Counterintuitive Characteristics of Optimal Distributed LRU Caching Over Unreliable Channels

Guocong Quan[1], Jian Tan[1,2], and Atilla Eryilmaz[1]

[1]The Ohio State University, Columbus, OH 43210, USA
[1]Email: {quan.72, tan.252, eryilmaz.2}@osu.edu
[2]Machine Intelligence Technology, Alibaba Group, Sunnyvale, CA 94085, USA
[2]Email: j.tan@alibaba-inc.com

***Abstract*—Least-recently-used (LRU) caching and its variants have conventionally been used as a fundamental and critical method to ensure fast and efficient data access in computer and communication systems. Emerging data-intensive applications over unreliable channels, e.g., mobile edge computing and wireless content delivery networks, have imposed new challenges in optimizing LRU caching systems in environments prone to failures. Most existing studies focus on reliable channels, e.g., on wired Web servers and within data centers, which have already yielded good insights with successful algorithms on how to reduce cache miss ratios. Surprisingly, we show that these widely held insights do not necessarily hold true for unreliable channels.**

We consider a single-hop multi-cache distributed system with data items being dispatched by random hashing. The objective is to achieve efficient cache organization and data placement. The former allocates the total memory space to each of the involved caches. The latter decides data routing strategy and data replication scheme. Analytically we characterize the unreliable LRU caches by explicitly deriving their asymptotic miss probabilities. Based on these results, we optimize the system design.

Remarkably, these results sometimes are counterintuitive, differing from the ones obtained for reliable caches. We discover an interesting phenomenon: asymmetric cache organization is optimal even for symmetric channels. Specifically, even when cache unreliability probabilities are equal, allocating the cache spaces unequally can achieve a better performance. We also propose an explicit unequal allocation policy that outperforms the equal allocation. In addition, we prove that splitting the total cache space into separate LRU caches can achieve a lower asymptotic miss probability than resource pooling that organizes the total space in a single LRU cache.

These results provide new and even counterintuitive insights that motivate novel designs for caching systems over unreliable channels. They can potentially be exploited to further improve the system performance in real practice.

I. Introduction

Caching is a fundamental and critical method in modern computer and communication systems that can efficiently accelerate data access [1], [2], [3], [4], [5] at the expense of a dedicated fast memory space. As default algorithms, the least-recently-used (LRU) caching and its variants [6], [7], [8] have been predominantly used to manage the allocated cache space in various computer systems [3], [4], [9]. Under the LRU algorithm, only the most recently used data items are stored in the cache. If the cache is full and the requested data items cannot be found therein, the data item that has not been used for the longest time will be moved out of the cache to make room for the newly requested one. To design efficient caching systems that minimize the miss ratios, a fundamental problem is to optimize cache organization and data placement. The former decides the optimal memory allocation to the involved cache spaces and the latter dispatches data requests to the right caches.

Most of the existing work on cache optimization focus on improving the performance over reliable channels, e.g., within data centers and on wired Web servers [10], [11], [12]. These studies have yielded good insights with successful algorithms in real systems [3], [13]. However, with the increasing popularity of emerging data applications over wireless and mobile networks, e.g., mobile edge computing and content delivery networks, cached data delivered over unreliable channels become substantial in data-intensive applications [14], [15], [16], [17]. Due to mobility, fading, communication errors, etc, the access to caches could fail intermittently or be significantly delayed. Consequently, a high fetching cost can be incurred on unreliable channels even when the requested data items are indeed in the cache. This fact is significantly different from reliable channels. Thus, caching the same data item in multiple caches, i.e., data replications, should almost always be considered in presence of channel failures. All these features impose new challenges in optimizing cache organization and data placement in environments prone to failures. It merits a deeper investigation on whether the insights and engineering practices that are optimized for reliable caches can still work well in unreliable environments. If not, what to change?

We consider a single-hop multi-cache distributed system with data items being dispatched through random hashing to multiple cache spaces. One important decision is to route the data items to the right cache space. The current practice unanimously relies on an effective scale-out method that is called consistent hashing [18]. Under consistent hashing, data requests are routed to different caches according to a hash function. To be general and analytically tractable for our

This work was primarily supported by the NSF grant: CNS-NeTS-1717060, and partially supported by the NSF grants: CCSS-EARS-1444026, CNS-NeTS-1514260, CNS-NeTS-1717045, CMMI-SMOR-1562065, CNS-ICN-WEN-1719371, CNS-SpecEES-1824337 and the DTRA grants: HDTRA1-15-1-0003, HDTRA1-18-1-0050.

modeling analysis, we consider a dispatching scheme based on hashing that satisfies the Simple Uniform Hashing Assumption (SUHA). To support data replication for unreliable channels, we assume the random hash function maps each data item to a set of caches instead of a single one. Then, we optimize the data replications by carefully designing the hash functions and the cache space allocation. In particular, we make the following contributions to the literature on LRU caching:

- We propose a tractable model for LRU caching over unreliable channels, which considers cache organization and data placement simultaneously. More importantly, we derive the asymptotic miss probability in a closed form, which provides an effective tool to optimize caching system performance.
- We characterize the property of the optimal cache space allocation and the data replication. Surprisingly, the results are quite different from those for reliable channels. A counterintuitive phenomenon is discovered: allocating the cache spaces unequally is better than the equal arrangement even when the channels have identical unreliability probabilities. Moreover, we propose an explicit non-identical separation policy that outperforms the identical separation policy. These new insights deepen our understandings on LRU caching over unreliable channels and can potentially be applied in real practice to further improve the system performance.

