On the Sensitivity of Cooperative Caching Performance to Workload and Network Characteristics

Kang-Won Lee Khalil Amiri Sambit Sahu Chitra Venkatramani
IBM Thomas J. Watson Research Center
Email: {kangwon, amirik, sambits, chitrav}@us.ibm.com

1. INTRODUCTION

A rich body of literature exists on several aspects of cooperative caching [1, 2, 3, 4, 5], including object placement and replacement algorithms [1], mechanisms for reducing the overhead of cooperation [2, 3], and the performance impact of cooperation [3, 4, 5]. However, while several studies have focused on quantifying the performance benefit of cooperative caching, their conclusions on the effectiveness of such cooperation vary significantly. The source of this apparent disagreement lies mainly in their different assumptions about workload and network characteristics, and about the degree of cooperation among caches.

To more comprehensively evaluate the practical benefit of cooperative caching, we explore the sensitivity of the benefit of cooperation to workload characteristics such as *object popularity distribution, temporal locality, one time referencing behavior*, and to network characteristics such as *latencies between clients, proxies, and servers*. Furthermore, we identify a critical workload characteristic, which we call *average access density*, and show that it has a crucial impact on the effectiveness of cooperative caching.

In this extended abstract, we report on a few important results selected from our extensive study reported in [6]. In particular, assuming an LFU-based cache management policy, we arrive at the following conclusions. First, cooperative caching is only effective when the *average access density* (defined as the ratio of the number of requests to the number of distinct objects in a time window) is relatively high. Second, the effectiveness of cooperative caching decreases as the skew in object popularity increases. Higher skew means that only a small number of objects are most frequently accessed reducing the benefit of larger caches, and therefore of cooperation.

2. COOPERATIVE CACHING TAXONOMY

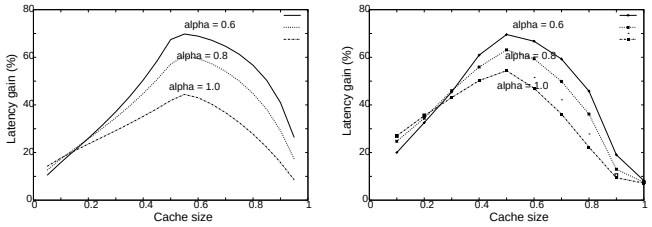
In this study, we consider a cluster of proxy caches, each serving a fixed client population, and employing a replacement policy driven by the objective of minimizing the average access latency across a particular group of clients. We classify cooperative caching algorithms according to two key aspects: (a) the level of cooperation in serving a requested object, and (b) the level of cooperation in object replacement decisions. Based on these two aspects, we can organize cooperative caching algorithms into three broad classes. Under the first class, or **No cooperation (NC)**, if a requested object is not available in the local proxy, it is fetched from the origin server. Under the second class, called **Cooperative lookup/selfish replacement (CSR)**, proxies cooperate in serving a request, but not in making object replacement decisions. If a requested object is not found in the local proxy, it is fetched from a neighboring proxy if possible. Otherwise, the request is directed to the origin server [2, 4, 5]. Replacement decisions at a proxy, however, strive to minimize object access latency only for a proxy's own clients. Finally, under the third class, referred to as **Cooperative lookup with cooperative replacement (CCR)**, proxies cooperate in serving requests as in CSR above. In addition, object replacement decisions at a proxy attempt to minimize average access latency for clients served by *all* proxies in the cluster [1, 3].

3. ANALYSIS

In this analysis, we assume that the proxies employ an LFU replacement policy and that the access frequency is *homogeneous* across proxies. That is, we assume that the access frequency of an object is the same across all proxies, i.e., $f_x(i) = f_y(i)$, where $f_x(i)$ represents the popularity of object i at proxy x, calculated considering only the clients directly served by proxy x. We further assume that each proxy can store up to k objects. Denoting the access latency between clients and their local proxies by τ_l, between neighboring proxies by τ_c, and between proxies and the origin server by τ_s, we can compute the steady state object access latency for a special case of a *two-proxy cluster* as follows: In steady state, NC ideally caches the k most popular objects at each proxy. Because of the homogeneity assumption, the two proxies would contain the exact same set of k objects. Furthermore, CSR, which is simply NC augmented with the ability to redirect misses to the other proxy, would observe no additional benefit over NC. The average latency for NC and CSR is therefore given by:

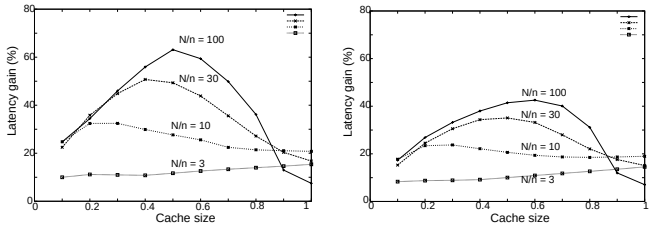
$$L_{CSR} = L_{NC} = \tau_l \sum_{i=1}^{n} f(i) + \tau_s \sum_{i=k+1}^{n} f(i) \qquad (1)$$

Computing the average access latency under CCR is much more involved since it requires knowledge of the cache contents under CCR in steady state. Unlike NC or CSR, the CCR policy may bring in a less popular object to replace one of the duplicates in either proxy. In this way, the total space available for caching in the cluster gets expanded. CCR performs such expansion only when the average access latency would decrease. To compute the cache content under CCR, our approach relies on an *imaginary* process



(a) Analysis (CCR/NC) (b) Simulation (CCR/NC)

Figure 1: Latency gain vs. Zipf parameter α.



