On Allocating Cache Resources to Content Providers

Weibo Chu[1], Mostafa Dehghan[2], Don Towsley[2], and Zhi-Li Zhang[3]
[1]Northwestern Polytechnical University, Xi'an, China
[2]University of Massachusetts, Amherst, USA
[3]University of Minnesota, Minneapolis, USA
wbchu@nwpu.edu.cn, {mdehghan, towsley}@cs.umass.edu, zhzhang@cs.ucsc.edu

ABSTRACT

In-network cache deployment is recognized as an effective technique for reducing content access delay. Caches serve content from multiple content providers, and wish to provide them differentiated services due to monetary incentives and legal obligations. Partitioning is a common approach in providing differentiated storage services. In this paper, we propose a *utility-driven cache partitioning* approach to cache resource allocation among multiple content providers, where we associate with each content provider a utility that is a function of the hit rate to its content. A cache is partitioned into slices with each partition being dedicated to a particular content provider. We formulate an optimization problem where the objective is to maximize the sum of weighted utilities over all content providers through proper cache partitioning, and mathematically show its convexity. We also give a formal proof that partitioning the cache yields better performance compared to sharing it. We validate the effectiveness of cache partitioning through numerical evaluations, and investigate the impact of various factors (*e.g.*, content popularity, request rate) on the hit rates observed by contending content providers.

CCS Concepts

•**Networks** → **Network management;**

Keywords

Information-Centric Networking; Resource Allocation; Cache Partitioning.

1. INTRODUCTION

The Internet today is filled up with data traffic caused by the transmission of a wide range of content accessed by users, *e.g.*, videos, audios, pictures. Content is constantly generated by a variety of applications such as online social networks, video streaming services, and file sharing platforms, to name a few. The massive amount of data traffic generated places an enormous burden on the Internet. For example, 300 hours of video are uploaded to YouTube every minute [17]. The increase in data traffic over past years is predicted to continue more aggressively, with global Internet traffic in 2019 estimated to reach 64 times of its volume in 2005 [1].

To better support global information publication, dissemination and retrieval, various solutions have been proposed. Among them, *content-centric* or *information-centric* networking (ICN) has received significant attention [2]. ICN tackles the problem by explicitly treating content as its first-class citizens, and adopting two new mechanisms, namely, *name-based routing* and *systematic in-network caching*. In ICN, every piece of content is identified by an addressable name and requests for content can be routed by network, hence it realizes the so-called *location-independent* (or *location-unaware*) content delivery. Furthermore, network storage becomes an integral part of the network substrate, where content can be cached and forwarded to satisfy future requests.

As in-network caching is fundamental to realizing efficient large-scale content delivery, the design of caching mechanisms for ICNs has become a significant research field, generating a flurry of recent studies. In this paper, being cognizant that content providers (CPs) will play a key role in allocating and utilizing cache resources in future ICNs, we go one step further to study the problem of *how to allocate network cache resources of an ICN among multiple CPs so as to achieve the maximum cache utilization efficiency or differentiated services/fairness among them?* This is a challenging problem since, according to current ICN implementations, caches are often managed by traditional policies such as LRU, FIFO, *etc.* These policies treat content from different CPs in a tightly coupled manner, making it rather difficult for the cache provider to optimize cache efficiency and implement differentiated services/fairness among CPs.

In this paper, we propose a *utility-driven cache partitioning* approach to address the aforementioned problem. We associate with each content provider a utility that is a function of the hit rate to its content. The cache provider then partitions the cache into slices with each slice being dedicated to a particular content provider. By restricting content contention for cache space into partitions for each CP, cache partitioning decouples the interactions among different CPs, and provides a natural means to realize differentiated services and fairness among them. Cache partitioning also potentially improves cache utilization efficiency since it prevents the eviction of a more popular object from one CP

ICN'16, September 26-28, 2016, Kyoto, Japan

DOI: http://dx.doi.org/10.1145/2984356.2984371

by a less popular object from another CP. For LRU caches, we in fact prove that sharing the cache is sub-optimal and partitioning yields better performance.

We formulate a utility-based optimization framework for maximizing a weighted sum of content provider utilities subject to cache capacity constraint at the cache service provider. We argue that such a framework is suitable for designing *online* cache partitioning algorithms for differentiated services, and has implications for the establishment of revenue based models where content providers react to prices set by (cache) service providers based on their utility functions.

