present a low-cost optimal solution for deciding cache replacement candidates.

1 Introduction

Web caching is an important technology for improving the services over Internet. Since the majority of web objects are static, caching them at various network components (e.g., client browser, proxy server) provides a natural way of decreasing network traffic. Moreover, web caching can also reduce users' access latency and alleviate server load.

A key factor that affects the performance of web caching is the cache replacement policy, which is a decision for evicting an object currently in the cache to make room for a new object. A number of cache replacement policies, which attempt to optimize various performance metrics, such as hit ratio, byte hit ratio, delay saving ratio, etc., have been proposed in the literature. However, all these polices are local replacement models that determine cache replacement candidates from the view of only a single node. Furthermore, they become inefficient in transcoding proxies due to the new emerging factors in the transcoding

* This work was partially supported by Japan Society for the Promotion of Science (JSPS) under its General Research Scheme **B** Grant No. 14380139). Corresponding author H. Shen (shen@jaist.ac.jp).

X. Lu and W. Zhao (Eds.): ICCNMC 2005, LNCS 3619, pp. 178–187, 2005.

proxy (e.g., the additional delay caused by transcoding, different sizes and reference rates for different versions of a multimedia object) and the aggregate effect of caching multiple versions of the same multimedia object. Although the authors have elaborated these issues in [1], they considered the cache replacement problem at only a single node. Cooperative caching, in which caches cooperate in serving each other's requests and making storage decisions, is a powerful paradigm to improve cache effectiveness [3,6]. There are two orthogonal issues to cooperative caching: object location (i.e., finding nearby copies of objects) and object management (i.e., coordinating the caches while making storage decisions). The object location problem has been widely studied [2,4,8]. Efficient coordinated object management algorithms are crucial to the performance of a cooperative caching system, which can be divided into two type of algorithms: placement and replacement algorithms. There are a number of research on finding efficient solutions for cooperative object placement [5,7,9]. However, there is little work done on finding efficient solutions for cooperative object replacement. Due to the interrelationship among different versions of the same multimedia object, cooperative caching in transcoding proxies becomes more important and complicated. We claim that this is very significant for the performance of a cooperative caching system since when a updated version is to be cached, an efficient replacement policy should decide cache replacement candidates by considering the cooperation of all the nodes on the path from the server to the client. Another important point is that the replacement decision on each node should be beneficial, i.e., the profit gained by caching the new object should be no less than the profit lost by removing some objects from the cache to make room for the new object. As the transcoding proxy is attracting an increasing amount of attention in the environment of mobile computing, it is noted that new efficient cache replacement policies are required for these transcoding proxies. In this paper, we address cooperative determination on cache replacement candidates for transcoding proxies. We first propose an original model which determines cache replacement candidates among all candidate nodes in a coordinated fashion with the objective of minimizing the total cost loss. Moreover, we formulate this problem of an optimization problem and present a low-cost optimal solution for deciding cache replacement candidates.

The rest of this paper is organized as follows: Section 2 introduces some preliminaries. We formulate the problem and present an optimal solution for this problem in Section 3. Finally, we conclude this paper in Section 4.

2 Preliminaries

We first introduce multimedia object transcoding in Section 2.1, and then notations and definitions in Section 2.2.

2.1 Multimedia Object Transcoding

Transcoding is used to transform a multimedia object from one form to another, frequently trading off object fidelity for size, i.e., the process of converting a

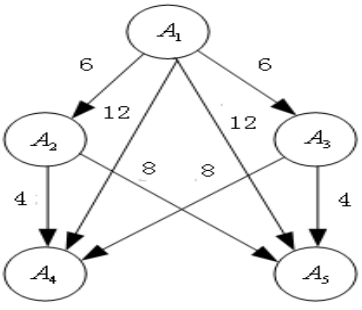

Fig. 1. An Example of A Weighted Transcoding Graph

media file or object from one format to another. Transcoding is often used to convert video formats (i.e., Beta to VHS, VHS to QuickTime, QuickTime to MPEG). But it is also used to fit HTML files and graphics files to the unique constraints of mobile devices and other Web-enabled products. These devices usually have smaller screen sizes, lower memory, and slower bandwidth rates. In this scenario, transcoding is performed by a transcoding proxy server or device, which receives the requested document or file and uses a specified annotation to adapt it to the client.

