

Self-Organizing Algorithms for Cache Cooperation in Content Distribution Networks

Sem Borst[†], Varun Gupta^{**}, Anwar Walid[†]

[†]Alcatel-Lucent, Bell Labs, 600 Mountain Avenue, P.O. Box 636, Murray Hill, NJ 07974-0636

^{*}Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213

1. INTRODUCTION

The on-demand delivery of video content is anticipated to show tremendous growth over the next few years, driven by the huge popularity of user-generated video clips and the expansion of VoD libraries. Even stronger growth is likely to be fueled by the proliferation of IPTV services with personalized features such as CatchUp/PauseLive TV and NPVR capabilities. The common characteristic of these ‘time-shifted’ TV services, as well as VoD libraries and sites like YouTube, is that users can select from a vast collection of content material at any time they want. This runs counter to the broadcast paradigm embraced in conventional TV networks, and raises a need for unicast sessions, increasing the overall bandwidth demands by orders of magnitude.

Caching strategies provide an effective mechanism for mitigating the massive bandwidth requirements associated with large-scale distribution of personalized high-definition video content. In essence, caching strategies exploit storage capacity to absorb traffic by replicating the most popular content closer to the network edge rather than storing it in a central location. The reduction in the traffic load translates into a smaller required transport capacity and capital expense as well as fewer performance bottlenecks, thus enabling better service quality at a lower price. In the present paper we develop light-weight distributed cooperative content placement algorithms so as to maximize the traffic volume served from cache, and thus minimize the bandwidth cost. As a canonical scenario, we focus on a cluster of distributed caches, connected either directly, or via a parent node, and formulate the content placement problem as an integer linear program in order to obtain a benchmark of the globally optimal performance. Under certain symmetry assumptions, the optimal solution of the relaxed linear program is found to have a rather simple structure. Besides interesting in its own right, the knowledge of the optimal structure offers useful insight for the design of low-complexity cooperative content placement and eviction algorithms. The performance of the proposed distributed algorithms is proved to be within a small constant factor from the performance of an optimal centralized scheme, while requiring only limited knowledge of global popularities, even in asymmetric scenarios. In contrast to [1, 2], we will focus on specific topologies, motivated by real system deployments, and use the insight from the linear program and the simple structure of its optimal solution to develop efficient distributed content placement algorithms with far more favorable performance ratios than the ones in [1, 2].

*Work done during internship at Bell Labs

2. MODEL DESCRIPTION AND PROBLEM FORMULATION

We focus on a cluster of cooperative caches offering a collection of N content items of sizes s_1, \dots, s_N . The cache cluster consists of M ‘leaf’ nodes indexed $1, \dots, M$, which are connected directly, or indirectly via node 0 as a common ‘parent’ node somewhere along the path to a ‘root’ node. (It is worth emphasizing that the cache cluster need not be a stand-alone network, but could be part of a larger hierarchical tree topology as is common for IPTV networks, with some degree of logical connectivity among nodes within the same hierarchy level, either directly or via a ‘U-turn’ through a common parent node.) Node i , $i = 0, \dots, M$, has a cache of size B_i , while the root node is endowed with a cache of sufficient capacity to store all the content items. The demand for the n -th content item at node i is d_{in} , $i = 1, \dots, M$. If a requested content item is not available at a node, then it may be fetched from another node. The unit bandwidth ‘cost’ of transferring content from the root node to the parent node is c_0 , from the parent node to leaf node i is c_i , and from leaf node j to leaf node i is $c_{ij} \leq c_0 + c_i$.

We aim to determine the optimal placement of the content items, subject to the cache capacity constraints, so as to minimize the bandwidth expenses, or equivalently, maximize the bandwidth savings. We can formulate this problem as the following integer linear program by introducing 0–1 decision variables x_{in} , indicating whether the n -th content item is stored in the cache of node i or not, and x_{ijn} , indicating whether requests for the n -th content item at node i are served from the cache of node j or not.

$$\begin{aligned} \max \quad & \sum_{n=1}^N s_n \sum_{i=1}^M d_{in} \left((c_0 + c_i)x_{in} + c_0 x_{i0n} + \sum_{j \neq i} c'_{ij} x_{ijn} \right) \\ \text{sub} \quad & \sum_{n=1}^N s_n x_{in} \leq B_i, \quad i = 0, \dots, M \\ & x_{ijn} \leq x_{jn}, \quad i = 1, \dots, M, i \neq j = 0, 1, \dots, M, \forall n \\ & x_{in} + x_{i0n} + \sum_{j \neq i} x_{ijn} \leq 1, \quad i = 1, \dots, M, \forall n, \end{aligned}$$

with $c'_{ij} := c_0 + c_i - c_{ij} \geq 0$ and $\sum_{j \neq i}$ short-hand notation for $\sum_{j=1}^{i-1} + \sum_{j=i+1}^M$. Note that optimality requires $x_{iin} \equiv x_{in}$.

In the sequel we relax the integrality constraints on the decision variables, so as to obtain a fractional linear program which provides an upper bound. In several cases, the optimal fractional solution can actually be exactly achieved if we allow nodes to partially cache items, called ‘chunking’.

3. INTRA-LEVEL CACHE COOPERATION

We now focus on the special case where content can only be stored at the leaf nodes and not at the parent node, i.e., $B_0 = 0$, which corresponds to the problem considered in [4], and will be referred to as *intra*-level cache cooperation. We refer to [3] for further details and proofs.

