High-Density Model of Content Distribution Network *

Craig Cameron Steven H. Low David Wei
Departments of CS and EE
California Institute of Technology
{cwcam,slow,weixl}@caltech.edu

December 14, 2001

Abstract

It is well known that optimal server placement is NP-hard. We present an approximate model of content distribution network for the case when both clients and servers are dense, and propose a simple server allocation and placement algorithm based on high-rate quantization theory. The key idea is to regard the location of a request as a random variable with probability density that is proportional to the demand at that location, and the problem of server placement as source coding, i.e., to optimally map a source value (request location) to a codeword (server location) to minimize distortion (network cost). This view leads to a joint server allocation and placement algorithm that has a time-complexity that is linear in the number of users.

1 Introduction

Content distribution network (CDN) reduces propagation delay, relieves server load, balances network traffic, improves service reliability, and disperses flash crowds. Content from a provider is distributed to multiple servers in the network, and a user request is served by a 'nearest' server. Here, proximity may refer to geographical distance, hop count, network congestion, server load or a combination. A central issue in the design and optimization of CDN is how to allocate and place servers in the network.

*This research is supported by the Caltech Lee Center for Advanced Networking and Yuen Research Fund.

The problem of optimal server placement is to decide how many servers to employ in a content distribution network and where to locate them. This is known as the K-median problem in graph theory: given a graph with N nodes, each node i with a request rate $r(i)$, pick $K(< N)$ nodes as servers and assign each node to one of these servers so that the total weighted distance between all nodes i and their servers, weighted by $r(i)$, is minimized. This problem is shown in [6, 8] to be *NP*-hard for general graphs. Subsequent efforts have been to find polynomial algorithms to solve special cases and to find approximation algorithms to solve the general case. These algorithms, however, are not applicable to large scale self-organizing CDN we envision, for two reasons. First, the best approximation algorithms are based on primal-dual schema and Lagrangian relaxation whose running time is not only large ($N^2 \log N$ or N^3, where N can be on the order of 10^8), more importantly, it is centralized and requires detailed global information throughout the execution of the algorithm. Hence it is neither scalable nor adaptive. Second, these algorithms are concerned with the placement of K servers to serve a single website (content provider). In CDN, websites have greatly varying popularity [2] and it is crucial to exploit this diversity in server allocation. Our ultimate goal is to develop simple distributed algorithms that can be used to self-organize CDN on a large scale dynamically based on current network traffic and user demands.

In this paper, we take a different approach, focusing on the case where both user and server densities

are high. In this regime, server placement can be regarded as a high-rate vector quantization problem. The key idea is to regard the location of a request as a random variable with a probability density that is proportional to the demand at that location, and the problem of server placement as source coding, i.e., to optimally map a source value (request location) to a codeword (server location) to minimize distortion (network cost). This view has led to a simple *joint* server allocation and placement algorithm with time complexity linear in NM where N is the number of users (e.g., client side proxies) and M is the number of content providers; in particular, it is *linear* in N. Preliminary simulation results suggest that it has a good performance-complexity tradeoff.

2 High density model

The proposal of [1] suggests the integration of storage with network where every router is potentially a small cache. Network nodes such as routers are in a best position to monitor traffic and self-organize as a CDN. The results of [10, 3] suggest that a relatively small client population is sufficient to attain a high hit rate. These motivate the study of large scale self-organizing CDN where there is a large number of servers, each serving a small population, that adapt their configuration to changes in user requests, network congestion, and server load. The main savings come from reduced network traffic and propagation delay, and the central issue is the optimal allocation and placement of these servers. This problem can be viewed as a high-rate vector quantization with dimension two [4]. In this section, we explain how this view leads to a simple server allocation and placement algorithm, presented in the next section. Finally we compare the performance of this algorithm with the best published approximation algorithm for the K-median problem.

