On Spawning Web Server and Object Replicas

Ilker Basaran, Shalendra Chhabra, Michalis Faloutsos

Computer Science and Engineering
University of California, Riverside
California, USA
{ibasaran,schhabra,michalis}@cs.ucr.edu

Abstract. In this paper we present an overview of the current research initiatives in replica placement strategies for content replication over the Internet. We describe web server replica placement problem in the presence of both single origin server and multiple origin servers. We explore the work on replica placement which explicitly takes network topology both at Autonomous System's level and Router level in account. We mention the recent work on object replication at the *per object granularity level*. We conclude with some interesting future directions for content replication in different kind of networks like push based networks, peer to peer networks with cooperation unlike traditional pull based networks.

Keywords: Content Distribution Networks (CDN), Autonomous System (AS), Border Gateway Protocol (BGP), NP Completeness, Replica Placement, Pull Based Networks, Push Based Networks, Peer to Peer Networks

1 Introduction

With the rapid growth of the World Wide Web, popular web sites have started experiencing heavy loads of traffic. In order to provide better service to their clients, these popular web based destinations are outsourcing their content for replication on to companies like Akamai [1], Digital Island [4] etc. Replication as served by these corporations improves Quality of Service, reliability, availability and performance.
In this paper we describe the state of the art of content replication over Internet. In Section 2 we describe the notion of replica placement, the web server replica placement problem and its computational complexity. In Section 3 we describe the web server replica placement algorithms. In Section 4 we describe topology informed replica placement algorithms. In Section 5 we cover the object replication problem in the presence of multiple origin servers. We conclude with some interesting directions for future research.

2 Replica Placement

We first explain the notion of replica placement. In a very common scenario, a popular web site might aim to improve its performance by serving its clients

better (for ex: by reducing its clients perceived latency). It can do so by duplicating its content on servers closer to the clients. This is an example of replica placement.

2.1 Replica Placement Problem

The Replica Placement Problem is modeled as choosing M replicas (or hosting servers) among N potential candidates *(N>M)* such that some objective cost function (can be clients' perceived latency or total bandwidth consumption or update cost) can be minimized under a given set of network conditions. All research initiatives on web replica placement problem described in this Section have assumed the presence of a *single origin server*.
The Replica Placement Problem bears close resemblance with some well known graph theoretic problems given the network topology and the user demands. We now describe these variants:

1. *Facility Location*
   The facility location problem is as follows. Given a set of locations i at which facilities may be built, building a facility at location i incurs a cost of f_i. Each client j must be assigned to one facility, incurring a cost of $d_j c_{ij}$ where d_j denotes the demand of the node j, and c_{ij} denotes the distance between i and j. The objective is to find a solution (both in the number of facilities and locations of the facilities) of the minimum total cost. This problem is `NP Hard`. The best approximation algorithm for this problem was developed by [2], who gave a 1.728-approximation algorithm.
2. *K-Median*
   The minimum K-Median problem is as follows. Given n points, we must select K of these as centers and assign each input point j to the selected center that is closest to it. For a location j assigned to a center i, the cost incurred is $d_j c_{ij}$. The objective is to choose K centers such that the sum of the assignment cost of input points to each of these centers is minimized. This differs from the K-Median problem in the sense that there is no associated cost with opening the centers.
3. *Bin Packing*
   Given N objects, of different sizes, the Bin Packing problem deals with dividing them to the minimum number of disjoint sets such as the total storage at each set does not exceed a threshold S. This is commonly used to simulate load balancing problems. [13] have discussed the problem of distributing documents in a cluster of web servers in order to balance their load. This paper also proposes a binning algorithm for the initial distribution and network flow formulations.

2.2 The Computational Complexity of Replica Placement Problem is NP Complete

The Computational Complexity of Replica Placement Problem is NP Complete. Any problem X is NP Complete if X is in the class NP and X can be transformed to NP Problem in polynomial time. [3] have shown that Replica Placement Problem is NP Complete by transforming it to *Exact Cover by 3-Sets (X3C)* Problem [5].

3 Server Replica Placement Algorithms

We now describe some algorithms for solving the Replica Placement Problem modeled after K-Median Problem as mentioned in [14].

3.1 Random

The Random Placement algorithm does not take into account the client workload, instead it randomly chooses M replicas among N potential sites from a uniform distribution. Using this approach each node has a uniform probability of hosting a replica. As the number of replicas grow, other placement algorithms increasingly outperform the random placement in terms of expected client replica latency. [8] have shown that the ratio of expected maximum client-replica latency between optimal and random placement increases logarithmically.