(a) Simulation(CCR/NC) (b) Simulation(CSR/NC)

Figure 2: Latency gain vs. average access density.

which starts from the steady state contents of a two-proxy cluster under CSR, with identical objects in both caches, and proceeds by replacing some duplicates with other objects to minimize the aggregate latency across all cluster clients. We call this the *duplicate removal process (DRP)*. We then estimate the latency reduction due to CCR by aggregating the savings induced by DRP. After replacing j duplicates with the most accessed uncached objects, the latency with CCR or L_{CCR} is:

$$L(j) = \tau_l \sum_{i=1}^{n} f(i) + \frac{\tau_c}{2} \sum_{i=k-j+1}^{k+j} f(i) + \tau_s \sum_{i=k+j+1}^{n} f(i). \quad (2)$$

Several empirical studies have shown that the popularity of HTTP objects follows a Zipf-like distribution [7], i.e., the frequency of request to the i^{th} popular object is proportional to $\frac{1}{i^{\alpha}}$, where α is the parameter that determines the slope of the popularity distribution curve. When $\alpha = 1$, we can predict the upper bound on the latency gain of CCR over NC, which we define as the latency improvement due to CCR normalized by L_{NC}, as follows:

THEOREM 1 (UPPER BOUND ON CCR BENEFIT). *The latency gain of CCR over NC defined as* $\Gamma_{\frac{CCR}{NC}} = \frac{L_{NC} - L_{CCR}}{L_{NC}}$ *for user access pattern that follows a Zipf object popularity distribution is upper-bounded by*

$$\Gamma_{\frac{CCR}{NC}} = \frac{\tau_s \sum_{i=k+1}^{k+j} f(i) - \frac{\tau_c}{2} \sum_{i=k-j+i}^{k+j} f(i)}{\tau_l \sum_{i=1}^{n} f(i) + \tau_s \sum_{i=k+1}^{n} f(i)}$$

where $j = k - \frac{\tau_c}{\tau_s}(k + 0.5) + 1$.

Note that $0 \leq \Gamma \leq 1$, where $\Gamma = 0$ when there is no benefit, and $\Gamma = 1$ when there is maximum benefit, or the resulting latency is zero.

4. PERFORMANCE EVALUATION

In this section, we evaluate the effectiveness of cooperative caching through simulation and compare our results against those predicted by the analytical model presented above. The simulation presumes a network configuration where $\tau_s/\tau_c = 10$, and $\tau_s/\tau_l = 20$. To generate client request traces, we used the ProWGen workload generator [7]. We select two key results from our study that we focus on here: (a) our simple analysis makes a reasonably good prediction of the performance benefit of cooperation, and (b) cooperative caching benefit is sensitive to the average access density.

Figure 1 shows the results predicted by our analysis and those produced by the simulation. The y-axis represents the latency gain and the x-axis represents the normalized cache size with respect to the number of distinct objects. The plots show that the analytical results approximately agree with the simulation results for CCR, despite the simplifying assumptions made in our analysis. Furthermore, the plots indicate that the *potential* gain due to cooperative caching is significant, reaching 70%. Lower values of α represent less skew in the popularity distribution and suggest a larger working set size. Since cooperation is most effective when the working set is large, its gain is greatest when α is lowest. Note that the gain from cooperation is substantial even under large cache sizes. This is because cooperation reduces the cost of compulsory misses by fetching the missed objects from a neighbor rather than the origin server.

Figure 2 plots the latency gain against *access density*, which is defined as N/n, where N is the number of requests in a given time window (about a day in our experiments), and n is the number of distinct objects referenced in that window. From the figure, we observe that the gain from cooperation decreases as the access density decreases. The empirical values of N/n vary widely from 2 or 3 (in Web proxy traces) to 100 or more (in Web server environments).

We therefore conclude that the benefits of cooperative caching will be most pronounced in an environment where the access density is high and the popularity distribution does not exhibit a high skew. In particular, cooperative caching will be marginally beneficial in a Web proxy environment since the access density is low, because of the diversity in user interests and the expiration of cached objects when new versions are continuously published by origin sites.

5. REFERENCES

[1] M. Rabinovich, J. Chase, and S. Gadde, "Not all hits are created equal: cooperative proxy caching over a wide-area network," *Computer Networks and ISDN Systems*, vol. 30, no. 22–23, 1998.

[2] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," in *Proc. of ACM SIGCOMM*, September 1998.

[3] M. R. Korupolu and M. Dahlin, "Coordinated Placement and Replacement for Large-Scale Distributed Caches," in *Proc. of the IEEE Workshop on Internet Applications*, July 1999.

[4] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. Levy, "On the scale and performance of cooperative Web proxy caching," in *Proc. of ACM SOSP*, December 1999.

[5] S. G. Dykes and K. A. Robbins, "A Viability Analysis of Cooperative Proxy Caching," in *Proc. of IEEE INFOCOM*, April 2001.

[6] K.-W. Lee, K. Amiri, S. Sahu, and C. Venkatramani, "Understanding the Potential Benefits of Cooperation among Proxies: Taxonomy and Analysis," RC22173, IBM Research Report, September 2001.

[7] M. Busari and C. Williamson, "On the Sensitivity of Web Proxy Cache Performance to Workload Characteristics," in *Proc. of IEEE INFOCOM*, April 2001.