We start with the case of a single website, and extend to the case of multiple websites. A 'website' in our model may represent a content provider, an entire website, a collection of files or applications, or a single file or application. A 'node' may represent an end user of the website, or more likely, a client-side proxy that serves a family of end users in the same local area network or same organization. By placing a server 'at a node', we mean placing a server 'near' the end user or client-side proxy represented by the node, e.g., on the same subnet.

2.1 Single website

Consider a set V of points (called *nodes*) in the $\Re^2$ plane indexed by $i = 1, 2, \ldots, I$. Let $z_i \in \Re^2$ be the coordinate of node i, and $Z = \{z_i \mid i \in V\}$ be the set of coordinates of all nodes; we will refer to a node both by i and by its coordinate z_i. Node i accesses the (single) website at a rate of $r(i)$ requests per minute. We are to place $K \geq 1$ servers at locations $s = (s_1, \ldots, s_K)$, where $s_k \in Z$, and the goal is to choose these locations s so as to minimize the total network cost of serving the requests, defined as follows.

Let $d(i, j)$ be the 'distance' of serving a request of node i by a server located at node j. Given server locations s, the distance measure $d(i, j)$ partitions V into what are called Voronoi cells $V_k \subseteq V$ defined by:

$$V_k \quad = \quad \{j \in V \mid d(j, s_k) \leq d(j, s_l), \ \ \forall l\}$$

such that $\cup_{k=1}^{K} V_k = V$. If $d(j, s_k) = d(j, s_l)$, ties are broken arbitrarily so that V_k are disjoint. Hence members of Voronoi cell V_k are *nearest neighbors* of server s_k.

The cost of serving a Voronoi cell V_k is $\sum_{j \in V_k} r(j) d(j, s_k)$. When there are K servers, the *network cost* is

$$c(K) \quad = \quad \min_{s} \sum_{k=1}^{K} \sum_{j \in V_k} r(j) d(j, s_k) \qquad (1)$$

where the minimization is over all server locations $s = (s_1, \ldots, s_K)$ in V.[1] This cost, in units of *rate-distance*, is a measure of minimum network capacity required, the minimum amount of network traffic, or total delayed weighted by demand, when there are K servers.

[1] This formulation assumes that the location of the origin server (content provider) can also be optimized. This is optimistic but the effect of this approximation is insignificant for a large network.

2.2 High-density approximation

We now approximate (lower bound) the network cost for the case when both nodes and servers are dense. Let *every* point $z = (z_1, z_2) \in \Re^2$ be a node. The request rate of node z is $r(z)$. Interpret the normalized request rate

$$f(z) \quad = \quad \rho^{-1} r(z) \text{ where } \rho := \int r(z)\, dz \quad (2)$$

as the *spatial density* of requests. The idea is to regard the location Z of a request as a random variable with probability density f, and the problem of server placement as a source coding, i.e., to optimally map a source value (request location) to a codeword (server location s) to minimize distortion (network cost). We now apply the techniques of high-rate vector quantization [4, Chapter 5] to derive a server allocation and placement algorithm.

The network cost is defined in an analogous way to (1) as, for a single website and K servers,

$$c(K, s) \quad := \quad \rho \sum_{k=1}^{K} \int_{V_k(s)} f(z) d(z, s_k)\, dz$$

Here, $s = (s_1, \ldots, s_K)$ are the locations of the K servers, $V_k(s) = \{z | d(z, s_k) \leq d(z, s_l),\ \forall l\}$ is the Voronoi cell containing server location s_k. We assume that K is large so that $V_k(s)$ is small, and that $f(z)$ is smooth so that $f(z) \simeq f(s_k)$ over $V_k(s)$. We further assume that the distance function is the Euclidean distance, $d(z, s) = ||z - s||$. Then

$$c(K, s) \quad \simeq \quad \rho \sum_{k=1}^{K} f(s_k) \int_{V_k(s)} ||z - s_k||\, dz$$