The remainder of this paper is organized as follows. A brief overview of related work is given in Section 2. Section 3 describes the problem setting and basic model. In Section 4, we formulate the cache partitioning problem, establish its convexity, and give a formal proof that cache partitioning provides performance gain over sharing for LRU caches. We present results from numerical evaluations in Section 5. Section 6 discusses future research directions, and Section 7 concludes the paper.

2. RELATED WORK

The issue of differentiated quality of service (QoS) in the context of web cache management has been extensively studied (e.g., see [3, 5] and references therein). Cache partitioning has been widely accepted as an effective means to provide performance enhancement for applications/processes in shared memory caches [10, 14, 16]. Kelly *et al.* [9] propose a biased replacement policies for web caches to provide differentiated quality of service (QoS) through prioritizing cache space across servers. A QoS architecture for a shared proxy cache is presented in [11] with hit rate guarantees for competing content classes. Lu *et al.* [12] describe a differentiated cache service architecture and propose a control-theoretic approach to cache resource management. Feldman and Chuang [5] develop a QoS caching mechanism through referential storage allocation and object transition based on priority queues. A general cache partitioning model for supporting multiple QoS classes by considering content priority and popularity is also presented in [6].

In the context of ICNs, however, very little effort has been dedicated to service differentiation or QoS guarantees. To the best of our knowledge, there's no existing work that explicitly deals with cache allocation among multiple content providers in ICNs through cache partitioning. Perhaps most closely related to our work is the study in [8] where a game-theoretic cache allocation approach is developed. This approach requires the content providers to report the true demands from their content access. In contrast, we develop a general utility maximization framework for studying the multi-CP cache allocation problem.

3. MODEL DESCRIPTION

Consider a cache of size C that is accessed by users for content generated by K content providers (CPs), as shown in Figure 1. Each CP k serves a set $F_k = \{f_{1k}, f_{2k}, \ldots, f_{N_k k}\}$ of $N_k = |F_k|$ equal size files that are distinct from files served by other CPs. Throughout this paper, we will use the terms content and file interchangeably. We assume that each file resides permanently at a content custodian that the cache can access. Requests for file f_{ik} from users arrive at the cache according to a Poisson process with rate λ_{ik}.

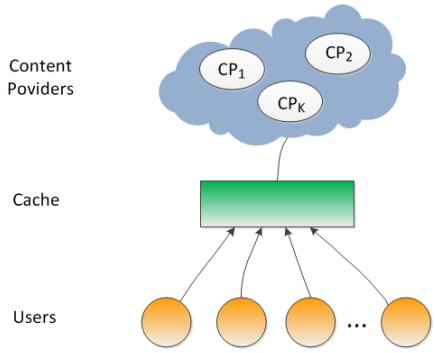


Figure 1: Network of single cache and multiple content providers.

The cache provider partitions the cache into (virtual) slices and allocates one slice to each CP, thereby restricting cache contention to the set of content objects owned by one CP. It is assumed that each cache slice employs the Least Recently Used (LRU) file replacement policy.

3.1 Cache Characteristic Time

For an LRU cache with Poisson arrivals, Che *et al.* [3] introduced the notion of *cache characteristic time* for the cache hit probability approximation: the probability o_i that a request for file i results in a hit can be approximated by

$$o_i = 1 - e^{-\lambda_i T},$$

where λ_i is the request rate for file i, and T is a constant denoting the *characteristic time* of the cache and is computed as the unique solution to the equation

$$\sum_{i=1}^{N} (1 - e^{-\lambda_i T}) = C,$$

where N is the number of files accessing the LRU cache of capacity C. The notion of cache characteristic time has been theoretically justified and shown to perform well in practice [7]. This notion has been generalized to a wider range of caching policies and more general arrival processes [13]. According to the above calculations, the average hit rate for a set of N files through an LRU cache can be expressed as follows

$$h = \sum_{i=1}^{N} \lambda_i (1 - e^{-\lambda_i T}).$$

4. CACHE PARTITIONING

Content providers have business relations with the cache provider and pay for cache resources. Associated with each CP k is a utility function $U_k(h_k)$ that captures the "satisfaction" perceived by CP k for observing hit rate h_k for its contents. Each $U_k(\cdot)$ is assumed to be a continuous, monotonically increasing, and concave function. The cache provider aims to maximize the overall cache utilization or the (weighted) sum of the utilities of the content providers so as to maximize its own profit.

