Placement Algorithm of Web Server Replicas

Seonho Kim[1], Miyoun Yoon[2], and Yongtae Shin[2]

[1] Dept. of Computer Science, Dongduk Women's University, Wolgok-Dong, Sungbuk-Ku, Seoul, Korea, 136-714.
shkim98@dongduk.ac.kr
[2]Dept. of Computer Science, Soongsil University, Sangdo-Dong, Dongjak-Ku, Seoul, Korea, 156-743.
myyoon@cherry.ssu.ac.kr
shin@computing.ssu.ac.kr

Abstract. Recently, contents distribution technologies have been used to cope with the explosive demand for Web services. In this paper, we addressed the issue of the optimal placement of replicas in the environment where Web contents are replicated. We placed replicas so that clients can have access to replicas with the proper delay and bandwidth. We attempted to solve the problem via dynamic programming considering cost of delay and traffic. We have come up with $O((n\text{-}d^h)\cdot|\ ch_{vi}\ |)$ time complexity that is less than $O(n^2)$. We defined the threshold and proved that our algorithm guarantees the reliable services.

1 Introduction

With the increasing popularity of Web, it is getting more and more difficult for a single web server to handle the explosive demand from its clients. Therefore, the CDN(Content Delivery Networks) technology is used by many web sites as it effectively reduces the client perceived latency and balance load. On the CDN, contents are replicated to distributed regional web servers and clients get contents from the appropriate replica server [1, 2]. The distributed system of replica web server(hereafter refered to as replica) is an effective solution to reduce a web server's load, Internet traffic, and user perceptive response time.

Under such circumstances, one of the most fundamental questions is where in the network replicas should be located to minimize the overall cost for the access to web servers. However, not many studies on the location strategy of replicas to further improve the CDN performance have been undertaken. Properly allocated and placed replica servers can have great impact on the performance of the system and reduce the cost and latency in accessing a web server.

Previous studies on the placement of caches and proxies have employed greedy algorithm to allocate a proper number of proxies and used NP-hard to place proxies. But these cannot adequately be applied to the replica server placement because of the very high computational complexity and the propensity of caches and proxies is different from that of the replica server [4, 10].

Therefore, in this paper, we suggest that the replica placement algorithm be used on the assumption that the underlying network topology is a tree and that the bottom-up

A. Laganà et al. (Eds.): ICCSA 2004, LNCS 3043, pp. 328–336, 2004.

dynamic programming approach be taken for the task. This approach can also be applied to the problem of request redirector placement of CDI(Content Delivery Internetworking)[3], the center of much of the recent attention.

In this research, the following questions are addressed:

(1) How many replicas are needed?

(2) Where in the Internet should the replica be located?

The goal here is to minimize the overall cost for clients in accessing a replica in the system.

2 Previous Work

Our problem is somewhat related to that of the proxy placement. Traditionally, most existing proxies have been placed in some obviously important locations, such as the router of a LAN, Gateway in the Internet or at some strategic points in the network of an institute or organization [5]. There is a trade-off relationship between minimizing a delay with replicas placed close to the client and maximizing the hit ratio with replicas placed at big traffic points. Most existing algorithms for the proxy placement approach it with the greedy method to find the optimal number of proxies and with NP-hard to determine the proper location of proxies.

Qiu et al. (2001) selected k from given n nodes, to be replicas to minimize the sum of the assignment costs. Each input node j was assigned to the selected replica closest to it. If j was assigned to a replica i, the cost $d_j c_{ij}$ would be produced where d_j represented the demand of the node j, and c_{ij} the distance between node i and j. They presented a number of algorithms for solving the minimum K-median problem and compared the time complexity of those algorithms as in Table 1. They evaluated the placement algorithms using several synthetic web traces and the AS[1]-level Internet topology for real network topologies. It was shown that the greedy algorithm that placed replicas based upon both a distance and request load performed best.