Related work: For reliable LRU caches, the miss probability can be accurately approximated when the cache space is sufficiently large [19], [20], [21], [10], [22], [23], [24], [25]. Extensive studies have been conducted to optimize cache space allocation with different objectives. In order to maximize the utility functions based on miss probabilities, it is proved in [11] that splitting the total memory space into separate LRU caches is at least as good as resource pooling when the whole memory space is allocated to a single LRU cache. In addition, when data sizes, popularity distributions, request rates and data overlaps are considered jointly, complex results can be obtained such that cache space separation can be asymptotically better than, equal to or worse than resource pooling depending on the afore-mentioned four factors [10]. For caching over unreliable channels, most existing work focus on optimizing data placement to improve caching gains [16], [26], [27]. For example, in [28], it is shown that the ubiquitous path replication algorithm combined with LRU replacement policy is suboptimal in caching networks, and novel adaptive algorithms with optimality guarantees are proposed to decide which data item should be stored in the cache. However, few existing work consider cache space allocation and data placement simultaneously in one framework. How to allocate cache space and dispatch data requests for LRU caching over unreliable channels still remains unexplored and deserves a thorough investigation.

II. Model Description

Consider a set of infinite data items $\mathcal{D} = \{d_1, d_2, \cdots\}$ and a data flow which is a sequence of data requests on the data set $\mathcal{D}$. Assume the size of each data item is identical and normalized to 1. Assume the requests arrive according to a Poisson process. Let $\{\tau_n, -\infty < n < +\infty\}$ denote the time points that the requests arrive. Define R_n as the data item that is requested at time τ_n, $R_n \in \mathcal{D}$. Assume R_n's are i.i.d random variables and define $\mathbb{P}[R_n = d_i] = q_i$, $i \geq 1$ as the popularity distribution. Empirical studies on real data traces have shown that the popularities often follow a Zipf's distribution. Therefore, we assume

$$q_i \sim c/i^{\alpha}, \alpha > 1, i \geq 1.$$

Note $f(z) \sim g(z)$ means $\lim_{z\to\infty} f(z)/g(z) = 1$. To characterize the miss ratio of the system, it is sufficient to focus on the request at one time point saying τ_0 when the system reaches stationarity, because the requests are assumed to be independent. Consider a set of M LRU caches $\mathcal{C} = \{C_m : 1 \leq m \leq M\}$. Assume the cache C_m can be accessed with a probability p independently at time τ_n. Define a hash function $H : \mathcal{D} \to 2^{\mathcal{C}} \backslash \emptyset$ which hashes a data item to a nonempty subset of all M caches. For example, in Fig. 1, the data request d_2 is hashed to $\{C_1, C_{M-1}\}$, but only C_1 can be accessed at that time. Let N_i denote the number of elements in the set $H(d_i)$, $i \geq 1$. We assume $H(\cdot)$ is randomly selected from a set of hash functions $\mathcal{H} = \{h_w(\cdot), w = 1, 2, \cdots\}$ that satisfies Assumption 1 (i.e., SUHA property). Note that although $H(\cdot)$ is random, once a hash function $h_w(\cdot)$ is selected, each data item will be mapped to a subset of caches deterministically by $h_w(\cdot)$.

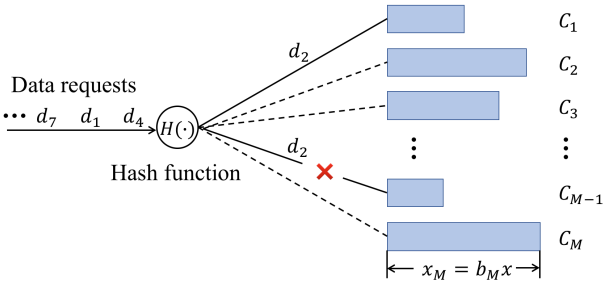

Fig. 1. Distributed caching over unreliable channels

Assumption 1 (SUHA)**.** *Assume N_i's are i.i.d. random variables with $\mathbb{P}[N_i = m] = \mu_m$, $1 \leq m \leq M$, $\sum_{m=1}^{M} \mu_m = 1$. Given N_i, assume $H(d_i)$'s are randomly chosen from all $\binom{M}{N_i}$ possible sets with an equal probability for each.*

Let $\mathcal{I}_n \triangleq H(R_n)$, $\mathcal{I}_n \subseteq \mathcal{C}$. Let $\mathcal{J}_n = \{C_m : C_m \in \mathcal{I}_n, C_m \text{ is accessible at } \tau_n, 1 \leq m \leq M\}$. When the request arrives at time τ_n, the system will fetch the data item R_n from the caches in the set $\mathcal{J}_n$. A cache hit occurs if and only if R_n is stored in at least one cache of $\mathcal{J}_n$. Otherwise, we call it a miss. Only the caches in $\mathcal{J}_n$ will be updated by the request R_n according to the LRU algorithm. We assume that the updating process can be completed as long as the cache is accessible when the request arrives. Let x denote the total cache space and $x_m = b_m x$ denote the space of cache C_m, $0 < b_m < 1$, $\sum_{m=1}^{M} b_m = 1$, $1 \leq m \leq M$. Without loss of generality, we assume the caches are sorted such that

x_m is non-increasing with m. The objective of this paper is to characterize the optimal hashing mechanism $\vec{\mu}^*$ and cache space allocation $\vec{b}^*$ under different settings.

