

Improved Algorithms for the Data Placement Problem

Sudipto Guha*

Kamesh Munagala†

1 Introduction

We study the *data placement* problem [1, 3], where the goal is to place data objects in fixed capacity caches in a network to optimize latency of access. Each user has a demand for a specific object. The latencies on the network are modeled by a metric distance. Each cache can store a certain number of web domains or pages. In addition, each cache has a bound on the number of users¹ it can serve since the delays of serving users increases exponentially with load. Thus we are balancing load across the caches. Note that since we are allowed to replicate objects in different caches, the load balancing is not just a simple assignment. We have to first decide how many copies of each object to open, and then decide in which caches to place these objects so that we obey capacity constraints, and optimize latencies. This problem therefore includes facility location [5] and generalized assignment [4] as special cases.

The data placement problem in the context of web caching was formulated independently in [1] and [3]. In the former paper, the authors consider only the bound on the number of users, and obtain a constant approximation on the latency while blowing up the cache capacities by the size of the largest object. The latter paper considers both the bounds, but they need to blow up the capacities by a logarithmic factor to obtain a constant approximation on the latency.

2 ILP Formulation

Formally, we are given a set P of objects, each object $p \in P$ has a size denoted s_p . We are given a set U of users, each user $u \in U$ has a

demand d_u . for an object $r(u) \in P$. We are given a set C of caches. Each cache $c \in C$ has a size S , denoting the total size of the objects it can store, and a demand bound D , denoting the total amount of user demand that can be routed to this cache. We denote the latency between nodes i and j as l_{ij} .

We can formulate the data placement problem as an integer program. Let y_{pc} denote the indicator variable if object p is located in cache c , and let x_{uc} denote whether user u accesses object $r(u)$ from cache c .

$$\text{Minimize } \sum_{u \in U, c \in C} l_{uc} \cdot d_u \cdot x_{uc}$$

$$\begin{aligned} \sum_{p \in P} s_p \cdot y_{pc} &\leq S && \forall c \in C \\ \sum_u d_u \cdot x_{uc} &\leq D && \forall c \in C \\ \sum_{c \in C} x_{uc} &\geq 1 && \forall u \in U \\ x_{uc} &\leq y_{r(u)c} && \forall u \in U, c \in C \\ x_{uc}, y_{pc} &\in \{0, 1\} && \forall u, p, c \end{aligned}$$

3 The Solution Strategy

We attempt a solution by clustering a set of users that request the same page in small clusters, and then assigning these clusters to caches such that the capacity is obeyed (upto small constants). The fact that we have small clusters automatically would ensure that the demand on a cache is small. In a sense we are creating a set of virtual objects that can be accessed by a small number of users.

We modify the problem by assuming that any copy of object p can be accessed by no more than $d_p = \lceil \frac{s_p D}{S} \rceil$ users. It is easy to see that an instance of the original problem is equivalent to an instance where this condition is true, but with a blowup in capacities of the caches by at most a factor of two. We relax $y_{pc} \in \mathbf{Z}$ and add

$$\sum_{u: r(u)=p} d_u \cdot x_{uc} \leq d_p y_{pc} \quad \forall p \in P, c \in C$$

Consider now the fractional relaxation of the integer program. Let $n_p = \sum_c y_{pc}$. This denotes

*AT&T Shannon Labs, Florham Park, NJ 07932. Email: sudipto@research.att.com

†Supported by ONR N00014-98-1-0589. Department of Computer Science, Stanford University CA 94305. This work was done while the author was visiting AT&T Shannon Labs. Email: kamesh@cs.stanford.edu.

¹By a *user*, we mean a collection of users in the same geographic location, like the traffic originating from an ISP's point of presence (POP).

the fractional value of the number of copies of object p . Let $C_p = \sum_{u:r(u)=p} \sum_c l_{uc} \cdot d_u \cdot x_{uc}$. This is the optimal cost paid by all users who wish to access $p \in P$. Let C^* denote the total optimal cost.

Lemma 1. *An instance of the original problem is equivalent to an instance where this condition is true, but with a blowup in capacities of the caches by at most a factor of two.*

In the first rounding stage, we consider the objects one at a time, and open copies of the object in various locations of the network, and assign users to these objects. The caches and their capacities play no role in this rounding stage.