The relationship among different versions of a multimedia object can be expressed by a weighted transcoding graph. An example of such a graph is shown in Figure 1, where the original version A_1 can be transcoded to each of the less detailed versions A_2, A_3, A_4, and A_5. It should be noted that not every A_i can be transcoded to A_j since it is possible that A_i does not contain enough content information for the transcoding from A_i to A_j. In our example, transcoding can not be executed between A_4 and A_5 due to insufficient content information. The transcoding cost of a multimedia object from A_i to A_j is denoted by $w(i, j)$. The number beside each edge in Figure 1 is the transcoding cost from one version to another. For example, $w(1, 2) = 6$, and $w(3, 4) = 4$. $\phi(i)$ is the set of all the versions that can be transcoded from A_i, including A_i. For example, $\phi(1) = \{1, 2, 3, 4, 5\}$, $\phi(2) = \{2, 4, 5\}$, and $\phi(4) = \{4\}$. In this paper, we use G to denote a weighted transcoding graph.

2.2 Notations and Definitions

We model the network as a graph $G = (V, E)$ in this paper, where $V = \{v_0, v_1, \cdots, v_n\}$ is the set of nodes or vertices, and E is the set of edges or links. We assume that every node is associated with a cache with the same size B and there are m multimedia objects, i.e., $O_1, O_2, \cdots, O_l$, maintained by server v_0. For each multimedia object O_j, we assume that it has m_j versions: $O_{j,1}, O_{j,2}, \cdots, O_{j,m_j}$ and all versions have the same size. Thus, each node can hold at most B objects. We denote the set of objects cached at node v_i by $Y^i = \{A_1^i, A_2^i, \cdots, A_m^i\}$, where $A_j^i \subseteq \{O_{j,k_1}, O_{j,k2}, \cdots, O_{j,k_j}\}$ is the set of different versions of object O_j cached at node v_i. Obviously, $Y = \{Y^1, Y^2, \cdots, Y^n\}$

is the set of all objects cached. For each version of object O_j, we associate each link $(u, v) \in E$ a nonnegative cost $L_{j,k}(u, v)$, which is defined as the cost of sending a request for version $O_{j,k}$ and the relevant response over the link (u, v). In particular, $L_{j,k}(u, u) = 0$. If a request goes through multiple network links, the cost is the sum of the cost on all these links. The cost in our analysis is calculated from a general point of view. It can be different performance measures such as delay, bandwidth requirement, and access latency, or a combination of these measures. Let $r_{i,j,k}$ denote the request for $O_{j,k}$ at node v_i and $f_{i,j,k}$ be the frequency of $r_{i,j,k}$.

For notational tidiness, we omit argument j in all parameters and functions throughout the following analysis since our analysis is based on a specific object. For example, O_k denotes version k of object j, A^i is the set of different versions of object j cached at node v_i, $L_k(u, v)$ denotes the cost of sending a request for version O_k and the relevant response over the link (u, v), $r_{i,k}$ denotes the request for O_k at node v_i, and $f_{i,k}$ denotes the frequency of $r_{i,k}$. We also make the following assumptions.

- *Assumption* 1*:* $L_k(v_{i_1}, v_{i_2}) = (i_1 - i_2)L$ for all $1 \leq k \leq m$ as there are $i_1 - i_2$ links on the path between node v_{i_1} and node v_{i_2}, and the cost on each link for each version of O_j is L.
- *Assumption* 2*:* The transcoding graph is a linear array and the transcoding cost between any two adjacent versions is constant, i.e., $t(O_{k_1}, O_{k_2}) = \sum_{k=k_1}^{k_2-1} t(O_k, O_{k+1}) = (k_2 - k_1)^+ T$, where $x^+ = x$ if $x \geq 0$ else $x^+ = \infty$.
- *Assumption* 3*:* There exists some positive integer δ such that $(\delta - 1)T \leq L$, and $\delta T > L$. If there does not exist such a δ, i.e., $L \gg T$ or $T \gg L$. Obviously, these are two trivial cases.