3.2 Greedy

Greedy algorithm works as follows. Given M replicas among N potential sites, one replica is placed at a time. Initially N potential sites are evaluated individually to determine each one's suitability for hosting a replica. The cost computation is done by assuming that accesses from all clients converge to that site. The site with the lowest cost is chosen. In the second run, a second replica site is searched which together with the site already selected, yields the lowest cost. Also the algorithm assumes that clients direct their requests to the nearest replica. This is iterated until M replicas are chosen.

3.3 Hot Spot

The Hot Spot algorithm works by placing replicas near the clients generating the greatest request load. After sorting N potential sites according to generated traffic within their proximity, this places replicas at the top M sites that generate the largest amount of requests. The definition of proximity in [14] is the circle centered at A with some radius r.

3.4 Tree Based

In [10], authors propose a solution for the K-Median problem by setting the graph G equal to a tree topology T. The algorithm was originally designed for web proxies cache placement, but it can be expanded for web server replica placement. They divide the tree T into several small trees T_i, and show that the best way of placing $t > 1$ proxies in the tree T is to place $t'i$ proxies the best way in each small tree T_i. The two main assumptions within the algorithm are that the topology is a tree and that the clients request only from the replica on the path towards the replica and not from a sibling proxy. Interestingly, in [10], it is claimed that these assumptions help the algorithm to find better placement choices. They used dynamic programming to solve the problem of placement.

4 Topology Aware Replica Placement

We now describe some recent work on replica placement which explicitly takes topology information - both at the AS Level and Router Level in account.

4.1 Exploiting Topology Information at Autonomous Systems' Level for Replica Placement

[8] use the AS level topology knowledge (both real world and randomly generated) for the replica placement. In their study they select nodes as replicas in decreasing order of their node degree [1], where each node represents a single AS, and a node link corresponds to AS-level BGP peering.

1. *Model*
   In their model [8] have a fixed number of candidate machines where mirrors can be placed. They call this as `Constrained Mirror Placement (CMP) Problem`.
   In `CMP`, Internet is modeled as a graph, $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ the set of links. $H \subseteq V$ is defined as the set of candidate hosts where mirrors can be placed, $M \subseteq H$ the set of mirrors of a particular server S, and $B \subseteq V$ the set of $S's$ clients. The objective of the `CMP` problem is to place the set of mirrors on the set of the candidate hosts such that an optimization condition $O(M, P)$ is satisfied for the client set. The satisfiability of the optimization condition depends upon the sizes and topological placements of the candidate hosts and clients. The sizes of the candidate host, mirror, and client sets are denoted as $|H|, |M|$ and $|B|$ and their topological placements as $P(H), P(M)$ and $P(B)$ respectively. The notations H, M and B are used to denote a specific size and placement of the sets. Only cases with $|M| \leq |H|$ are considered. The model investigates the effect of changing $|M|$ and $P(M)$ while holding $|H|$ constant with $H \cap B = \Phi$ and $H \cup B \leq V$. The experiments have been carried with

[1] A node degree represents the number of links connecting a node to its neighbor

uniformly distributed $P(H)$ and by placing candidate hosts on nodes with the highest outdegree first. Also, both uniformly distributed and trace based $P(B)$ has been tried.

Optimization Condition $O(M, p)$
The optimization condition is modeled after the following two goals of mirror placement:
(a) Reducing round-trip time between a client and the closest mirror
(b) Alleviating load at the mirrors
In order to direct clients to the closest mirror the shortest path from the client to all mirrors using Dijkstra's algorithm are computed when the network topology is known. When the network topology is unknown, client re-direction is done randomly.

2. *Mirror Placement Algorithm*
The following placement algorithms and *Transit Node* heuristics have been tried in this model:
   (a) Min K -center Algorithm (a variant of K-Median Problem as described in Section 2)
   (b) Greedy Algorithm (As described in Section 3.2)
   (c) Random Algorithm (As described in Section 3.1)
   (d) Transit Node Heuristics (Mirrors are chosen out of candidate hosts in descending order of outdegree)
3. *Experimental Setup and Simulation*
The random topologies in this paper were generated using `Inet Topology Generator` [7]. Several random topologies with $3,037$ [2] nodes each are generated. Each generated network is a connected graph on the plane, with nodes representing an AS; a link between nodes representing AS Connectivity and its Euclidean distance denotes the latency between the two connected nodes. The `CMP` was also evaluated with trace-based experiments on the Internet. They used Bell Labs web server in this case.
4. *Results*
Their results suggest that the AS-level node degree based replica placement can perform almost as well as the greedy placement. They also report that, regardless of the placement method, increasing the number of replicas is effective in reducing client download time only for a very small number of replicas. (i.e. their result demonstrate the `law of diminishing returns` [12])

4.2 Exploiting Topology Information at the Router Level for Replica Placement

[15] have extended the work of [8] and evaluated node degree based replica placement using the router level topology. They retrieve the router level topology from [6]. They generate router-level paths using approximate models of inter-AS routing policy [16] instead of shortest-path routing.