To differentiate the service provided to different CPs, the cache provider partitions the cache into K slices, and allocates a partition of size C_k to CP k. Our goal is to find the optimal partitioning of the cache. Hence, we are interested

in the following optimization problem,

$$\begin{aligned} \text{maximize} \quad & \sum_{k=1}^{K} w_k U_k\Big(h_k(C_k)\Big) \\ \text{such that} \quad & \sum_{k=1}^{K} C_k \le C \\ & C_k \ge 0, \quad k = 1, 2, \ldots, K, \end{aligned} \tag{1}$$

where w_k is a weight chosen by the cache provider which reflects business preferences such as financial incentives or legal obligations. In the simple case where $w_k = 1$ and $U_k(h_k) = h_k$, the objective becomes maximizing the overall cache hit rate, which provides a measure of the cache utilization efficiency.

As mentioned in Section 3, it is assumed that the cache partitions employ the LRU policy. This allows us to adopt the notion of *cache characteristic time* to evaluate the cache hit rate for each partition. More specifically, for each CP k accessing cache slice with size C_k, the probability that a request for file f_{ik} results in a hit can be approximated by $o_{ik} = 1 - e^{-\lambda_{ik} T_k}$, where λ_{ik} is the request rate for file f_{ik}, and T_k is a constant denoting the *characteristic time* of the cache slice allocated to CP k. As explained in Section 3.1, T_k is computed as the unique solution to the equation:

$$\sum_{i=1}^{N_k} (1 - e^{-\lambda_{ik} T_k}) = C_k.$$

The hit rate for CP k is then expressed as

$$h_k = \sum_{i=1}^{N_k} \lambda_{ik} (1 - e^{-\lambda_{ik} T_k}).$$

We are now ready to study the properties of formulation (1). In the remainder of this section, we prove that the partitioning problem has a unique optimal solution. Moreover, we prove that partitioning the cache achieves better utilization than sharing the cache among all CPs. We start with the following lemma:

Lemma 1. *The hit rate h_k is an increasing and concave function of C_k.*

Proof. See Appendix A. □

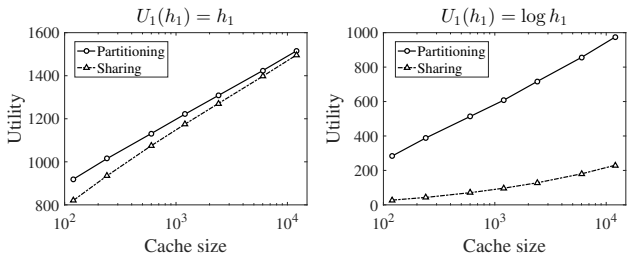
Theorem 1. *The single cache resource allocation problem* (1) *has a unique optimal solution.*

Proof. In Lemma 1, we show that h_k is concave in C_k. Since U_k is also concave in h_k, the objective function in (1) is concave. Furthermore, since the feasible solution set is convex, a unique maximizer, called the optimal solution, exists. □

The following theorem establishes that for K content providers, partitioning the cache into K slices provides better performance gains compared to sharing the whole cache.

Theorem 2. *Sharing the entire cache among multiple content providers is sub-optimal.*

Proof. Through the notion of cache characteristic time, we give a mathematical reasoning in Appendix B showing that sharing the entire cache among content providers is equivalent to partitioning the cache in a specific way, and hence leaves room for further performance enhancement through optimizing the partition sizes. From the perspective of contents, this phenomenon can be explained as follows: partitioning the cache and allocating each content provider a dedicated slice can prevent the eviction of a more popular object from one CP by a less popular object from another CP, thereby increasing cache efficiency. A complete proof is given in Appendix B. □



(a) (b)

Figure 2: Showing the efficacy of cache partitioning. Aggregate utility obtained by partitioning versus sharing, for (a) $U_1(h_1) = h_1$, and (b) $U_1(h_1) = \log h_1$. In both cases, $U_2(h_2) = h_2$.