III. Counterintuitive Insights from Analysis

The performance of reliable LRU caching ($p = 1$) has been investigated for a long time. Let $Q(x)$ denote the miss probability of a single LRU cache with a cache space x. According to Theorem 3 of [20], we have for Zipf's popularity distributions (i.e., $q_i \sim c/i^\alpha$, $\alpha > 1$),

$$Q(x) \sim \frac{\Gamma(1-1/\alpha)^\alpha}{\alpha} \frac{c}{x^{\alpha-1}}, \text{ as } x \to \infty, \tag{1}$$

where $\Gamma(1-1/\alpha) = \int_0^\infty t^{-1/\alpha} e^{-t} dt$ is the gamma function. For multiple LRU caches over reliable channels organized by the hashing mechanism described in Section II, the miss probability of the system can be accurately approximated by [25]. In Lemma 1, we show that pooling the total cache space into a single LRU cache can achieve a better asymptotic miss probability than splitting the cache space to multiple caches, when channels are reliable.

Lemma 1. *For M LRU caches organized by the hashing mechanism described in Section II, if $p = 1$, then as $x \to \infty$, we have almost surely for all H,*

$$\mathbb{P}[\textit{Miss for } R_0|H] \gtrsim Q(x),$$

where the equality holds asymptotically if and only if $b_m = 1/M$, $1 \le m \le M$.

The proof is presented in Section VII-A. Lemma 1 implies two insights for LRU caching over reliable channels:

- The asymptotic miss ratio under resource pooling is always better than or equal to that under resource separation.
- Allocating cache space unequally will achieve a worse asymptotic miss probability than allocating equally.

Interestingly, when channels are not reliable (i.e., $p < 1$), none of these insights will hold. In Section IV, we rigorously prove the following counterintuitive results for distributed LRU caching over unreliable channels, all building on Theorem 1.

- *When the cache spaces are required to be the same, splitting the total memory space into multiple LRU caches can achieve a better miss probability than resource pooling (cf. Theorem 2). We further characterize the splitting required to minimize the miss probability as a function of the channel unreliability level and the total available memory space (cf. Theorem 3).*
- *When the cache spaces are allowed to be different, allocating the total memory space unequally to the caches can achieve a better miss probability than the equal cache space allocation, even when the unreliability level is identical for each channel (cf. Theorem 4). We further develop a cache allocation policy that can yield significant improvements on the miss probability compared to the equal cache space allocation (cf. Theorem 5).*

These contradictory results for reliable and unreliable channels indicate the importance to consider channel reliability when organizing caching systems. They reveal that previously successful engineering methods for optimizing caches over reliable channels may not work well in unreliable environments. Fortunately, new insights are provided in this paper and can be potentially exploited to modify existing algorithms and further improve the system performance. Detailed proofs of our statements are provided in Section VII.

IV. Performance Analysis

In this section, we first derive the asymptotic miss probability for the distributed LRU caching over unreliable channels modeled in Section II, and then characterize the optimal cache space allocation and hashing mechanism with the objective to minimize the miss probability.

A. Miss probability

In this section, we derive accurate approximations for the cache miss probability. Given a set of M caches $\mathcal{C}$, there are $\binom{M}{m}$ different subsets that contain m caches, $1 \le m \le M$. Let $\mathcal{S}_k^{(m)}$, $1 \le k \le \binom{M}{m}$ denote the subsets that contain m caches, and $I_{k,i}^{(m)}$, $1 \le i \le m$ be the indices of the caches in set $\mathcal{S}_k^{(m)}$. Assume $I_{k,i}^{(m)}$ are sorted as an increasing sequence in i. For example, if $\mathcal{S}_k^{(3)} = \{C_9, C_4, C_{10}\}$, then $(I_{k,1}^{(3)}, I_{k,2}^{(3)}, I_{k,3}^{(3)}) = (4, 9, 10)$. Define $W_k^{(m)} \triangleq \mathbb{P}[\mathcal{I}_0 = \mathcal{S}_k^{(m)}|H] = \sum_{i \in \{i:H(d_i)=\mathcal{S}_k^{(m)}\}} q_i$. We derive the miss probability in Theorem 1.