Next we approximate the region $V_k(s)$ by a circular disk with the same area centered at s_k described by:

$$\{z \mid ||z - s_k|| \leq a_k\} \qquad \text{with } a_k := \sqrt{\frac{|V_k(s)|}{\pi}}$$

where $|A|$ denotes the area of set A. Then $c(K, s)$ is roughly: [2]

$$\rho \sum_{k=1}^{K} f(s_k) \int_{\{z: ||z - s_k|| \leq a_k\}} ||z - s_k||\, dz$$

By changing the variable of integration, we get

$$c(K, s) \quad \simeq \quad \frac{2\rho}{3\sqrt{\pi}} \sum_{k=1}^{K} f(s_k) |V_k(s)|^{3/2} \quad (3)$$

We specify server location in this continuum model by *server density* $\lambda(z)$, with the interpretation that the *fraction* of servers in an infinitesimally small area dz around z is $\lambda(z)dz$. Hence the *number* of servers in any region A is $K \cdot \int_A \lambda(z)dz$. Note that $\int \lambda(z)dz = 1$ so λ can also be regarded as the probability density of server location. We approximate $\lambda(z)$ by $\lambda(s_k)$ over the small Voronoi cells $V_k(s)$. Hence, since there is exactly one server in each Voronoi cell, we have

$$1 \quad = \quad K \cdot \int_{V_k(s)} \lambda(z)\, dz \quad \simeq \quad K \cdot \lambda(s_k)\, |V_k(s)|$$

or

$$\sqrt{|V_k(s)|} \quad \simeq \quad \frac{1}{\sqrt{\lambda(s_k) K}}$$

Substituting into (3), and when K is large so that $V_k(s)$ is small, we have

$$\begin{aligned} c(K, s) \quad &\simeq \quad \frac{2\rho}{3\sqrt{\pi K}} \sum_{k=1}^{K} \frac{f(s_k)}{\sqrt{\lambda(s_k)}}\, |V_k(s)| \\ &\simeq \quad \frac{2\rho}{3\sqrt{\pi K}} \int \frac{f(z)}{\sqrt{\lambda(z)}}\, dz \end{aligned}$$

Using Hölder's inequality, it can be shown that the right-hand side is lower bounded by [4, Chapter 5]

$$c^*(K) \quad = \quad \frac{2}{3\sqrt{\pi}} \left(\int f(z)^{2/3} dz \right)^{3/2} \cdot \frac{\rho}{\sqrt{K}} \quad (4)$$

and this lower bound is achieved with the server density

$$\lambda^*(z) \quad = \quad a\, f(z)^{2/3} \quad (5)$$

[2] Indeed, the following expression is a lower bound because, among sets of the same area, the moment of inertia about s_k of any set A is the smallest when it is a circular disk centered at s_k [4, Lemma 5.3.1].

where $a := \left(\int f(z)^{2/3}\,dz\right)^{-1}$ is a normalization constant. Using (2), we can also express $c^*(K)$ and $\lambda^*(K)$ directly in terms of the request rate $r(z)$. We have $\left(\int f(z)^{2/3}\,dz\right)^{3/2}\rho = \left(\int(\rho f(z))^{2/3}\,dz\right)^{3/2} = \left(\int r(z)^{2/3}\,dz\right)^{3/2}$. Hence $c^*(K)$ in (4) can be rewritten as

$$c^*(K) = \frac{2}{3\sqrt{\pi}} \cdot \frac{||r||_{2/3}}{\sqrt{K}} \quad (6)$$

where $||r||_p$ is the L_p norm defined by

$$||r||_p = \left(\int r(z)^p\,dz\right)^{1/p}$$

and $\lambda^*(K)$ in (5) can be rewritten as

$$\lambda^*(z) = a'\,r(z)^{2/3} \quad (7)$$

where $a' := \left(\int r(z)^{2/3}\,dz\right)^{-1}$ is a normalization constant. For ease of reference, we will called λ^* the optimal server density and $c^*(K)$ the minimum cost in this paper.