4.1 Implications

Using different utility functions in the optimization formulation (1) yields different partition sizes for the content providers. In this sense, each utility function defines a notion of fairness in allocating storage resources to different content providers. The family of β-fair utility functions expressed as

$$U_k(h_k) = \frac{h_k^{1-\beta}}{1-\beta},$$

unifies different notions of fairness in resource allocation [15]. Some choices of β lead to especially interesting utility functions. For example, $\beta = 0$ yields $U_k(h_k) = h_k$ and the objective $\max \sum_k h_k$ is to maximize the overall hit rate. With $\beta \to 1$, we have $U_k(h_k) = \log h_k$ and the objective $\max \sum_k \log h_k$ is known to implement *proportional* fairness. Also, $\beta \to \infty$ yields the objective $\max \min_k h_k$, which corresponds to *max-min* fairness.

With such notions of fairness associated with utility functions, our utility-driven cache partitioning formulation provides a general and unified framework for allocating cache resources for implementing *fair* cache partitioning algorithms.

5. NUMERICAL EVALUATION

In this section, we perform numerical experiments to evaluate the efficacy of cache partitioning, and to understand the effect of various parameters on the performance observed by content providers. For our base case, we consider two content providers serving $N_1 = 2 \times 10^5$ and $N_2 = 10^6$ content files. File popularities for the two content providers follow Zipf distributions with parameters $\alpha_1 = 1.2$ and $\alpha_2 = 0.8$, respectively. Requests for the files from the two content providers arrive as Poisson processes with aggregate rates $\lambda_1 = 1500$ and $\lambda_2 = 1000$. Unless otherwise specified, we assume that the utilities of the two content providers are expressed as $U_1(h) = U_2(h) = h$. Moreover, we assume the two content providers are equally important to the service provider, *i.e.*, $w_1 = w_2 = 1$.

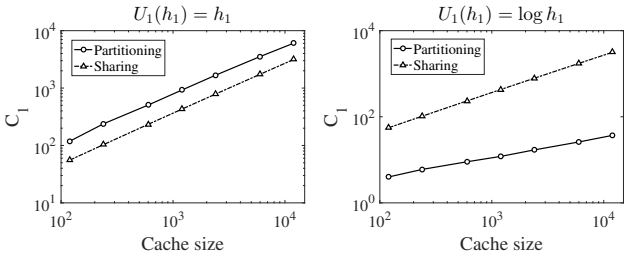


Figure 3: Cache size allocated to CP1 when partitioning the cache compared to average cache storage consumed by CP1 files when (a) $U_1(h_1) = h_1$, and (b) $U_1(h_1) = \log h_1$. In both cases, $U_2(h_2) = h_2$.

5.1 Sharing vs. Partitioning

To understand the efficacy of cache partitioning, we compute the utility obtained by solving the optimization problem (1), and compare it to the utility obtained when sharing the cache without partitioning. Figure 2 shows the utility obtained with and without partitioning the cache when (a) $U_1(h_1) = h_1$, and (b) $U_1(h_1) = \log h_1$. It is clear that, in both cases, partitioning achieves better performance than sharing. When $U_1(h_1) = h_1$, more than 10% increase in total utility is observed for some cache sizes. With $U_1(h_1) = \log h_1$, the overall utility is increased more than 90% at some points. These results also indicate that the major gain in partitioning is observed when a content provider has a utility function other that $U(h) = h$.

To understand how the partition sizes C_1 and $C_2 = C - C_1$ allocated to each CP change as cache size increases, we compare C_1 obtained by partitioning to the average cache storage used by CP1 when sharing the cache. Figure 3 shows how C_1 changes with cache size. When $U_1(h_1) = h_1$, partitioning allocates more cache space to CP1 than it would receive if the cache was shared. As cache size increases, the change in C_1 appears to be independent of partitioning or sharing the cache in this case. With $U_1(h_1) = \log h_1$, however, CP1 is allocated a smaller portion of the cache when it is partitioned. Moreover, with partitioning, C_1 seems to increase with a smaller rate as the cache size increases.

5.2 Parameter Impact

We next look at the effect of various parameters on cache partitioning and hit rates observed by the content providers. We fix the parameters of CP2, and study the effect of changing the aggregate request rate λ_1 of CP1, the weight parameter w_1, the skewness parameter α_1 for the Zipfian file popularity distribution, and the number of files N_1 served by CP1. Cache size is set to $C = 1200$ (which can hold 0.1% of all contents). Figure 4 shows the effect of each parameter on the hit rates and partition sizes for both content providers.

