Surveying Formal and Practical Approaches for Optimal Placement of Replicas on the Web

TR020701

April 2002

Erbil Yilmaz
Department of Computer Science
The Florida State University
Tallahassee, FL 32306
yilmaz@cs.fsu.edu

Yanet Manzano
Department of Computer Science
The Florida State University
Tallahassee, FL 32306
manzano@cs.fsu.edu

Abstract

Replicating content has become one of the most employed techniques to solve the currently experienced slow download time and rapidly growing demand for web contents. However the effectiveness of the solution may dramatically decrease without optimization. In this paper we survey the lately proposed solutions to optimize the placement.

I. Introduction

In the 21st century, computers are the backbone of all system in our society. Today, digitalize information is the faster growing medium used to store data, and transfer data in the world. With the introduction of the World Wide Web to the scene this practices intensify to reach massive proportion. With the rapid growth of the web traffic, the load on popular individual web sites as well as overall network has been dramatically increased. One of the most commonly used methods to alleviate these problems is replicating content. Creating a replica, gives us the possibility to allocate capacity and reroute traffic to accommodate demand and performance requirements. Distributing the content decreases latency; improves user-perceived performance and reduces overall network traffic [6].

Two extensively employed methods for replicating content are mirror server and web caches. Although the former one seems only to improve the performance of popular web sites and the later one is used for overall improvement for web traffic, from the web performance point of view, both can be seen as tools to improve overall web traffic. We will discuss those two examples in more detail in section two.

Ideally, content replicas are placed where there is a large centrality of clients requested the content of the web server. However, in real world it becomes a problem to optimize the placement. With the optimal placement strategies for

content replicas, we may minimize the number of necessary mirrors and web caches without losing too much from the performance gain from the employment of these methods. Further even if the placement cost of content replicas were inexpensive, still we would face an optimization problem, since the decision for placement of replicas might be crucial.

In this paper we will discuss some methods used to improve web performance by doing optimal placement. We will explore some formal algorithms, and practical approaches to the general placing problem. Moreover, we drive some results from the performance comparison in case analysis.

In section two we will discuss some practical examples of the placement problem. In sections three we will then formally define the problem, and introduce some formal approaches in section four. In section five we will discusses the limitation of formal approaches, and introduce some practical approaches that attempt to improve performance while taking into considerations practical issues ignored by the formal methods.

II. Application Domains for Placement Methods

A. Web cache

Web cache sits between a server and clients watching request for content, while keeping a copy for itself. When a request for content is detected, there are two possible scenarios. In scenario one, (see figure 1) where the cache does not have a copy of the content in which case it sends the request over to the server and makes a copy once the server delivers the

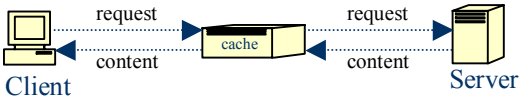

Figure 1: Requested content not found in cache storage

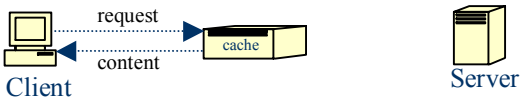

Figure 2: Requested content found in cache storage

content. In scenario two, (see figure 2) the cache has a copy of the content requested so it simply delivers the stored content.

B. Mirror Servers

Mirror servers, as suppose to web cache are selected and paid for by the website owner to store a copy of the whole, or most popular content on their website. The idea of mirror server has created a new type of website, popularly called content providers, whose sole purpose is to store the content of other websites.

Two of the most commonly used mirror server scenarios, are automatic redirecting, and gateway redirecting. In the case of automatic redirecting, (see figure 3) the client sends a request for content to the server, who passes the request to the mirror closet to the client to deliver the content. In the case of gateway redirecting the clients connects to a gateway website that contains a list of mirrors where the requested content can be requested from. It is up to the client then what mirror to choose.

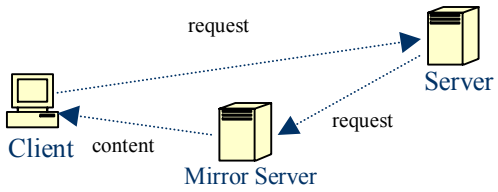

Figure 3: Automatic Redirecting

C. Web Cache vs. Mirror Servers

One of the main differences between cache and mirror servers is that client access to a mirror always finds the requested content. However, using mirror servers implies selecting the appropriate server to make the techniques effective. Unfortunately, although a number of server selection mechanism have being proposed, ad hoc continues to be the main selection methods currently being used [8].

III. Formulating the Optimal Placing Problem

Replicating content as we mentioned in previous sections is not only one of the most currently used solutions to improve performance on the web, but can also be considered as an effective solution provided that the distribution of the replicas can be optimized to achieve maximum performance.

Optimizing the placement of replicas will be the major consideration in the rest of this paper. Before the survey of some formal, and practical methods used in optimal placement, we will define the optimal placement problem itself, and establish the common grounds used by all the solutions surveyed in the paper.