This suggests a server placement strategy, when cost is measured by the Euclidean distance, where server density $\lambda^*(z)$ is proportional to the $2/3$-power of the request density, $f(z)^{2/3}$, or equivalently, of the request rate, $r(z)^{2/3}$. The strategy incurs an approximate (lower bound on) cost $c^*(K)$ that is proportional to $||r||_{2/3}$ and inversely proportional to the square root of K. Expressions (4–7) highlight the importance of *spatial* distribution of requests.

2.3 Multiple websites

Consider J websites indexed by $j = 1, 2, \ldots, J$. Suppose requests to website j has a total volume of ρ_j and a spatial density $f_j(z)$ (or equivalently, a request rate $r_j(z) = \rho_j f_j(z)$). Out of a total of K servers, k_j servers are devoted to serve website j such that

$$\sum_{j=1}^{J} k_j = K$$

and they are placed according to the optimal server density λ_j^* so that the cost associated with website j is

$$c_j(k_j) = \frac{\alpha_j}{\sqrt{k_j}}$$

where

$$\alpha_j := \frac{2\rho_j}{3\sqrt{\pi}}\left(\int f_j^{2/3}\right)^{3/2} = \frac{2}{3\sqrt{\pi}}\left(\int r_j^{2/3}\right)^{3/2}$$

Note that servers for different websites can be colocated at the same node. We will choose server allocation k_j to minimize the network cost:

$$c^*(K) = \min_{k_j} \sum_j c_j(k_j) = \sum_j \frac{\alpha_j}{\sqrt{k_j}}$$
$$\text{s. t.} \quad \sum_j k_j = K, \quad k_j \in \{0, 1, \ldots, K\}$$

For large K, relax the constraint that k_j must be integers. We hence solve the following simple convex program:

$$c(K) = \min_{k_j} \sum_j \frac{\alpha_j}{\sqrt{k_j}}$$
$$\text{s. t.} \quad \sum_j k_j = K, \quad k_j \geq 0$$

The necessary and sufficient condition for k_j^* to be optimal is the Karush-Kuhn-Tucker condition:

$$\frac{\alpha_j}{2k_j^{*\,3/2}} = \beta \quad \text{for all } j$$

which, together with the feasibility condition, yields the optimal allocation and cost:

$$k_j^* = \frac{\alpha_j^{2/3}}{\sum_l \alpha_l^{2/3}} \cdot K \quad (8)$$
$$= \frac{\int r_j^{2/3}}{\sum_l \int r_l^{2/3}} \cdot K \quad (9)$$
$$c^*(K) = \left(\sum_j \alpha_j^{2/3}\right)^{3/2} \cdot \frac{1}{\sqrt{K}} \quad (10)$$
$$= \frac{2}{3\sqrt{\pi}}\left(\sum_j \int r_j^{2/3}\right)^{3/2} \cdot \frac{1}{\sqrt{K}}$$

Remarks:

1. Recall that ρ_j represents the popularity of website j, and f_j represents the spatial density of requests for website j. They are related through the request rate $r_j(z) = \rho_j f_j(z)$. Hence, optimal allocation depends critically on website popularities as well as spatial densities of requests. Specifically, the fraction of servers devoted to website j should be proportional to $\int r_j^{2/3}$ (or equivalently, to $\int f_j^{2/3}$. The minimum cost is proportional to $\left(\sum_j \int r_j^{2/3}\right)^{3/2}$ and inversely proportional to the square-root of K.

2. We can combine equations (7) for optimal placement and (9) for optimal allocation to express optimal number of servers in a unit area for each website j directly in terms of the total number K of servers:

$$\lambda_j^*(z)k_j^* = \frac{r_j(z)^{2/3}}{\sum_l \int r_l^{2/3}} \cdot K$$

   Hence, the optimal density is proportional to $r_j(z)^{2/3}$, as a fraction of total request rate for *all* websites.