Table 1. Time Complexity

Tree-based [6]	Greedy	Random	Hot Spot
$O(N^3M^2)$	$O(N^2M)$	$O(NM)$	$N^2+min(NlogN+NM)$

Li et al. (1999) attempted to look into the appropriate placement of web proxies and recommended the application of the algorithm to minimize the overall latency of searching for a target web server based on resources and traffic patterns when the underlying topology is a tree. Given M proxies, they found the optimal placement of multiple web proxies M among N nodes using dynamic programming problem. In this study, they reached the optimal solution in $O(N^3M^2)$ times. It hinders its practical use in topologies with thousand of nodes because of its high computational complexity.

[1]Autonomous System

Jamin et al. (2000) and Radoslavov et al. (2001) attempted to solve the replica placement problem, minimizing the distance between the client and a replica through the K-median approach. They tried simulation by employing BGP routing table to modeling the real Internet topology. Radoslavov et al. (2001) proved that locating replica at a high degree node performed better than did the greedy under AS and router level Internet topology. However, if all the nodes have a similar low degree, it probably will not result in the same way.

Jamin et al. (2001) showed that increasing the number of replicas is effective in reducing the client download time under the AS-level topology. They treated an AS as a node, but it is not suitable due to the diversity of scales of ASs.

Krishnan et al. (2000) argued for the use of the cache-proxies placement algorithm under the linear and ring topologies. Each algorithm's time complexity was $O(NM^3)$, $O(NM^4)$.

3 Replica Server Placement Algorithm

3.1 Overview

To find the optimal number and placement of replicas we used the bottom-up dynamic programming method under a tree topology of which the root is the origin web server. We attempted to solve the problem by placing replicas where they were guaranteed not to exceed the given the threshold cost for traffic and delay, while dividing the tree to subtrees from the leaves to the root. Consequently, it was presumed that replicas would be properly placed in ASs under the router level topology, thus alleviating the network traffic and improving the response time of Web access.

Proposed network model was based on the following assumptions:

- The understructure is a tree that route a request from low to the upper level.
- All clients always select the replica with the lowest cost.
- We ignore the maximum number of clients which each replica can cover.

This model uses notations as shown in Table 2 and the network of nodes can be described as a graph *G*.

Table 2. Notations

Given a graph G=(V, E),
V : Set of Nodes , E : Set of Links for $E \subseteq V * V$
S : Origin Web Server, $S \in V$
N : Number of Nodes
R : Set of Replicas, $R = \{r_1, r_2, \ldots\ldots, r_m\}$
$G = T_1 \bigcup T_2 \bigcup \ldots\ldots \bigcup T_m$,
T_i*: Shortest Path Tree Rooted from Node i,* $1 \le i \le m$, $T_i \bigcap T_j = \Phi$ *for* $i \neq j$
$\lvert R \rvert = \lvert r_1 \rvert + \lvert r_2 \rvert + \ldots\ldots + \lvert r_m \rvert = k$, *k is a Number of Replicas in G*

3.2 Algorithm

Given a tree, T which consists of n nodes, random node v has delay time due to the distance between itself and child node, and has traffic generated from the link with child nodes. Therefore, for all $v \in V$, the cost of each node can denote the link delay(d) and the amount of traffic(w). The link delay is the time needed for a packet to traverse the link. It is the sum of transmission delay and propagation delay. We can compute the cost of random node v by multiplying traffic of v by delay from child nodes and then adding it to the child node's cost, as shown in the numerical expression (1).

$$Cost(v) = \sum_{i \in Ch_v} (\omega(i) * d(i,v) + Cost(i)) \quad (1)$$

Ch_v denotes child node of v, and if the $Cost(v)$ exceeds the threshold(τ), the node with the maximum cost among the child nodes of v becomes a replica. And then, the tree of which the selected replica is the root is omitted from the tree, T. If the $Cost(v)$ does not exceed the threshold(τ), a replica is unnecessary in that subtree. Next, we go to the upper level of the tree and compute the cost of upper nodes repeatedly until we meet the origin web server.