3 Cooperative Cache Replacement for Transcoding Proxies

3.1 Problem Formulation

Before formulating the problem, we give some explanation on how the requests are served. As shown in Figure 2, a request goes along a routing path from the client (node v_n) to the server (node v_0). Note that any request $r_{i,k}$ could find the service from $S(r_{i,k})$, where $S(r_{i,k})$ denotes the serving object for $r_{i,k}$. Assume that $S(r_{i,k}) = O_{k_1} \in A^{i_1}$ with $k_1 \leq k$ and $i_1 \leq i$, then there may be the following ways of serving $r_{i,k}$ by $O_{k_1} \in A^{i_1}$.

- O_{k_1} is first sent from node v_{i_1} to node v_i and then transcoded to O_k at node v_i.
- O_{k_1} is first transcoded to O_k at node v_{i_1} and then O_k is sent from node v_{i_1} to node v_i.
- O_{k_1} is first sent from node v_{i_1} to node v_{i_2}, transcoded to O_k at node v_{i_2}, and then then O_k is sent from node v_{i_2} to node v_i.

– O_{k_1} is first sent from node v_{i_1} to node v_{i_2} and transcoded to O_{k_2} at node v_{i_2}, and then O_{k_2} is sent from node v_{i_2} to node v_{i_3} and transcoded to O_{k_3} at node v_{i_3}, then O_{k_3} is sent from node v_{i_3} to node v_i and transcoded to O_k at node v_i.

– ⋮

Fig. 2. System Model for Multimedia Object Caching

All these cases would cost the same under our cost model even though in practice. However, when a new or updated version of a multimedia object to be cache, denoted by O_{i_0}, is passing through each node between nodes $v_{i'}$ and v_i, it should be decided where O_{i_0} should be cached and which version should removed from the relevant cache to make room for it depending on how $r_{i,k}$ is served. Given X (i.e., the set of cached objects) and $O_{k'} \in A^{i'}$ $(i' \leq i)$. Let $d(r_{i,k}, O_{k'})$ denote the cost of serving $r_{i,k}$ by $O_{k'}$ at node $v_{i'}$. Then $d(r_{i,k}, O_{k'})$ is defined as follows:

$$d(r_{i,k}, O_{k'}) = (i - i')L + (k - k')^+ T \tag{1}$$

where $(x-y)^+ = \begin{cases} x-y & \text{if } x-y \geq 0 \\ 0 & \text{if } x-y < 0 \end{cases}$

Now we begin to formulate the problem addressed in this paper, i.e., determining where a new or updated version O_{i_0} should be cached among nodes $\{v_1, v_2, \cdots, v_n\}$ and which version of object j should be removed at that node to make room for O_{i_0} such that the total cost loss is minimized. Suppose that $P \subseteq V$ is the set of nodes at each of which $X_{i,k_i} \in A^i$ should be removed to make room for O_{i_0}, then this problem can be formally defined as follows:

$$L(P^*) = \min_{P \subseteq V} \{L(P)\} = \sum_{v_i \in P} (l(X_{i,k_i}) - g_i(O_{i_0})) \tag{2}$$

where $L(P)$ is the total relative cost loss, $l(X_{i,k_i})$ is the cost loss of removing X_{i,k_i} from node v_i, and $g_i(O_{i_0})$ is the cost saving of caching O_{i_0} at node v_i.

3.2 Dynamic Programming-Based Solution

Before presenting the solution, we evaluate the two items, i.e., $l(X_{i,k_i})$ and $g_i(O_{i_0})$, shown in Equation (2) in detail .