[2] $3,037$ was the size of Internet in November 1997

1. *Model*
   In their model [15], each client selects the replica closest in the number of hops to it. Also, their is no limit on the number of clients that can be assigned to a particular replica. The following replica placement methods have been considered in this model:
   (a) *Greedy Placement*
   This is the same as [8] and [14] and has been described in Section 3.2
   (b) *Max-router fanout placement*
   Given a network topology and the fanout(degree) of each node, replicas are chosen in decreasing order of their degree until all replicas have been chosen
   (c) *Max-AS/max-router fanout placement*
   If M is the number of replicas to be chosen, first M replicas that have the largest fanout (on the AS-level topology) are selected. Within each selected AS, the router with the largest router-level fanout is chosen
   (d) *Max-AS/min-router fanout placement*
   This method is similar to the above but instead of choosing the router with the largest fanout within each chosen AS, the router with the smallest fanout is selected. This is done to evaluate the sensitivity of network performance to replica placement within an AS
   (e) *Random Placement*
   The replicas are chosen at random with uniform probability among all nodes in the topology

   The metrics considered in this model are *average client latency* and *overall network overhead*. The latency between two nodes is proportional to the number of link-hops between them. This is similar to assumption made in [14]. The *average client latency* is defined as

$$AveClientLatency = \frac{\sum_{clients(c)} Dist(c, Replica(c))}{NumberOfClients} \tag{1}$$

   where $Replica(c)$ is the replica node for client c, and $Dist(c, Replica(c))$ is the distance between them in number of hops.
   The *overall network overhead* for all clients is computed using the following formula:

$$NetworkOverhead = \sum_{clients(c)} Dist(c, Replica(c)) \tag{2}$$

   Thus,

$$AveClientLatency = \frac{NetworkOverhead}{NumberOfClients} \tag{3}$$

   The *Efficiency Ratio* of a Method is defined as (using greedy placement method as a base for comparison):

$$EffRatio(Method) = \frac{NetworkOverhead(Method)}{NetworkOverhead(Greedy)} \tag{4}$$

2. *Experimental Setup and Simulation*
   A real-world router level topology collected by sending a large number of `traceroute` requests over the Internet was used [17]. The resulting topology had 102639 nodes and 142303 links. The topology of *Internet Core* was obtained by removing all the nodes with a fanout of 1. The number of clients were fixed and number of replicas were varied and vice versa. The access logs of a busy web server were used to create the set of web-derived clients. Random graph and power law graph topologies were also used in this model.
3. *Results*
   Their results indicate that replica placement based on underlying topology at the router level is almost as good as the greedy placement (within a factor of 1.1-1.2). Their results are independent of the client locations unlike greedy algorithm based replica placement and hold true for random graphs, generated and real-world AS level topologies but are not valid for overlay topologies such as Mbone [11]. Another observation is that fanout-based placement methods seem to perform well on power-law and random graphs but may not be a good solution for graphs with nodes having relatively low fanout.

5 Object Replica Placement Algorithms

The Replica Placement Algorithms in Sections 3 and 4 focus on the problem of placing the replica servers for a *single origin server*. [9] have studied replication problem in a wider context of duplicating content from *several origin servers*. They also make replication decisions on a *per-object granularity level*. They also describe peer-to-peer content distribution and develop a model for studying the benefits of cooperations between peers.
In their model each AS is a node with finite storage capacity for replicating objects. The optimization problem is to replicate objects so that when the clients fetch objects from the nearest CDN server with the requested object, the average number of ASs traversed is minimized. This problem is `NP Complete`. We now describe their common object replica placement heuristics.

1. *Random*
   Objects are assigned to storage nodes randomly provided the storage constraint allows. The object and the node is picked with uniform probability, and the object is stored in that node. If the selected node already has that object, then a new object and a new node is picked. As a result, an object can be assigned to several nodes, but a node will have at maximum one copy of an object.
2. *Popularity*
   Each node stores the most popular (requested) objects among its clients. The objects are sorted in decreasing order of popularity *(popularity index)* and nodes store objects up to their capacity.