Theorem 1. *Under the model described in Section II, as the total cache space $x \to \infty$, we have almost surely for all H,*

$$\mathbb{P}[\textit{Miss for } R_0|H] \sim \sum_{m=1}^{M} \sum_{k=1}^{\binom{M}{m}} (1-p)^m W_k^{(m)} + \left(\sum_{m=1}^{M} L\left(m, \vec{b}\right) \mu_m \right) \left(\sum_{m=1}^{M} m\mu_m \right)^{\alpha-1} Q(x),$$

and

$$\mathbb{P}[\textit{Miss for } R_0] \sim P\left(x; \vec{\mu}, \vec{b}\right) \triangleq \sum_{m=1}^{M} (1-p)^m \mu_m + \left(\sum_{m=1}^{M} L\left(m, \vec{b}\right) \mu_m \right) \left(\sum_{m=1}^{M} m\mu_m \right)^{\alpha-1} Q(x),$$

where $Q(x)$ is defined in (1) and

$$L\left(m, \vec{b}\right) = \frac{1}{\binom{M}{m}} \sum_{k=1}^{\binom{M}{m}} \sum_{(j,l):\mathcal{S}_j^{(l)} \subseteq \mathcal{S}_k^{(m)}} \left(p^l (1-p)^{m-l} \cdot \left(\sum_{i=1}^{l} (1-p)^{i-1} \left(M b_{I_{j,i}^{(l)}} \right)^\alpha \right)^{1/\alpha - 1} \right).$$

The detailed proof of Theorem 1 is presented in Section VII-B. For a given total memory space x, a hashing

mechanism $\vec{\mu}$ and a space allocation strategy $\vec{b}$, the asymptotic miss probability can be explicitly approximated by the function $P(x;\vec{\mu},\vec{b})$ defined in Theorem 1. With this effective tool, next we characterize the optimal LRU caching policy over unreliable channels. Specifically, for a given total cache space x, let $\vec{\mu}^*(x)$, $\vec{b}^*(x)$ denote the optimal hashing mechanism and the optimal cache space allocation that minimize the miss probability. We aim to characterize the asymptotic behavior of the optimal solutions, i.e., $\lim_{x\to\infty}\vec{\mu}^*(x)$ and $\lim_{x\to\infty}\vec{b}^*(x)$.

B. Equal cache space allocation

In this section, assuming the total memory space is equally allocated to each cache, i.e., $b_1 = b_2 = \cdots = b_M = 1/M$, we optimize the hashing mechanism $\vec{\mu}$ as well as the cache space allocation by determining the number of caches M.

Applying Theorem 1, we derive the asymptotic miss probability for equal cache space allocation in Corollary 1.

Corollary 1. *For $b_1 = b_2 = \cdots = b_M = 1/M$, as the total caches space $x \to \infty$, we have almost surely for all H,*

$$\mathbb{P}[\textit{Miss for } R_0] \sim P\left(x;\vec{\mu},\vec{b}\right) = \sum_{m=1}^{M}(1-p)^m\mu_m + \left(\sum_{m=1}^{M}L(m,\vec{b})\mu_m\right)\left(\sum_{m=1}^{M}m\mu_m\right)^{\alpha-1}Q(x),$$

where

$$L(m,\vec{b}) = \sum_{i=1}^{m}\binom{m}{i}p^i(1-p)^{m-i}\left(\frac{p}{1-(1-p)^i}\right)^{1-1/\alpha}.$$

Next, assuming the number of caches M is finite and fixed, we optimize the hashing mechanism by considering the following optimization problem.

$$\begin{aligned}\min_{\vec{\mu}} \quad & P\left(x;\vec{\mu},(1/M,\cdots,1/M)\right)\\ \text{subject to} \quad & \sum_{m=1}^{M}\mu_m = 1, \\ & 0 \le \mu_m \le 1, \qquad m = 1,2,\cdots,M.\end{aligned} \tag{2}$$

Let $\vec{\mu}^*(x)$ denote the optimal solution to Problem (2) for a given total cache space x. We define two policies as:

Resource Pooling (RP) Policy: Allocate the total cache space to a single LRU cache.

Equal Allocation (EA) Policy: Set $\vec{b} = (\frac{1}{M},\frac{1}{M},\cdots,\frac{1}{M})$, and $\vec{\mu} = \vec{\mu}^*(x)$.

Letting $\mathbb{P}_{\text{miss}}^{\text{RP}}$ and $\mathbb{P}_{\text{miss}}^{\text{EA}}$ denote the miss probability under the RP and EA policy, respectively, we have the following theorem.

Theorem 2. *The optimal solution to Problem (2) satisfies*

$$\lim_{x\to\infty}\mu_m^*(x) = 0 \quad \textit{for } 1 \le m \le M-1,$$
$$\lim_{x\to\infty}\mu_M^*(x) = 1,$$

under which, we have

$$\lim_{x\to\infty}\mathbb{P}_{\textit{miss}}^{\textit{RP}} = 1-p, \qquad \lim_{x\to\infty}\mathbb{P}_{\textit{miss}}^{\textit{EA}} = (1-p)^M.$$

The proof is provided in Section VII-C. For sufficiently large cache space, simply hashing the data item to all M caches can achieve the optimal miss probability. Moreover, as the total cache space goes to infinity, the miss ratio of RP and EA policies will converge to the probability that all channels fail when the request arrives. Consequently, compared to resource pooling, allocating the total cache space to multiple caches can reduce the miss probability dramatically when the cache space is large enough. This conclusion is contradictory to the results for reliable caches in Lemma 1, where the asymptotic miss probability achieved by resource pooling is always smaller than or equal to that achieved by resource separation.