We define the threshold(τ) as below, thus we can guarantee that all nodes don't exceed the maximum bandwidth and the delay limit produced from exponential function in the dynamic network situation.

τ (Threshold) = $d_{avg}' * \varpi$

$\varpi = B_{max} * \alpha$, $0 < \alpha < 1$

B_{max} : the approved maximum bandwidth

α : current ration of approved bandwidth

d_{avg}': $\beta * d_{avg} + (1-\beta) * d_{new}$, $0 < \beta < 1$

d_{avg} : average value of previous delay with linked nodes

d_{new} : average value of current delay with linked nodes

The detailed algorithm is given in Fig 1.

τ : Threshold
w(i) : amount of traffic generated by node i
d(i, v): delay time of traversing link (i, v)
Ch_{vi} : set of child nodes of node vi
l : level of node vi in a tree
k : number of replicas

for all vi at level l -1
if l = 0 stop;

$$Cost(vi) = \sum_{vj \in Ch_{vi}} (\omega(vj) * d(vj, vi) + Cost(vj))$$

if $Cost(vi) > \tau$

$\overline{vi} = max[Cost(vj)]_{vj \in Chvi}$;

$r_k = \overline{vi}$;

$V = V - (\{r_k\} \bigcup Chr_k)$;

k++;

endif

l-- ;

Fig. 1. Replica placement algorithm

From the proposed replica placement algorithm (Fig. 1), we can get the number of replicas(k) and the location of the replica. $k=1$ means that there is no replica and just an origin web server exists, and $k=N$ means that all nodes are replicas.

Theorem 1 Given the delay, traffic and the threshold, the number of replicas(k) calculated via proposed algorithm is the minimum value for satisfying the threshold(τ).
Proof If we suppose that k is not the minimum value, then the number of replicas smaller than k can satisfy the services. To make $k-1$ the minimum value, one random replica r_i should be omitted from the replica set, R. Then, the cost of the parent of node i exceeds the threshold(τ), not satisfying the services. Therefore, k is the minimum number of replica.

Theorem 2 The cost for each node to have access to the closest replica r_k does not exceed the threshold(τ).
Proof If the cost exceeds τ, there should exist another replica before r_k. Then, the number of replicas is $k+1$, which is larger than the optimal value k. This appears to violate Theorem 1. Therefore, the cost of each replica does not exceed the threshold, which proves Theorem 2 correct.

4 Performance Analysis

To prove proposed algorithm's superiority, we prove that proposed algorithm is better than random placement which fails to take into account the cost for the access to replicas by a numerical analysis. And we compare proposed algorithm with previous ones in terms of the time complexity. We generate link delay according to the exponential distribution which has acquired an average delay from real experiments [11], and we obtain the amount of traffic from random probability.

4.1 Analysis by Numerical Example

Now, we evaluate the proposed algorithm in relation to the random placement through the numerical example given in Fig. 2 and Fig. 3. There are 17 nodes and the root is the origin web server. The amount of traffic and delay is given as ($w(j)$, $d(i, j)$) where node i is the parent of the node j. Link delay is generated by the exponential distribution which has average delay 9.02 (=1/λ) [11]. The equation for the generation of delay is $d = -\frac{\ln R}{\lambda}$, 0< R < 1. In this way, we get the threshold of 24.5 when α and β = 0.8, the average delay of 3.9 and the maximum traffic of 9.8.

First, to select replicas by the proposed algorithm, we compute each node's cost of the tree, T in Fig. 2 from the second lowest level to the highest except leaves of the tree until we reach the origin web server using the numerical expression (1).