First, we begin with presenting a solution for finding the best way of serving $r_{i,k}$, i.e., finding $S(r_{i,k})$. Based on Equation (1), the cost of serving $r_{i,k}$, denoted by $c(r_{i,k})$, is defined as follows:

$$c(r_{i,k}) = \min \left\{ \min_{O_{k'} \in A^{i'},\ 1 \le i' \le i} d(r_{i,k}, O_{k'}), iL \right\} \tag{3}$$

Therefore, the object for serving $r_{i,k}$, denoted by $S(r_{i,k})$, is determined as follows:

$$S(r_{i,k}) = \begin{cases} O_{k'} \in A^{i'} & \text{if } c(r_{i,k}) \ge d(r_{i,k}, O_{k'}) \\ v_0 & \text{if } c(r_{i,k}) = iL \end{cases} \tag{4}$$

The following property will help us simplify the problem of finding the best way of serving $r_{i,k}$.

Theorem 1. *If both O_{k_1} and O_{k_2} are cached at node $v_{i'}$, then we have $d(r_{i,k}, O_{k_1}) < d(r_{i,k}, O_{k_2})$ for $k > k_1 > k_2$.*

Proof. Based on the definition of $d(r_{i,k}, O_k)$, we have $d(r_{i,k}, O_{k_1}) = (i - i')L + (k - k_1)^+ T$ and $d(r_{i,k}, O_{k_2}) = (i - i')L + (k - k_2)^+ T$. Since $(k - k_1)^+ < (k - k_2)^+$, we have $d(r_{i,k}, O_{k_1}) < d(r_{i,k}, O_{k_2})$. Hence, the theorem is proven.

From Theorem 1, we can see that for request $r_{i,k}$, we can consider only the least detailed version that can be transcoded to version k. Thus, Equation (3) can be simplified as follows:

$$c(r_{i,k}) = \min \left\{ \min_{1 \le i' \le i} d(r_{i,k}, O_{k^*}), iL \right\} \tag{5}$$

where O_{k^*} is the least detailed version of object j cached at node $v_{i'}$ that can be transcoded to version k.

It is easy to see that the time complexity for computing $S(r_{i,k})$ is $O(\log n)$, where n is the number of nodes in the network. So the total complexity for computing all $S(r_{i,k})$ ($1 \le i \le n$ and $1 \le k \le m$) is $O(mn \log n)$ since there are n nodes and object j has m different versions.

For each object $x \in X$, the set of requests served by x is expressed as $R(x) = \{r_{i,k} | S(r_{i,k}) = x\}$ and the total cost for the requests served by x is $C(x) = \sum_{r_{i,k} \in R(x)} f_{i,k} d(r_{i,k}, x)$. In this paper, we use R_s to denote the set of requests served by the server.

Regarding to $R(x)$, we have the following property.

Property 1. If $r_{i,k} \in R(x)$, then $r_{i',k'} \in R(x') \ \forall\ i' \le i$ and $k' \le k$.

Proof. Suppose that $x \in A^{i_1} = O_{k_1}$, $x' \in A^{i_2} = O_{k_2}$ and there exists $i' \le i$ and $k' \le k$ such that $r_{i',k'} \in R(O_{i_2})$. Since $S(r_{i',k'}) = x'$, we have $d(r_{i',k'}, x') \le d(r_{i',k'}, x)$. Therefore we have $(i' - i_2)L + (k' - k_2)T \le (i' - i_1)L + (k' - k_1)T$, i.e., $(i_2 - i_1)L + (k_2 - k_1)T \le 0$. Therefore we have $d(r_{i,k}, x) = (i - i_1)L + (k - k_1)T = (i - i_2)L + (k - k_2)T + (i_2 - i_1)L + (k_2 - k_1)T = d(r_{i,k}, x') + (i_2 - i_1)L + (k_2 - k_1)T \le d(r_{i,k}, x')$. So we have $S(r_{i,k}) = x'$, which contradicts $r_{i,k} \in R(x)$. Hence, the property is proven.