In addition, the limiting miss probability of the EA policy decreases exponentially with the number of caches M. Does it indicates that a larger number of caches can always guarantee a better miss probability for a given total memory space? Let $M^*(x)$ denote the optimal number of caches for the EA policy given the total cache space x. We characterize the limiting behavior of $M^*(x)$ in the following theorem.

Theorem 3. *Assuming the data items are hashed to all M caches and $\lim_{x\to\infty} x/M = \infty$, we have, as $x \to \infty$*

$$M^*(x) \sim \frac{1-\alpha}{\log(1-p)}\log x.$$

See Section VII-D for the proof. Theorem 3, in conjunction with Theorem 2, implies that the miss probability under the EA policy tends to zero for large x, which is better than the RP policy that will always have a positive lower limit (i.e., $1-p$). When the number of caches increases, although the probability that channels fail decreases, a miss will be more likely to happen in each cache. The optimal cache number $M^*(x)$ balances this trade-off.

C. Unequal cache space allocation

In this section, we answer the following question. Given a total memory space and M caches, if the cache spaces are allowed to be unequal, can we achieve a better miss probability than allocating the total memory space equally? For LRU caching over reliable channels ($p = 1$), it is shown in Lemma 1 that $b_m = 1/M$, $1 \le m \le M$ is the optimal solution. Will this result still hold when channels are not reliable ($p < 1$)? In this section, we show that if $p < 1$, choosing b_m's unequally can further reduce the asymptotic miss probability.

For a fixed M and any given space allocation method $\vec{b}$ satisfying $b_m \neq 0$ for $\forall 1 \le m \le M$, similar to the equal allocation case (Theorem 2), the optimal hashing mechanism is $\mu_M^* = 1$, $\mu_m^* = 0$, $1 \le m \le M-1$, when the total memory space x is large enough. To minimize $P(x;\vec{\mu}^*,\vec{b})$ which is defined in Theorem 1, we formulate the following problem.

$$\begin{aligned}\min_{\vec{b}} \quad & P\left(x;(0,\cdots,0,1),\vec{b}\right)\\ \text{subject to} \quad & \sum_{m=1}^{M}b_m = 1, \\ & 0 \le b_m \le 1, \qquad m = 1,2,\cdots,M.\end{aligned} \tag{3}$$

It can be verified that Problem (3) is nonconvex. Finding the global optimum $\vec{b}^*(x)$ still remains an open problem. However, we prove that allocating cache space equally is not the optimal solution and provide an easy-to-implement policy to improve the performance of equal allocation.

Theorem 4. *For M caches with $p \in (0,1)$, the optimal solution to Problem (3) satisfies $b_i^*(x) > b_j^*(x)$ for $\forall 1 \le i < j \le M$.*

The proof is presented in Section VII-E. Theorem 4 shows a very counterintuitive result that allocating cache space unequally can achieve a better miss probability than equal allocation, even when the unreliable probability p is identical for each channel. To understand why equal separation is not the optimal, let us take a look at the optimal static caching policy. Under a static policy, the popularity of each data item is pre-known, based on which the cache space allocation and the data placement are designed. For reliable caches, the optimal static policy stores the most popular data items in one of the caches. For unreliable caches, however, the static optimal policy is nontrivial due to the potential benefits of data replications. Let x_m°, $1 \le m \le M$, denote the memory space allocated to the cache C_m under the optimal static policy. Let $b_m^\circ = x_m^\circ / \sum_{i=1}^M x_i^\circ, 1 \le m \le M$. The solution of x_m°, $1 \le m \le M$, is not unique. In the following lemma, we characterize one optimal static policy for unreliable channels.

Lemma 2 (Characteristics of an Optimal Static Policy)**.** *If the popularity of each content were known a priori, the following static policy minimizes the miss probability for M caches with a total memory space x:*

Cache space allocation: Set $x_1^\circ \ge x_2^\circ \ge \cdots \ge x_M^\circ$, satisfying

$$x_m^\circ = 0 \ \text{ or } \ x_m^\circ \ge 1, \quad 1 \le m \le M,$$
$$\frac{(1-p)^{j-1}}{(\lfloor x_j^\circ \rfloor)^\alpha} \ge \frac{(1-p)^{k-1}}{(\lfloor x_k^\circ \rfloor + 1)^\alpha} \quad \text{for } \forall x_j^\circ \ge 1, x_k^\circ \ge 1,$$
$$\sum_{m=1}^{M} x_m^\circ = x;$$

Data placement: Store data items $\{d_i : 1 \le i \le \lfloor x_m^\circ \rfloor\}$ in cache C_m, if $x_m^\circ \ge 1$.

As the cache space goes to infinity, the optimal static allocation can be explicitly calculated as

$$\lim_{x\to\infty} b_m^\circ(x) = (1-p)^{(m-1)/\alpha} \frac{1-(1-p)^{1/\alpha}}{1-(1-p)^{M/\alpha}}. \tag{4}$$

Based on Lemma 2, we summarize the following key insights, which intuitively explain why unequal allocation can achieve better miss ratios than equal allocation.