Formal Model:

To formulating the problem as presented in [7], we consider the Internet topology as a graph $G = (V,E)$, where $V = v_i$ for $i = 1$ *to* n is the set of nodes, and E is the set of edges $d(e)$, the length of e reflects the delay caused by this edge, and $d(v_i, v_j)$ is the sum of the link distances along the route between nodes v_i and v_j. We also have a set of demands $F: V \times V \rightarrow N$, that defines the flow or amount of data (in bytes) requested by each client from each node, and a set of hit ratio $P: V \times V \rightarrow [0,1]$, that hit ratio of the flow specified in F.

To solve the optimal placement problem for k replicas, we need to find a set of nodes $K \subset V$ of size k that allows us to minimize the maximum distance between a node and its closet replica. This problem is known in literature as the K-center problem.

Although the K-center problem is NP-complete, we can relax the optimization condition to make the problem solvable in P-time, creating in the process a relatively optimal solution. In the next section, we will introduce some formal algorithms that can be used to solve the K-center problem in P-time.

IV. Formal Approaches

In this section, we will discuss some formal algorithm that attempt to solve the K-center problem. The algorithms are based on two common assumptions: one that the network topology is known, and two that a cost for each connection is assigned. For this section, we denote the set of the candidate hosts, replicas and clients as H, M, B

respectively and their corresponding sizes as $|H|$, $|M|$, and $|B|$.

A. Random Algorithm

Random algorithms are currently being used to place replicas on the Internet. The algorithm does not take into account client workload, instead it randomly places the replicas in G. Using this approach each candidate in H has a uniform probability of hosting a mirror.

According to [1] the ratio of expected maximum client-replica distance between optimal and random placement increases logarithmically. As the number of replicas grows, optimal placement increasingly outperforms the random placement in terms of expected client-replica distance. When clients are uniformly distributed, optimal placement can achieve good load balancing in directing clients to the closet mirrors.

B. Greedy Algorithm

The greedy algorithm approaches the problem by placing replicas on the tree iteratively in a greedy fashion without replacing already placed replicas. The algorithm has complexity $O(nk)$, where k is the number of replicas to be placed, and n is the maximum number of nodes in the tree. [7]

In the algorithm, given in Figure 3, when $l = 0$, the algorithm exhaustively checks each node in H to determine the node that best satisfies the optimization condition for given B. Generally, the algorithm assumes the clients direct their accesses to the nearest replica, and therefore the selection process is chosen in such a way as to yields the node with the lowest cost. As explained in [1], in general the algorithm allows l step(s) backtracking: checks all possible combination of removing l of the already placed replicas and replacing them with $l+1$.

C. K-min Algorithm

It is a variant of the K-center problem that tolerates the maximum distance between a node and its closest center up to twice the distance of the maximum node-closest center distance in the optimal solution and can be solved in O(N|E|) time. This variant is known as 2-approximate minimum K-center problem.

The algorithm for this problem is given in Figure 5, and presented in [6]. For a given graph $G = (V,E)$ such that its edges are in non-decreasing order by edge cost, i.e. $c(e_1) <= c(e_2) <= \ldots <= c(e_n)$, construct the square graph of G, where square of G, G_2 is a graph containing V and edge $e(u,v)$ for all u,v such that $u!=v$ and the there exists a path in G from u to v of at most two hops. Apparently, G_2 contains more edges than G. An

Algorithm for 2-approximate minimum K-center problem

1. Construct G_1^2, G_2^2, ..., G_m^2
2. Compute M_i for each G_i^2
3. Find smallest I such that $|M_i| <= K$, say j
4. M_j is the set of K centers

Figure 4: Algorithm *l*-Greedy

Algorithm *l*-Greedy

1. **if** ($|M| <= l$)
2. Choose among all sets M' with $|M'| = |M|$ the set M'' with minimal $O(M'', p)$
3. **return** set M''
4. **end**
5. Set M' to be an arbitrary set of size l
6. ***while*** ($|M'| < |M|$)
7. Among all sets X of l elements in M' and among all sets Y *of* $l+1$ elements in $V - M'+X$, choose the set X,Y with minimal $O(M'-X+Y,p)$
8. $M' = M'-X+Y$
9. **end**
10. **return** set M'

Figure 5: 2-approximate min K-center Algorithm

independent set of a graph $G=(V,E)$ is defined as a subset of nodes in graph G, $V'<= V$, such that for all u,v in V' the edge $e(u,v)$ is not in E. Further a maximal independent set M is defined as an independent set of V' such that all the nodes in V-V' are at most one hop away from the nodes in V'.

D. Tree Based Algorithms

In [9], author proposes a solution for the K-center problem by setting the graph G equal to T, where T represents a tree topology. The algorithm was originally designed for web proxies cache placement, but it can be expanded to web replica placement. As summarized in [6] the algorithm proposes to divide T into several small trees T_i , and shows that the best way of placing t>1 replicas in T is to place t_i replicas the best way in each small three T_i where $\Sigma_i\ t_i' = t$.