The integer program restricted to object p and users u s.t. $r(u) = p$ is:

$$\begin{aligned} & \text{Minimize } \sum_{u \in U} \sum_{c \in C} l_{uc} \cdot d_u \cdot x_{uc} \\ & \sum_{c \in C} x_{uc} \geq 1 \quad \forall u \in U \\ & \sum_{c \in C} y_{pc} \leq n_p \\ & \sum_{u \in U} d_u x_{uc} \leq d_p \cdot y_{pc} \quad \forall c \in C \\ & x_{uc} \leq y_{pc} \quad \forall u \in U, c \in C \\ & x_{uc}, y_{pc} \in \mathbf{Z}^+ \cup \{0\} \quad \forall u \in U, c \in C \end{aligned}$$

Let us denote the optimum objective value as C'_p . Clearly, $C'_p < C_p$. Furthermore, the constraints encode classical capacitated facility location, and we can obtain a solution in which the number of copies of p we open is no more than n_p and the average latency of users accessing p is no more than $6C_p$ while ensuring that the capacity constraints are violated by at most a factor of 4. We use the LP rounding scheme described in [5, 2]. No copy of an object has more than $4d_p$ users assigned to it.

Consider any open instance o_p of object p . Let U_{o_p} denote the users mapped to this object, and S_{o_p} denote the set of fractional objects closed while opening o_p . The rounding scheme maintains the following properties:

1. There is a specific user u_{o_p} which was connected to all the objects in S_{o_p} .
2. Without loss of generality we can assume that the distance of u_{o_p} to these objects were the smallest among all the distances between U_{o_p} and S_{o_p} , and that all these distances are equal.

4 Assignment to Caches

At the end of the previous stage, we have opened a certain number of virtual copies of each object

p and assigned the users to these copies at cost comparable to the optimal cost, while approximately satisfying the capacity constraints. We have to now pack the copies into the caches satisfying the capacity constraints.

We now consider the integer copies of the objects we opened, and their sizes s_p . We ignore the user capacities. We now find a feasible way to pack these virtual objects into the caches at cost comparable to the optimal cost. We do this as follows. Consider instance o_p of object p . We map its demand to the fractional objects S_{o_p} which we closed in opening this object, in the fraction by which these objects were open.

Lemma 2. *This fractional assignment of the demands to the caches is feasible, and has cost at most $4C^*$.*

If $y_{o_p c}$ denotes the assignment of instance o_p of object p to cache c , we have the following integer program:

$$\begin{aligned} & \text{Minimize } \sum_{p,c} l_{o_p c} \cdot d_p \cdot y_{o_p c} \\ & \sum_c y_{o_p c} \geq 1 \quad \forall o_p \\ & \sum_{o_p} s_p y_{o_p c} \leq S \quad \forall c \\ & y_{o_p} \in \{0, 1\} \quad \forall o_p \end{aligned}$$

This is just an instance of the Generalized Assignment Problem [4], and can be solved optimally while blowing up the capacity of the caches by an additive $\max_p s_p$.

Finally, we have the following theorem:

Theorem 1. *The data placement problem can be solved to a factor of 10 with page capacities being $8S + \max_p s_p$ and user capacities being $8D + \max_p d_p$.*

References

- [1] I. Baev and R. Rajaraman. Approximation algorithms for data placement in arbitrary networks. *Proceedings of 12th ACM-SIAM SODA*, 2001.
- [2] J.-H. Lin and J. S. Vitter. ϵ -approximations with minimum packing constraint violations. *Proceedings of 24th ACM STOC*, 1992.
- [3] A. Meyerson, K. Munagala, and S. Plotkin. Web caching using access statistics. *Proceedings of 12th ACM-SIAM SODA*, 2001.
- [4] D. B. Shmoys and É. Tardos. Scheduling unrelated machines with costs. *Proceedings of 4th ACM-SIAM SODA*, pages 448–454, 1993.
- [5] D. B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. *Proceedings of 29th ACM STOC*, 1997.