Arrival rate: As λ_1 increases, a larger partition is allocated to CP1. This results in a lower hit rate for CP2 but the hit rate for CP1 increases such that the aggregate hit rate increases. Also, when the cache is shared, contents from CP1 occupy a larger fraction of the cache as λ_1 increases.

Weight: Clearly, the weight w_1 has no effect on cache performance when it is shared. With partitioning, as it is expected, when w_1 increases, a larger fraction of the cache is allocated to CP1. Hence, the hit rate observed by CP1 increases. The opposite happens to CP2.

Skewness parameter: When the cache is shared, a smaller fraction of the cache is occupied by contents from CP1 as α_1 increases. This is because with a larger α_1 a smaller fraction of CP1 files generates most of the traffic for CP1. When partitioning the cache, the partition size of CP1 does not show an incremental increase or decrease with α_1, but is adjusted to increase the aggregate hit rate.

Number of files: When sharing the cache, increasing the number of files served by CP1 results in a larger cache fraction occupied by CP1 files. The opposite is true when the cache is partitioned, as the aggregate hit rate needs to be maximized. CP1 observes a smaller hit rate as N_1 increases, with and without sharing the cache.

6. DISCUSSION

In this section, we explore the implications of utility-driven cache partitioning framework and present some future research directions.

6.1 Developing online algorithms

The framework developed in (1) provides a means to compute the optimal partition sizes for the content providers given that all the information such as the request rates and number of files served by each content provider is known. System parameters change over time, and hence, cache providers need online algorithms to implement the optimal strategy and adapt to changes by collecting limited information. We argue that similar techniques as in [4] can be used based on the dual formulation of (1) to develop such algorithms.

6.2 Alternate formulation

The formulation presented in (1) considers maximizing a weighted sum of content provider utilities and focuses on differentiated services to content providers. It is also possible to consider a user-centric formulation with the goal of providing differentiated services to end-users. For example, we can define a user utility as a function of the average content access delay perceived by the user. Then, the goal would be to size the partitions allocated to content providers such that a weighted sum of user utilities is maximized.

7. CONCLUSION

We study the problem of allocating cache resources among multiple content providers in ICNs, and propose a utility-driven cache partitioning approach. Cache partitioning restricts content contention of each CP within its dedicated slice and hence it allows the cache provider to tune the performance of each CP through adjusting the size of its allocated cache slice. We associate with each CP a utility that is a function of its performance, and propose an optimization framework with the objective to maximize the aggregate utilities over all CPs through proper cache partitioning. Our framework supports the implementation of differentiated services and wide notions of fairness, and has implications for the establishment of a viable cache market economic model for ICNs.

Acknowledgement

This work was conducted under NSFC grant 61502393, NSF grants CNS-1413998, CRI-1305237 and CNS-1411636.