3.1 Symmetric demands, cache sizes and costs

We start with the simplest case where $B_i \equiv B$ and $d_{in} \equiv d_n$ for all $i = 1, \dots, M$. Further, we assume the leaf nodes to be symmetric in terms of bandwidth cost, i.e., $c_i \equiv c$ and $c_{ij} \equiv c'$. For compactness, denote $c'' := M(c_0 + c) - (M-1)c' = M(c_0 + c - c') + c'$. The problem of maximizing the bandwidth savings then reduces to a fractional knapsack problem, with a knapsack of size MB and $2N$ items of sizes $b_n = s_n$, $b_{N+n} = (M-1)s_n$, $n = 1, \dots, N$. The value of item n is $c''d_n$, while the value of item $N+n$ is $(M-1)c'd_n$. Assume that the content items are indexed in descending order of d_n , i.e., $d_1 \geq d_2 \geq \dots \geq d_N$. The optimal content placement then has a relatively simple structure:

- (i) For some N_1 , items $1, \dots, N_1$ are replicated at each node;
- (ii) Item N_1+1 is possibly replicated but not at all the nodes;
- (iii) For some $N_2 > N_1+1$, items N_1+1, \dots, N_2 have a single copy in the cluster.

Based on the above insights, we now propose a simple distributed algorithm, **Local-Greedy**, for cooperative caching, where each node modifies its local content so as to achieve the largest gain in global utility. For convenience, we assume unit-size content items, i.e., $s_n = 1$ for all $n = 1, \dots, N$.

Algorithm Local-Greedy

At the turn of node i , it:

1. associates utility $U_n = d_n c''$ for an item n if item n is not cached anywhere in $[M] \setminus \{i\}$;
2. associates utility $V_n = d_n c'$ for an item n if item n is cached somewhere in $[M] \setminus \{i\}$;
3. replaces its contents with B items of the highest value.

While in the quite restricted case of symmetric demands, cache sizes and bandwidth costs, it might seem that the **Local-Greedy** algorithm should be able to reach the globally optimal configuration, this is not always the case as the next theorem shows.

THEOREM 3.1. *Under symmetric demands, cache sizes and bandwidth costs, Local-Greedy is a $\frac{4}{3}$ -approximation algorithm for the metric of maximizing bandwidth savings. Further, there exist demand vectors for which Local-Greedy can not achieve more than $\frac{3}{4}$ th of the optimal.*

3.2 Arbitrary demands, cache sizes and costs

In this section we provide matching upper and lower bounds for the performance of **Local-Greedy** under arbitrary demand vectors, cache sizes and bandwidth costs.

THEOREM 3.2. *Local-Greedy is a 2-approximation algorithm for the metric of maximizing bandwidth savings under arbitrary demands, cache sizes and bandwidth costs. Further, there exist demand vectors for which Local-Greedy can not achieve more than $\frac{1}{2}$ of the optimal, even under equal cache sizes and symmetric bandwidth costs.*

Observe that when $c_{ij} \equiv c'_j$, i.e., the bandwidth cost between peers depends only on the source node, to implement

Local-Greedy, leaf nodes only require information about local and aggregate global demand for content items.

4. INTER-LEVEL CACHE COOPERATION

We now focus on the special case where content can only be fetched from the parent node and not from a peer, i.e., $c_{ij} = \infty$, which will be referred to as *inter*-level cache cooperation.

4.1 Homogeneous demands, equal cache sizes

We assume that the leaf nodes have equal cache sizes, i.e., $B_i \equiv B$ for all $i = 1, \dots, M$, and that the demand characteristics satisfy $d_{i1} \geq d_{i2} \geq \dots \geq d_{iN}$ for all $i = 1, \dots, M$, possibly after suitable reindexing of the content items. Note that the demand characteristics are no longer assumed to be symmetric, and could for example only be identical up to node-dependent scaling factors.

The optimal content placement then has a relatively simple structure. Specifically, the most popular content items are fully replicated in all the leaf nodes. Single copies of the second-tier items are stored in the cache of the parent node.

It is worth observing that the optimal content placement can be achieved through a simple ‘greedy’ strategy where each individual node aims to maximize the local hit rate, i.e., the fraction of traffic served from cache of the requests it receives as follows: Each of the leaf nodes stores the most popular content items. The parent node will not receive any requests for the most popular items, and ends up storing the second-tier items.

4.2 Arbitrary demands, cache sizes

We now analyze the performance of the greedy content placement strategy under arbitrary demands and cache sizes. While the strategy will no longer yield the optimal content placement in general, it is guaranteed to achieve a certain fraction of the maximum achievable bandwidth savings as the next theorem shows.

THEOREM 4.1. *For arbitrary cost values, item sizes, and cache sizes, the greedy content placement strategy is guaranteed to achieve at least a fraction $\frac{(M-1)c_{\min} + M c_0}{(M-1)c_{\min} + (2M-1)c_0} \geq \frac{M}{2M-1}$ of the maximum achievable bandwidth savings, with $c_{\min} := \min_{i=1, \dots, M} c_i$.*

The performance ratio in the above proposition is tight, and attained in (unnatural) cases where some items are highly popular in some leaf nodes and not popular at all in others.

5. REFERENCES

- [1] I.D. Baev, R. Rajaraman (2001). Approximation algorithms for data placement in arbitrary networks. *Proc. SODA 2001*, 661–670.
- [2] I.D. Baev, R. Rajaraman, C. Swamy (2008). Approximation algorithms for data placement problems. *SIAM J. Comput* **38**, 1411–1429.
- [3] S.C. Borst, V. Gupta, A. Walid (2008). Self-organizing algorithms for cache cooperation in content distribution networks. Technical Memorandum, Alcatel-Lucent, Bell Labs.
- [4] A. Leff, J.L. Wolf, P.S. Yu (1993). Replication algorithms in a remote caching architecture. *IEEE Trans. Parallel Distr. Syst.* **4** (11), 1185–1204.