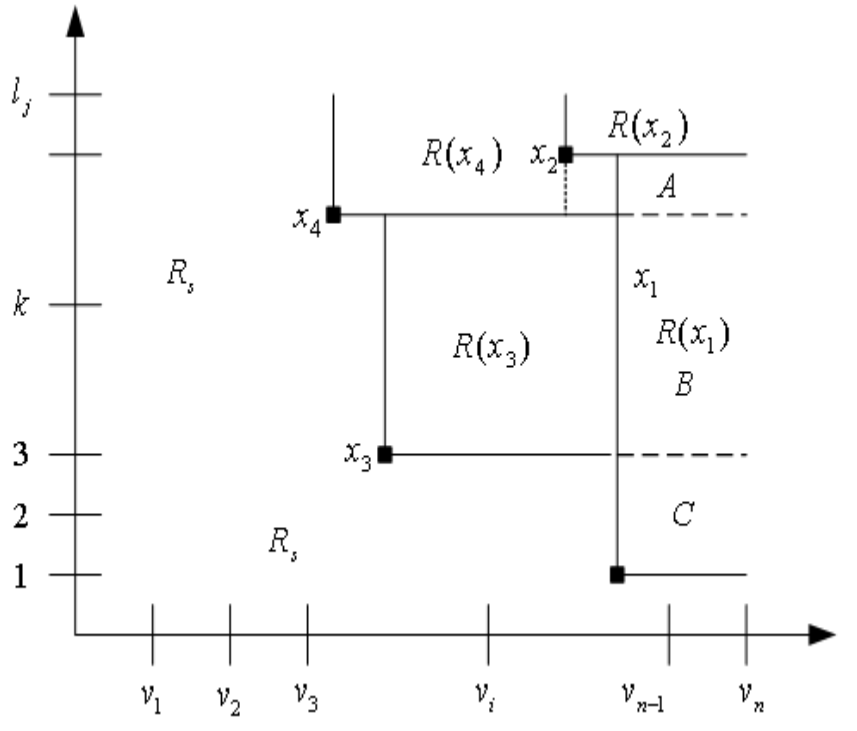

Fig. 3. Example for Calculating $l(x)$

From Property 1, we can see that $R(x)$ should be a region that can be divided into several rectangular regions. This can be seen from Figure 3. For example, $R(x_4)$ can be divided into two regions by the vertical broken line from x_2.

Regarding to calculating $l(X_{i,k_i})$, we first give the following theorem.

Theorem 2. *Suppose that only X_{i,k_i} is cached at node v_i, then we have* $l(X_{i,k_i}) = \sum_{r_{i,k} \in B_0} f_{i,k}[i \cdot L - d(r_{i,k}, X_{i,k_i})] + \sum_{i=1}^{n} \sum_{r_{i,k} \in B_i} f_{i,k}[d(r_{i,k}, X^i_{k_i}) - d(r_{i,k}, X_{i,k_i})]$, *where* $B_0 = \{(\alpha, \beta) | \alpha = i_0, \beta \in R_0 \bigcap R(X_{i,k_i})\} \bigcap R(X_{i,k_i})$ *and* $B_i = \{(\alpha, \beta) | \alpha = i_0, \beta \in R(X^i_{k_i}) \bigcap R(X_{i,k_i})\} \bigcap R(X_{i,k_i})$.

Proof. It is obvious that $B_i \bigcap B_j = \phi$ for $i \neq j$. This guarantees that each request's access cost is only calculated one time. Now we prove the correctness of the calculation of $l(X_{i,k_i})$, i.e., the requests in B_i should be served by $X^i_{k_i}$. Suppose that there exists a request $r_{i',k'} \in b_i$ which is not served by $X^i_{k_i}$. Based on Property 1, we have all the requests in the region $B'_i = \{(\alpha, \beta) | i \leq \alpha \leq i_0, k_i \leq \beta \leq k_0$ will be not served by $X^i_{k_i}$. It is easy to see that $R(X^i_{k_i}) \bigcap B'_i \neq \phi$, i.e., there exist some requests in region $R(X^i_{k_i})$ that are not served by $X^i_{k_i}$. This

obviously contradicts the fact that all the requests in region $R(X^i_{k_i})$ are served by $X^i_{k_i}$. Hence, the theorem is proven.

For example, in Figure 3, if x_1 is removed, $R(x_1)$ can be divided into three regions (i.e., A, B, and C), which will be served by x_4, x_3, and the server, respectively. Thus, we have $l(x_1) = \sum_{r_{i,k} \in A} f_{i,k}[d(r_{i,k}, x_4) - d(r_{i,k}, x_1)] + \sum_{r_{i,k} \in B} f_{i,k}[d(r_{i,k}, x_3) - d(r_{i,k}, x_1)] + \sum_{r_{i,k} \in B} f_{i,k}[i \cdot L - d(r_{i,k}, x_1)]$.