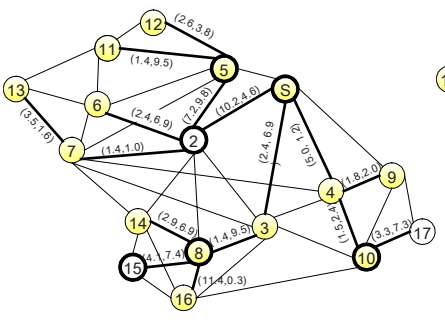

Fig. 2. Selected replicas by proposed algorithm

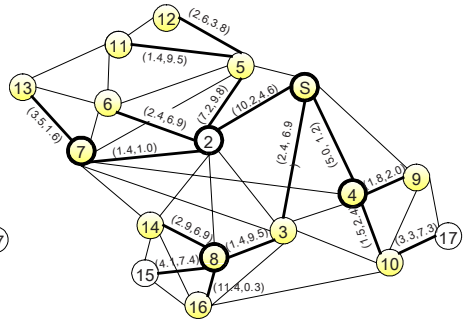

Fig. 3. Selected replicas by random table

Step 1: Compute cost of nodes 5, 6, 7, 8, 9, 10
Cost(5) = 1.4*9.5 + 2.6*3.8 = 23.2
Cost(6) = 0
Cost(7) = 3.5*1.6= 5.6
Cost(8) = 2.9*6.9 + 4.1*7.4 + 11.4*0.3 = 53.8
Cost(9) = 0
Cost(10) = 3.3*7.3 = 24.1

The cost of node 8 exceeds the threshold(τ). Therefore, node 15, which has the maximum link cost with node 8, becomes replica r_1, and node 15 is extracted from the tree, T.

Step 2: Compute the cost of node 2, 3, 4 which are at the next upper level.
Cost(2) = (7.2*9.8 + 1.4*9.5 +2.6*3.8) + (2.4*6.9) + (1.4*1.0 + 3.5*1.6) = 117.3
Cost(3) = (1.4*9.5) + (2.6*6.9 + 11.4*0.3) = 34.7
Cost(4) = (1.8*2.0) + (1.5*2.4 + 3.3*7.3) = 31.3

All nodes exceed the threshold(τ). Therefore, node 5, 8 and 10 which have the maximum cost among the child nodes of node 2, 3 and 4, become replicas and the subtree whose root is node 5, 8 and 10 are extracted from the tree, T.

Step 3: Compute the cost of the root which is at the next upper level of the tree.
Cost(S)= (10.2*4.6+2.4*6.9+1.4*1.0+3.5*1.6) + (2.4*6.9) + (5.0*1.2+1.8*2.0)
= 96.6

This node also exceeds the threshold(τ). Therefore, node 2, which has the maximum cost among the child nodes of the root, becomes a replica.

As a result, the number of replicas(k) is 5, and the locations of replicas are nodes 2, 5, 8, 10, 15. The cost of all replicas does not exceed the threshold. Then, all clients have access to stable services from a replica.

And next, we select, from the random table, nodes 2, 4, 7, 8 as replicas like Fig. 3 and compute their costs.

Cost(2)= (7.2*9.8 + 1.4*9.5 +2.6*3.8) + (2.4*6.9) + (1.4*1.0 + 3.5*1.6) = 117.3
Cost(4)= (1.8*2.0) + (1.5*2.4 + 3.3*7.3) = 31.3
Cost(7) = 3.5*1.6= 5.6
Cost(8) = 2.9*6.9 + 4.1*7.4 + 11.4*0.3 = 53.8

In this case, most of replicas exceed the threshold and the replica at node 7 has low efficiency because it is selected randomly that it does not take into account the cost for the access to replicas. Therefore, clients who access a replica cannot receive reliable services.

4.2 Algorithm Verification

Theorem 3 The proposed replica placement algorithm has $O((n\text{-}d^h)\cdot|\, ch_{vi}\,|\,)$ time complexity that is less than $O(n^2)$ such that $|\, ch_{vi}\,| = d*(d^{\,h-1} -1 / d-1) +1$, $d > 1$. Here, d stands for degree, h depth of tree, l level of *vi* node.