3. *Greedy-Single*
   Each node i calculates a contribution $C_{ij} = p_j d_{ij}(xo)$ for each object j. Here p_j is request rate of an object j and d_{ij} is the shortest distance of node i to object j. The node then sorts the objects in decreasing order of C_{ij} and stores objects upto their storage constraints. The request rate is obtained as in the Popularity heuristic, but the CDN also needs information about the network topology in order to estimate the $d'_{ij}s$. Cost function is calculated only once. This implies that every node stores objects independently of all the other nodes and no cooperation between nodes is required.
4. *Greedy-Global*
   The cost function $C_{ij} = \lambda_i p_j d_{ij}(xo)$ is calculated where p_j is request rate of an object j, d_{ij} is the shortest distance of node i to object j and λ_i is client request rate. The CDN then picks the node-object-pair which has the highest C_{ij} and stores that object in that node. This yields a new placement, say x_1. The CDN then re-calculates the costs C_{ij} under the new placement and picks the node-object-pair with the highest cost. This object is stored in this node and a new placement x_2 is obtained. This is iterated until all the storage nodes have been filled.
   The *Greedy-Global* heuristic outperforms other heuristics.

6 Conclusion and Future Work

We have described the state of the art of web server replica placement problem both in the presence of single origin server and multiple origin servers. We have described in detail recent work on replica placement which explicitly takes network topology information in account. We have also explored recent work on object replication at the *per object granularity level*.
There is a lot of rich potential for research in the area of replica placement in push based networks, peer to peer cooperative networks and for *replica replacement* in *dynamic networks* unlike traditional pull based networks. None of the mentioned papers have formalized all the constraints that a given network (or a network in *real time*) might experience. Studying replica placement problem in conjunction with network sensitivity in the the aforementioned networks is an interesting direction for future research.

References

[1] Akamai. `http://www.akamai.com`.

[2] M. Charikar and S. Guha. Improved Combinatorial Algorithms for the Facility Location and k-Median Problems. In *IEEE Symposium on Foundations of Computer Science*, pages 378–388, 1999.

[3] S. A. Cook, J. K. Pachl, and I. S. Pressman. The Optimal Location of Replicas in a Network using a READ-ONE-WRITE-ALL Policy. *Distrib. Comput.*, 15(1):57–66, 2002.

[4] Digitalisland. `http://www.digitalisland.com`.

[5] M. Garey and D. Johnson. *Computers and Intractability.* W. H. Freeman and Company, 1979.

[6] R. Govindan and H. Tangmunarunkit. Heuristics for Internet Map Discovery. In *IEEE INFOCOM 2000*, pages 1371–1380, Tel Aviv, Israel, March 2000. IEEE.

[7] Inet Topology Generator. `http://topology.eecs.umich.edu/inet/`.

[8] S. Jamin, C. Jin, A. R. Kurc, D. Raz, and Y. Shavitt. Constrained Mirror Placement on the Internet. In *INFOCOM*, pages 31–40, 2001.

[9] J. Kangasharju, J. Roberts, and K. Ross. Object Replication Strategies in Content Distribution Networks. In *J. Kangasharju, J. Roberts, and K. W. Ross. Object Replication Strategies in Content Distribution Networks. In Proceedings of WCW'01: Web Caching and Content Distribution Workshop, Boston, MA, June 2001.*, 2001.

[10] B. Li, X. Deng, M. J. Golin, and K. Sohraby. On the Optimal Placement of Web Proxies in the Internet: The Linear Topology. In *HPN '98: Proceedings of the IFIP TC-6 Eigth International Conference on High Performance Networking*, pages 485–495. Kluwer, B.V., 1998.

[11] M. R. Macedonia and D. P. Brutzman. MBone Provides Audio and Video Across the Internet. *IEEE Computer*, 27(4):30–36, 1994.

[12] T. Malthus. The Law of Diminishing (Marginal) Returns. In *Essay on the Principle of Population*, 1798.

[13] B. Narendran, S. Rangarajan, and S. Yajnik. Data Distribution Algorithms For Load Balanced Fault-Tolerant Web Access. In *SRDS '97: Proceedings of the 16th Symposium on Reliable Distributed Systems (SRDS '97)*, page 97. IEEE Computer Society, 1997.

[14] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. On the Placement of Web Server Replicas. In *INFOCOM*, pages 1587–1596, 2001.

[15] P. Radoslavov, R. Govindan, and D. Estrin. Topology-Informed Internet Replica Placement. In *Proceedings of WCW'01: Web Caching and Content Distribution Workshop, Boston, MA*, June 2001.

[16] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The Impact of Routing Policy on Internet Paths. In *INFOCOM*, pages 736–742, 2001.

[17] USC/ISI. The Scan Project. `http://www.isi.edu/scan/`.