In practice, the general case is that several versions of the same multimedia object are cached at node v_i at the same time (see Figure 4). In this case, calculating $l(x)$ should also consider the mutual effect of the least more detailed cached version on the removed version since the requests served by the removed version could be satisfied by this detailed version. For example, when calculating $l(x_2)$, $R(x_2)$ might be divided into four parts A, B, C, and D which will be served by x_4, x_5, x_3, and x_1, respectively.

Taking into consideration the caching dependence along the path, calculating $l(X_{i,k_i})$ becomes more complex and it is so obvious to obtain an optimal solution.

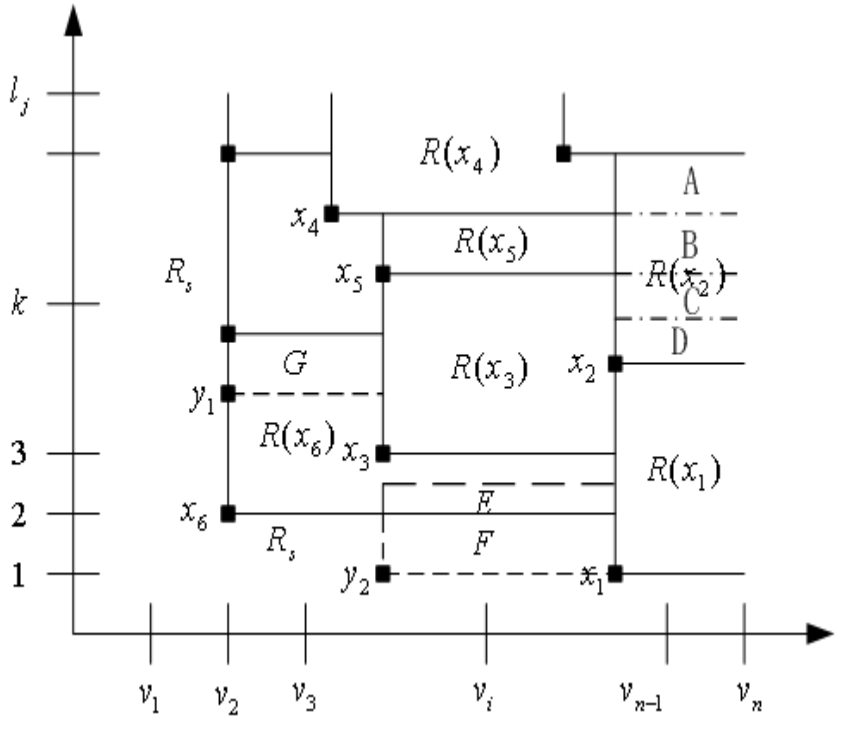

Fig. 4. Example for Calculating $l(x)$

Similarly, we can calculate the cost saving of caching O_{i_0} at node v_i. For example in Figure 4, if $i_0 = y_1$, then $R(x_6$ can be divided in to two parts: E and F; if $i_0 = y_2$, then $R(x_6$ can also be divided in to two parts. So we have $g(y_1) = \sum_{r_{i,k} \in G} f_{i,k}[d(r_{i,k}, y_1) - d(r_{i,k}, x_6)]$ and $g(y_2) = \sum_{r_{i,k} \in E} f_{i,k}[d(r_{i,k}, x_6) - d(r_{i,k}, y_1)] + \sum_{r_{i,k} \in F} f_{i,k}[i \cdot L - d(r_{i,k}, y_1)]$.

Now we begin to present an optimal solution for the problem as defined in Equation 2. In the following, we call the problem a k-optimization problem if we determine cache replacement candidates from nodes $\{v_1, v_2, \cdots, v_k\}$. Thus, the original problem (Equation (2)) is an n-optimization problem. Theorem 3 shows an important property that the optimal solution for the whole problem must contain optimal solutions for some subproblems.