Key insights: The above optimal static policy explicitly characterizes how many caches that each item must be stored in. While the most popular items in the set $\{d_i : 1 \le i \le \lfloor x_M^\circ \rfloor\}$ are stored in all M caches, progressively less popular items are stored in decreasing number of caches, as described in Lemma 2. As such, this policy optimally balances the trade-off between the cost of storing the same item in multiple caches and the likelihood of finding a requested item in at least one of the connected caches. Despite its optimality guarantee, the optimal static policy is not directly implementable since it requires the knowledge of the popularity of each item to determine the cache allocation and data placement. However, it provides useful insights for the design of dynamic cache management policies where the popularity of the items are unknown. In particular, by allocating cache space *unequally*, as dictated by the static design, and by using LRU cache management at each of the cache, which adaptively maintains more popular requests in its cache, we can obtain a counter-intuitive design of a dynamic *unequal* caching policy over unreliable channels.

Next, we propose an unequal cache space allocation policy that significantly improves the performance of equal cache space allocation in a large range of channel unreliability levels.

Unequal Allocation (UA) Policy: Set $\vec{\mu} = (0, 0, \cdots, 1)$ and

$$\vec{b}(x) = \begin{cases} \vec{b}^\circ(x), & \text{if } p > p_{\text{th}}, \\ (1/M, \cdots, 1/M), & \text{otherwise}, \end{cases}$$

where p_{th} is the unique solution to $L(M, (\frac{1}{M}, \cdots, \frac{1}{M})) = L(M, \vec{b}^\circ(x))$. Let $\mathbb{P}_{\text{miss}}^{\text{UA}}$ denote the miss probability of the UA policy.

Note that as the total cache space goes to infinity, the miss probability under any policies including the EA, the UA and the optimal policy will all converge to the probability that no cache is accessible when the request arrives, i.e., to $(1-p)^M$. The following theorem characterizes how much faster the UA policy converges to this limit compared to the EA policy.

Theorem 5. *Let us define $\rho \triangleq \lim_{x\to\infty} \frac{\mathbb{P}_{miss}^{UA} - (1-p)^M}{\mathbb{P}_{miss}^{EA} - (1-p)^M}$, which measures how much faster the unequal cache space allocation policy converges to the limit $(1-p)^M$ compared to the equal cache space allocation policy. Then, we have*

$$\rho = \min\left\{1, \frac{L\left(M, \lim_{x\to\infty} \vec{b}^\circ(x)\right)}{L(M, (\frac{1}{M}, \cdots, \frac{1}{M}))}\right\},$$

which is strictly less than 1 if $p > p_{th}$.

The proof is provide in Section VII-F. Setting $\alpha = 1.4$, we plot ρ as a function of p for different M's in Fig. 2. It can be

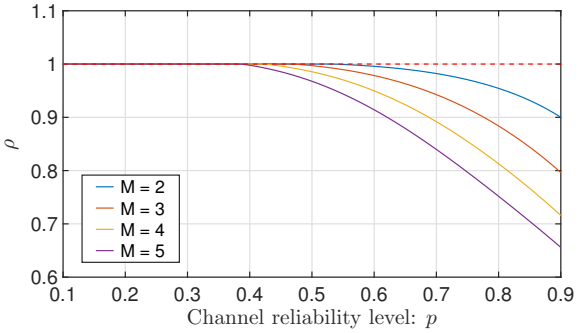

Fig. 2. Benefits of the UA policy over the EA policy

observed that in a large range of channel unreliabilities, e.g.,

$p \in (0.6, 0.9)$, ρ is much smaller than 1, which indicates that the UA policy gains considerable improvements over equal cache space allocation.

V. Experiments

To validate our theoretical analysis, we conduct 4 simulation experiments. In Experiment 1, we simulate 5 caches with general $\vec{b}$ and $\vec{\mu}$, which validates Theorem 1. In Experiments 2 and 3, we compare the equal cache space allocation with resource pooling and unequal cache space allocation, respectively. In Experiment 4, we evaluate the proposed policies using real data traces. Experiment results successfully validate the counterintuitive insights revealed by theoretical analysis.

Experiment 1. In this experiment, we validate Theorem 1 by simulating 5 caches over unreliable channels. Let $\vec{b} = (0.3, 0.2, 0.2, 0.15, 0.15)$, $\vec{\mu} = (0.1, 0.15, 0.2, 0.25, 0.3)$. Set $q_i = c/i^{1.8}$ for $1 \le i \le 10^7$, where $c = 1/\sum_{i=1}^{10^7} i^{-1.8} \approx 0.5313$. In Fig 3, we plot the miss probabilities for $p = 0.8, 0.85, 0.9$, respectively. The theoretical results approximated by Theorem 1 match very well with the empirical ones obtained by simulations, even when the total cache space is relatively small.