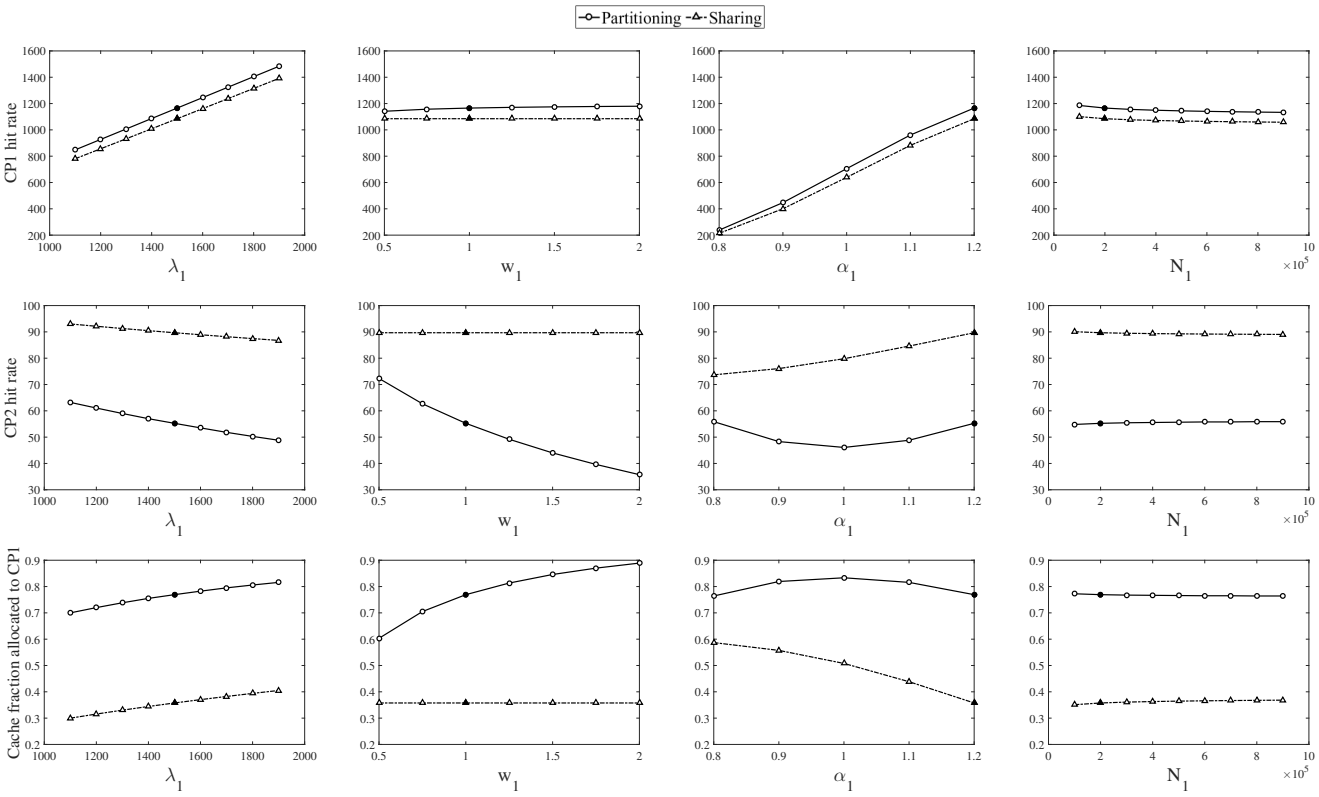


Figure 4: Effect of the parameters on hit rates and partition sizes allocated to content providers.

8. REFERENCES

[1] Cisco visual networking index: Forecast and methodology, 2014-âĂŞ2019. White paper, May 2015.

[2] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. A survey of information-centric networking. *IEEE Communications Magazine 50*, 7 (2012), 26–36.

[3] Che, H., Wang, Z., and Tung, Y. Analysis and design of hierarchical web caching systems. In *INFOCOM* (2001), pp. 1416–1424.

[4] Dehghan, M., Massoulie, L., Towsley, D., Menasche, D., and Tay, Y. A utility optimization approach to network cache design. In *INFOCOM* (2016).

[5] Feldman, M., and Chuang, J. Service differentiation in web caching and content distribution. In *IASTED CCN* (2002).

[6] Feng, W., and Zhang, Y. A general cache partition model for multiple qos classes: algorithm and simulation. In *IEEE CIMCA-IAWTIC* (2005), vol. 2, pp. 544–549.

[7] Fricker, C., Robert, P., and Roberts, J. A versatile and accurate approximation for lru cache performance. In *ITC* (2012).

[8] Hoteit, S., El Chamie, M., Saucez, D., and Secci, S. On fair network cache allocation to content providers. *Elsevier Computer Networks* (2016).

[9] Kelly, T., Chan, Y. M., Jamin, S., and MacKie-Mason, J. K. Biased replacement policies for web caches: Differential quality-of-service and aggregate user value. In *International Web Caching Workshop* (1999).

[10] Kim, S., Chandra, D., and Solihin, Y. Fair cache sharing and partitioning in a chip multiprocessor architecture. In *IEEE PACT* (2004), pp. 111–122.

[11] Ko, B.-J., Lee, K.-W., Amiri, K., and Calo, S. Scalable service differentiation in a shared storage cache. In *IEEE ICDCS* (2003), pp. 184–193.

[12] Lu, Y., Abdelzaher, T. F., and Saxena, A. Design, implementation, and evaluation of differentiated caching services. *IEEE Transactions on Parallel and Distributed Systems 15*, 5 (2004), 440–452.

[13] Martina, V., Garetto, M., and Leonardi, E. A unified approach to the performance analysis of caching systems. In *INFOCOM* (April 2014), pp. 2040–2048.