Proof We should compute the cost of child nodes of each node to get the cost of the node. It needs $O(\,|\, ch_{vi}\,|\,)$ times to check whether the cost of each node exceeds τ . And proposed algorithm needs $O((n\text{-}d^h)\cdot|\, ch_{vi}\,|\,)$ times because it should traverse to the root. Furthermore, $O((n\text{-}d^h)\cdot\ |\, ch_{vi}\,|\,) < O(n^2)$ is absolutely true because $(n\text{-}d^h) < n$ and $|\, ch_{vi}\,| < n$. Therefore, the time complexity of proposed algorithm is simpler than that of greedy and tree-based dynamic programming of [6].

Theorem 4 The cost utilization(ρ) of each replica, which is placed by proposed algorithm (Fig. 1.), satisfies that $0 < \rho \le \dfrac{\tau}{\sum_{i=1}^{m} Cost_i}$

where, we define the cost utilization(ρ) of each replica by

$$\rho = \frac{Cost_k}{\sum_{i=1}^{m} Cost_i} \qquad \text{for } \exists k < n, \forall i < n$$

Proof Suppose that $V = \{v_1, v_2, \ldots\ldots v_n\}$ is a set of nodes. Then by definition,

for $\exists v_k \in V$, $\rho = \dfrac{Cost_k}{\sum_{i=1}^{m} Cost_i}$ $(0 < m < n)$ can be satisfied.

By proposed algorithm, it can be stated that

$$Cost_k \leq \tau,\ Cost_k > 0,\ \tau > 0, \text{ and } Cost_i > 0.$$

Therefore, we arrive at the following result;

$$\rho = \frac{Cost_k}{\sum_{i=1}^{m} Cost_i} \leq \frac{\tau}{\sum_{i=1}^{m} Cost_i}$$

This shows that the cost utilization of each replica does not exceed a proper limit. This means that all clients could receive reliable services in terms of the cost of delay and traffic. In case of random placement, the cost utilization of each replica could fluctuate and the client-perceived quality of the service is unstable.

5 Conclusion and Future Research

In this study, we have dealt with the replica placement problem. We defined the threshold and cost with regards to the link delay and amount of traffic. And we solved the problem by dynamic programming method on the assumption that the underlying topology is a tree. We obtained $O((n\text{-}d^h)\cdot|ch_{vi}|)$ time complexity for the replica placement and proved that proposed algorithm guarantees reliable services.

But it should be noted that the Internet topology is not always a tree. Therefore, future studies must attempt simulations using the real Internet topology.

References

1. Rabinovich, M., Spatscheck, O.: Web Caching and Replication. Addison Wesley Professional (2002)
2. Peng G.: CDN: Content Distribution Network. In Stony Brook University Tech. Reports, TR-125, School of Computer Science Department, Stony Brook University (2002)
3. Biliris, A., Cranor, C., Douglis, F., Rabinovich, M., Sibal, S., Spatscheck, O., Sturm, W.: CDN Brokering. In proceedings of WCW (2001)
4. Qiu, L., Padmanabhan, V.N., Voelker, G.M.: On the Placement of Web Server Replicas. In proceedings of IEEE INFOCOM (2001)
5. Nabeshima, M. The Japan Cache Project: An Experiment on Domain Cache, In Proceedings of the Sixth International WWW Conference (1997)
6. Li, B., Golin, M.J., Italiano, G.F., Deng, X., Sohraby, K.: On the Optimal Placement of Web Proxies in the Internet. In proceedings of IEEE INFOCOM (1999)
7. Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: On the Placement of Internet Instrumentation. In proceedings of IEEE INFOCOM (2000)

8. Radoslavov, P., Govindan, R., Estrin, D.: Topology-Informed Internet Replica Placement. In proceedings of WCW (2001)
9. Jamin, S., Jin, C., Kurc, A., Raz, D., Shavitt, Y.: Constrained Mirror Placement on the Internet. In proceedings of IEEE INFOCOM (2001)
10. Krishnan, P., Raz, D., Shavitt, Y.: The Cache Location Problem. IEEE/ACM Transactions on Networking (2000) 8(2) 568-582
11. Park, JH.: An Improved Overlay Multicast Scheme for Enhancing Transport Efficiency of High Capacity Contents. Soongsil University, Seoul, Korea, Master's thesis (2003)