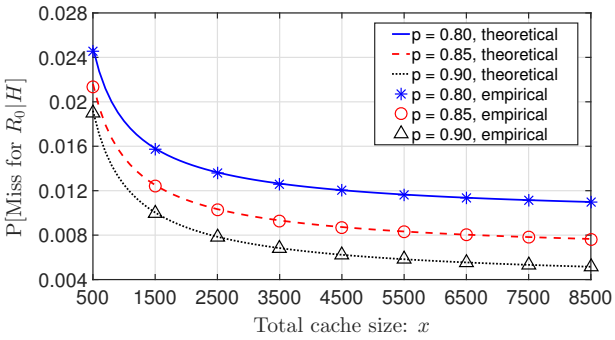

Fig. 3. Multiple unreliable caches accessed by hashing

Experiment 2. In this experiment, we compare the miss probabilities achieved by the RP policy with that achieved by the EA policy. Set $\vec{b} = (1/M, \cdots, 1/M)$, $\vec{\mu} = (0, \cdots, 0, 1)$, $p = 0.9$ and $q_i = c/i^{1.8}$ for $1 \le i \le 10^7$, where $c = 1/\sum_{i=1}^{10^7} i^{-1.8} \approx 0.5313$. In Fig. 4, we plot the miss probabilities for $M = 1, 2, 3, 4$, respectively. It can be observed that resource separation ($M \ge 2$) achieves much better miss probabilities than resource pooling ($M = 1$), which validates our statements in Theorem 2. Moreover, as what we comment in Theorem 3, allocating the total cache space to more caches may not guarantee a better miss probability (e.g., allocating the total cache space to 3 caches achieves lower miss probabilities than allocating to 4 caches when $x \in (100, 900)$). In fact, since the theoretical approximations for miss probabilities are sufficiently accurate even when the cache space is relatively small (e.g., $x = 200$), the optimal number of caches can be theoretically calculated by minimizing $L(M, (1/M, \cdots, 1/M))$ over M.

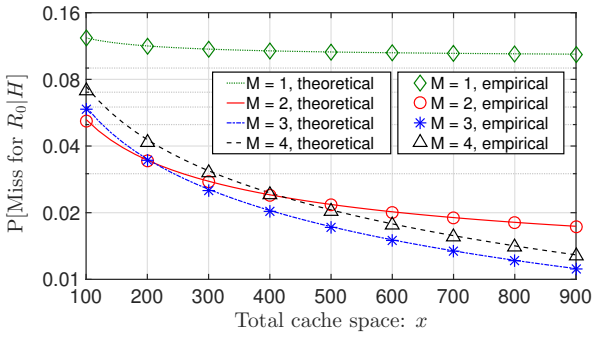

Fig. 4. Resource pooling v.s. equal cache space allocation

Experiment 3. In this experiment, we compare the EA policy with the UA policy. Let $M = 5, q_i = c/i^2$ for $1 \le i \le 10^7$, where $c = 1/\sum_{i=1}^{10^7} i^{-2} \approx 0.6079$. First, setting the total cache space $x = 500$ and applying Theorem 5, we compute ρ under different channel reliability levels and plot the results in Fig.5[left]. Then, setting $p = 0.7$, we plot the miss probabilities achieved by the EA policy and the UA policy in Fig. 5[right]. All empirical results match well with theoretical ones. It can be observed from Fig. 5[left] that when p is greater than the threshold p_{th} (≈ 0.4), ρ will be strictly less than one (i.e., the UA policy outperforms the EA policy). Furthermore, by setting $p = 0.7$, it is shown in Fig. 5[right] that the miss probabilities achieved by the UA policy (unequal cache space allocation) can be significantly smaller than that achieved by the EA policy (equal cache space allocation). For $p > 0.7$, an even larger improvement is expected.

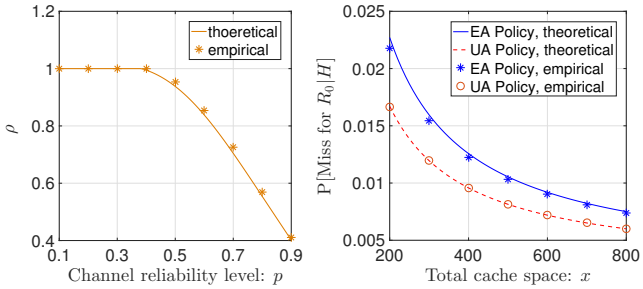

Fig. 5. Equal v.s. unequal cache space allocation

Experiment 4. In this experiment, we compare the RP, EA and UA policies using a data trace collected on a content delivery network. The trace is also used for evaluation and labeled as cdn1 in [29], [30]. Our objective is to check whether

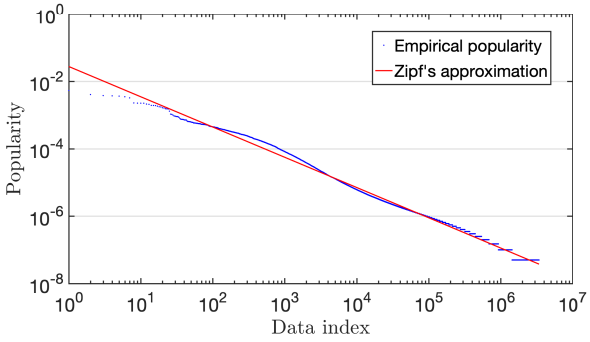

Fig. 6. Popularity of trace data

our designs perform well under popularity distributions obtained from real-world traces. For this experiment, we use 20 millions requests.[1] We plot the empirical popularities in Fig. 6, as well as the Zipf's approximation (i.e., $0.0273/i^{0.897}$, $1 \leq i \leq 3417123$). Notably, $\alpha = 0.897$ is less than 1 and therefore beyond the scope of this paper (see [22] for LRU caching with $\alpha < 1$). However, the UA policy can still be obtained from Equation (4), and the insights revealed by our theoretical analyses still hold according to the following experiment results. Setting $M = 2$, $x = 20000$, we first compare the miss ratios of the RP, EA and UA policies under different channel reliability levels, and plot the results in Fig. 7[left]. It can be observed that the UA policy