[14] Qureshi, M. K., and Patt, Y. N. Utility-based cache partitioning: A low-overhead, high-performance, runtime mechanism to partition shared caches. In *IEEE/ACM International Symposium on Microarchitecture* (2006), pp. 423–432.

[15] Srikant, R., and Ying, L. *Communication networks: an optimization, control, and stochastic networks perspective*. Cambridge University Press, 2013.

[16] Suh, G. E., Rudolph, L., and Devadas, S. Dynamic partitioning of shared cache memory. *The Journal of Supercomputing 28*, 1 (2004), 7–26.

[17] Thomee, B., Shamma, D. A., Friedland, G., Elizalde, B., Ni, K., Poland, D., Borth, D., and Li, L.-J. Yfcc100m: The new data in multimedia research. *Communications of the ACM* (2016), 64–73.

APPENDIX

A. PROOF OF LEMMA 1

Proof. We have $o_{ik} = 1 - e^{-\lambda_{ik} T_k}$, $C_k = \sum_{i \in F_k} o_{ik}$, and $h_k = \sum_{i \in F_k} \lambda_{ik} o_{ik}$. Since $\frac{\partial o_{ik}}{\partial T_k} \geq 0$, and $\frac{\partial^2 o_{ik}}{\partial T_k^2} \leq 0$, we have $\frac{\partial C_k}{\partial T_k} \geq 0$ and $\frac{\partial^2 C_k}{\partial T_k^2} \leq 0$. This means that C_k is a non-decreasing concave function of T_k which implies that T_k is a non-decreasing convex function of C_k, *i.e.*, $\frac{\partial T_k}{\partial C_k} \geq 0$ and $\frac{\partial^2 T_k}{\partial C_k^2} \geq 0$. Taking derivatives from the two sides of the equation $C_k = \sum_{i \in F_k} o_{ik}$ with respect to C_k we get $1 = \sum_i \frac{\partial o_{ik}}{\partial C_k}$, and taking a second derivative we get $0 = \sum_i \frac{\partial^2 o_{ik}}{\partial C_k^2}$. Moreover, from $o_{ik} = 1 - e^{-\lambda_{ik} T_k}$ we get $\frac{\partial o_{ik}}{\partial C_k} = \lambda_{ik} e^{-\lambda_{ik} T_k} \frac{\partial T_k}{\partial C_k}$ and

$$\frac{\partial^2 o_{ik}}{\partial C_k^2} = -\lambda_{ik}^2 e^{-\lambda_{ik} T_k} \Big(\frac{\partial T_k}{\partial C_k}\Big)^2 + \lambda_{ik} e^{-\lambda_{ik} T_k} \frac{\partial^2 T_k}{\partial C_k^2}.$$

The above equation implies that $\frac{\partial^2 o_{ik}}{\partial C_k^2} < 0$ if $\frac{\partial^2 T_k}{\partial C_k^2} < \lambda_{ik} \Big(\frac{\partial T_k}{\partial C_k}\Big)^2$. This means that $\frac{\partial^2 o_{ik}}{\partial C_k^2}$ is negative for larger values of λ_{ik} and is positive for smaller values of λ_{ik}. Therefore, $\sum_i \frac{\partial^2 o_{ik}}{\partial C_k^2} = 0$ implies that $\sum_i \lambda_{ik} \frac{\partial^2 o_{ik}}{\partial C_k^2} \leq 0$ and hence $\frac{\partial^2 h_k}{\partial C_k^2} \leq 0$, hence h_k is a concave function of C_k. □

B. PROOF OF THEOREM 2

Proof. Let T denote the characteristic time of the cache of size C shared by the K providers. Based on the characteristic time approximation, we know that T is the solution to

$$C = \sum_{k=1}^{K} \sum_{i \in F_k} (1 - e^{-\lambda_{ik} T}).$$

Now, partitioning the cache into K slices and allocating size $C_k = \sum_{i \in F_k} (1 - e^{-\lambda_{ik} T})$ to CP k achieves the same hit rate, and hence same utility for each CP. Therefore, the performance of sharing the cache can be obtained by partitioning it into K slices as explained above. Since $(C_1, C_2, \ldots, C_K)$ is a feasible solution, *i.e.*, falls in the solution space defined by (1), the solution to the optimization problem (1) will be as good or better. □