3 Allocation and placement algorithms

The approximate model in the last section suggests the following joint server allocation and placement algorithm where a fraction of servers that is proportional to $\int r_j^{2/3}$ is allocated to each website j, and the spatial distribution of these servers should be proportional to the $\frac{2}{3}$-power of the spatial distribution of requests.

Given:

1. I points (nodes) in $\Re^2$, indexed by $i = 1, 2, \ldots, I$.
2. J websites, indexed by $j = 1, 2, \ldots, J$;
3. $I \times J$ request rate matrix r where $r(i, j)$ represents the request rate from node i to website j;
4. the total number K of servers.

Joint allocation and placement algorithm:

1. Compute $a := \sum_i \sum_j r(i, j)^{2/3}$.
2. Partition $\Re^2$ into $N < I$ disjoint grids $A_1, A_2, \ldots A_N$.
3. For each grid A_n, for each website j, compute

$$b_{jn} := \frac{K}{a} \sum_{i \in A_n} r(i, j)^{2/3}$$

   Let $\lfloor b_{jn} \rfloor$ be the integer part of b_{jn} and $\check{b}_{jn} := b_{jn} - \lfloor b_{jn} \rfloor$ be the fractional part of b_{jn}.
4. Choose $\lfloor b_{jn} \rfloor$ number of nodes in grid A_n to be servers for website j. With probability $\check{b}_{jn} \in (0, 1)$, an additional node in grid A_n is chosen to be a server for website j.

4 Conclusion

By viewing server placement as a vector quantization problem in the case when both users and servers are dense, we have derived a simple server allocation and placement algorithm. Simulation results suggest that it has a good tradeoff between performance and complexity. Our results highlight the importance of both the *spatial* distributions of user requests for each content provider and the total request volume for each content provider. The total volume for each content provider represents its popularity. It has been found to follow a Zipf-like distribution [2]. It determines the server allocation: more popular contents are allocated more servers. The spatial distribution of requests determines where these servers should be placed around the network. However spatial distribution of requests is not well exploited in current systems, both because of the difficulty in measuring it empirically [7, 5, 9], and because of the lack of a theoretical understanding of its role.

Our current joint sever allocation and placement algorithm, though much simpler, still requires non-local information to implement. The idea is to have

each local region decide how many servers to employ and what contents to store based only on the fraction of local user requests for different contents, as a fraction of total volume on the network, a piece of non-local information. The scheme naturally adapts to changes in user request pattern to reduce propagation delay and balance network load. We will develop a distributed version of this algorithm that is suitable for real-time implementation on a large scale.

References

[1] S. Bhattacharjee, K. Calvert, and E. W. Zegura. Self-organizing wide-area network caches. In *Proceedings of IEEE Infocom*, March/April 1998.

[2] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *Proceedings of the IEEE Infocom*, March 1999.

[3] Syam Gadde, Jeff Chase, and Michael Rabinovich. Web caching and content distribution: A view from the interior. *Computer Communications*, 24(2), February 2001.

[4] Robert M. Gray. *Source coding theory*. Kluwer Academic Publishers, 1990.

[5] J. Kangasharju, K. W. Ross, and J.W. Roberts. Locating copies of objects using the domain name system. In *Proceedings of the 4th International Caching Workshop*, March 1999.

[6] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. II: the p-medians. *SIAM J. Appl. Math.*, 37(3):539–560, December 1979.

[7] Balachander Krishnamurthy and Jia Wang. On network-aware clustering of web clients. In *Proceedings of ACM SIGCOMM 2000*, August 2000.

[8] C. Papadimitriou. Worst-case and probabilistic analysis of a geometric location problem. *SIAM J. Comput.*, 10:542–557, 1981.

[9] A. Shaikh, R. Tewari, and M. Agrawal. On the effectiveness of DNS-based server selection. In *Proceedings of IEEE Infocom*, April 2001.

[10] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy. On the scale and performance of cooperative web proxy caching. *Operating Systems Review*, 34(5):16–31, December 1999.