Theorem 3. *Suppose that* $X = \{X_{i_1,k_{i_1}}, X_{i_2,k_{i_2}}, \cdots, X_{i_\alpha,k_{i_\alpha}}\}$ *is an optimal solution for the* α*-optimization problem and* $X' = \left\{X_{i'_1,k_{i'_1}}, X_{i'_2,k_{i'_2}}, \cdots, X_{i'_\beta,k_{i'_\beta}}\right\}$ *is an optimal solution for the* $k_{i_\alpha} - 1$*-optimization problem. Then* $X^* = \left\{X_{i'_1,k_{i'_1}}, X_{i'_2,k_{i'_2}}, \cdots, X_{i'_\beta,k_{i'_\beta}}, X_{i_\alpha,k_{i_\alpha}}\right\}$ *is also an optimal solution for the* α*-optimization problem.*

Proof. By definition, we first have $L(X^*) = l(X_{i'_1,k_{i'_1}}) + l(X_{i'_2,k_{i'_2}}) + \cdots + l(X_{i'_\beta,k_{i'_\beta}}) + l(X_{i_\alpha,k_{i_\alpha}}) = L(X') + l(X_{i_\alpha,k_{i_\alpha}}) \geq l(X_{i_1,k_{i_1}}) + l(X_{i_2,k_{i_2}}) + \cdots + l(X_{i_\beta,k_{i_\beta}}) + l(X_{i_\alpha,k_{i_\alpha}}) = L(X)$. On the other hand, since X is an optimal solution for the α-optimization problem, we have $L(X) \geq L(X^*)$. Therefore,we have $L(X) = L(X^*)$. Hence, the theorem is proven.

Based on Theorem 3, an optimal solution for the n-optimization can be obtained by checking all possible removed candidates from node v_1 to node v_n in order. Therefore, it is east to get that the time complexity of this solution is $O(n^2 + mn \log n)$ based on our previous result that the complexity for computing all $S(r_{i,k})$ is $O(mn \log n)$, where n is the number of nodes in the network and m is the number of versions of object j.

4 Conclusion

The transcoding proxy is attracting more and more attention since it plays an important role in the functionality of web caching. In this paper, we presented a coordinated cache replacement model in transcoding proxies where multimedia object placement and replacement policies are managed in a coordinated way. Our model is formulated as an optimization problem and the optimal solution is obtained using a low-cost dynamic programming-based solution.

References

1. C. Chang and M. Chen. *On Exploring Aggregate Effect for Efficient Cache Replacement in Transcoding Proxies*. IEEE Trans. on Parallel and Distributed Systems, Vol. 14, No. 6, pp. 611-624, June 2003.
2. A. Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz, and K. Worrell. *A Hierarchical Internet Object Cache*. Proc. of the USENIX Technical Conference, pp. 22-26, 1996.
3. 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 First Symp. Operating Systems Design and Implementations, pp. 267-280, 1994.
4. L. Fan, P. Cao, and J. Almeida. *Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol*. Proc. of ACM SIGCOMM Conference, pp. 254-265, 1998.
5. K. Li, H. Shen, F. Chin, and S. Zheng. *Optimal Methods for Coordinated En-Route Web Caching for Tree Networks*. ACM Trans. on Internet Technology (TOIT), Vol. 5, No. 2, May 2005.
6. M. R. Korupolu and M. Dahlin. *Coordinated Placement and Replacement for Large-Scale Distributed Caches*. IEEE Trans. on Knowledge and Data Engineering, Vol. 14, No. 6, pp. 1317-1329, 2002.
7. X. Tang and S. T. Chanson. *Coordinated En-Route Web Caching*. IEEE Trans. on Computers, Vol. 51, No. 6, pp. 595-607, June 2002.
8. X. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay. *Design Considerations for Distributed Caching on the Internet*. Proc. of the 19th Int'l Conference Distributed Computing Systems (ICDCS), pp. 273-284, 1999.
9. J. Xu, B. Li, and D. L. Li. *Placement Problems for Transparent Data Replication Proxy Services*. IEEE Journal on Selected Areas in Communications, Vol. 20, No. 7, pp. 1383-1398, 2002.