The algorithm works on the bases of two assumptions: one that the topology is a tree and two that clients request from the replica on the path toward the replica. One constrain is that clients cannot request content from a sibling replica. However according to [6] this two assumption can yield better placement choices.

V. Practical Approaches

Formal approaches although useful seems to ignore certain practical aspect that need to be consider for optimal placement in real networks. Formal approaches work under two main assumptions that seem to render them as inappropriate solutions for practical situations: one is that the topology of the network is known, and two that a cost for each is assigned.

In this section we introduce two practical approaches that attempt to eliminate the limitations of formal approaches, by not making the previously mentioned assumptions. In the first approach, IDMaps, we introduce an attempt to create a mechanism to

realistically determine the distance between any two hosts on the Internet. In the second approach, client clustering, the author relays on a more intuitive notion to determine optimal placement.

A. IDMaps

The IDMaps project proposes a general architecture for the global Internet host distance estimation service. The project attempts to provide a map the Internet that can be used to determine the distance between any two hosts [6].

In the placement problem one of the factors to consider is the distance between a client and its closest mirror. One of the assumptions that weighted heavily against the formal approaches is the assumption of the know Internet topology. With IDMap, we can realistically determine the network topology at any point in time, which allows us to optimize the placement of replicas.

The architecture consists of a network of instrumentation boxes, called Tracers. A tracer measures the distance among themselves and AP ("Address Prefixes") regions of the Internet. As explained in [6] assuming known topology of the Internet provides for a more optimal placement of the tracers, but it is not necessary. Once the Tracers are placed on the Internet, they start tracing to each other and to regions of the Internet. Clients of IDMaps, then collect the advertised traces, and use them to create distance maps.

B. Client Clustering

It has been discussed in the first section as a common sense approach to the placement problem to place the web replicas close to large concentration of clients interested in the content of those replicas. The idea behind this approach is moving the replicas closer to the cluster of users that request the content, will definitely decrease the portion of network load due to those users as well as the user-perceived latency. The concept of clusters is defined in [4] as the group of users that are close together topologically and likely to be under common administrative control.

Although client clustering seems easy under the assumption that the cluster of a user can be determined from the first three bytes of its IP addresses, this assumption fails under the validity tests that measures the performance of classifications. Indeed this approach may classify the users from different administrative entities or even from different geographical locations into the same cluster as explained in detail in [4].

As a solution to the clustering problem network-aware clustering has been proposed in [4]. The method makes use of the prefix and netmask information ejected from the Border Gateway Protocol routing and forwarding tables. Under the different validation tests for clustering the network-aware clustering approach gave outstanding performance. Also a self-correction and adaptation mechanism was proposed to further improve the network-aware clustering.

VI. Conclusion

We try to give a brief overview of the main aspects of the placement problem both from theoretical and practical point of view. Although the performance of the proposed methods

heavily depend on the simulation and testing environment setup, the methods produce very promising improvement over the current solutions. In most case studies, the performance analysis generally yields an improvement of performance when using the methods surveyed on this paper over random placement, which is what is currently being used.

One aspect to consider however, is that the effectiveness of the approach is also dependant on the constrains imposed in the result. For example, the effectiveness of IDMaps, depends on the metric used, and the level of accuracy needed from the collected traces. The effectiveness of the IDMap as a solution may be evaluated based on how accurate the estimated distances is compared to actual distances. However, this metric becomes less dominant if we are trying to solve the closest server selection problem, since the ordering of the distances and not their accuracy is what matters in that case.

VII. Bibliography:

[1] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt, "Constrained mirror placement on the internet," Proc. of IEEE INFOCOM,Mar. 2001.

[2] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," Proc. of IEEE INFOCOM, Mar. 2001.

[3] H. Yu, L. Breslau, and S. Shenker, "A scalable web cache consistency architecture," Proc. of ACM SIGCOMM, Sep. 1999.

[4] B. Krishnamurthy, and J. Wang, "On network-aware clustering of web clients," Proc. of ACM SIGCOMM, Aug. 2000.

[5] V. Cardellini, M. Colajanni, and P. S. Yu, "Dynamic load balancing on web server system," IEEE Inernet Computing, pp.28-39, May-June 1999.

[6]S. Jamin, C. Jin, D. Raz, Y. Shavitt, and L. Zhang, "On the placement of internet instrumentation," Proc. of IEEE INFOCOM, Mar. 2000.

[7] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," ACM/IEEE Transactions on Networking, vol. 8, no. 5, Oct. 2000.

[8] A. Mayers, P. Dinda, and H. Zhang, "Performance characteristic of mirror services on the internet," Proc. of IEEE INFOCOM, Mar. 1999.

[9] B. Li, M. J. Golin, G. F. Ialiano, and X. Deng, "On the optimal placement of web proxies in the internet," Proc. of IEEE INFOCOM, Mar. 1999.

[10] V. N. Padmanabhan, and L. Qiu, "The content and access dynamics of a busy web site: findings and implications," Proc. of ACM SIGCOMM, Sep. 2000.