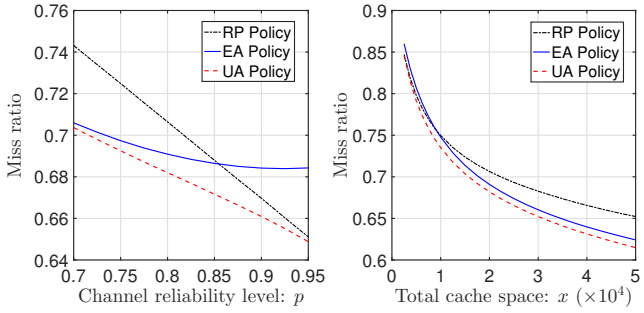

Fig. 7. Performance evaluation based on traces

always achieves the best miss ratios. Moreover, when p is relatively large (respectively small), the EA (respectively RP) policy achieves much worse performance, which validates the insights revealed by Theorem 5. Next, setting $M = 2$ and $p = 0.8$, we compare the proposed policies under different cache spaces. The results are plotted in Fig. 7[right]. Note that the UA policy outperforms the RP and EA policies in the whole range of x. In addition, when x is small, resource pooling can even achieve better miss ratios than equal cache space allocation, which verifies that a larger number of caches may not always result in a better miss ratio (see Theorem 3).

VI. Conclusion

In this work, we studied the distributed LRU caching over unreliable channels by explicitly approximating the miss probability and discovering counterintuitive insights in optimizing the cache space allocation and the data placement. Our investigation revealed two counterintuitive insights that are in stark contrast with the principles of distributed LRU caching under reliable conditions: (i) that resources-pooling is no longer optimal in the presence of channel unreliabilities, and (ii) that, even under symmetric unreliabilities, it is necessary to allocate unequal cache spaces to otherwise identical distributed caches. Our analysis framework also allowed us to develop an explicit unequal allocation policy which outperforms the equal allocation in a large range of channel unreliability levels. These insights and designs are expected to help with the development and implementation of efficient distributed caching solutions under unreliable conditions.

[1]In this experiment, the item sizes are set to be 1.

VII. Proofs

A. Proof of Lemma 1

Proof. If $p = 1$, storing the same data item in multiple caches will waste the cache space and bring no additional benefits. Therefore, the optimal hashing vector is $\vec{\mu}^* = (1, 0, \cdots, 0)$. Then, applying Theorem 2 of [25], we have

$$\mathbb{P}[\text{Miss for } R_0|H] \sim Q\left(\bar{x}\right),$$

where

$$\bar{x} = \left(\sum_{m=1}^{M} {b_m}^{1-\alpha} M^{-\alpha}\right)^{-1/(\alpha-1)} x.$$

Moreover, applying the Hoeffding's inequality, we have $\sum_{m=1}^{M} {b_m}^{1-\alpha} M^{-\alpha} \geq 1$, where the equality holds if and only if $b_m = 1/M$ for $1 \leq m \leq M$. Since $Q(x)$ is decreasing with x, we finish the proof. □

B. Proof of Theorem 1

To prove Theorem 1, we establish the following lemmas.

Lemma 3. *Conditional on $\mathcal{I}_0 = \mathcal{J}_0 = \{C_1, \cdots, C_m\}$, as the total cache space $x \to \infty$, we have*

$$\mathbb{P}[\textit{Miss for } R_0|\mathcal{I}_0 = \mathcal{J}_0 = \{C_1, \ldots, C_m\}, H] \sim Q(\bar{x})$$

where $\bar{x} = \left(\sum_{k=1}^{m} (1-p)^{k-1} {b_k}^{\alpha}\right)^{1/\alpha} x$.

Proof. For $1 \leq k \leq M$, let $\tau_{-\sigma_k}$ denote the last time before τ_0 when the data item R_0 was requested and the cache C_k was accessible then. Define

$$T_k(n) = \sum_{i \geq 1} \mathbf{1}\left(\cup_{j=1}^{n} \{\{R_{-j} = d_i\} \cap \{C_k \in \mathcal{I}_{-j}\}\}\right),$$

which is the number of distinct data requests that successfully access cache C_k between time τ_{-n} and time τ_{-1}. We also define the inverse of $T_k(n)$ as $T_k^{\leftarrow}(x) = \min\{n : T_k(n) \geq x\}$. It can be verified that

$$\{\text{Miss for } R_0|\mathcal{J}_0 = \mathcal{I}_0, H\} \Leftrightarrow \cap_{k=1}^{m} \{\sigma_k > T_k^{\leftarrow}(b_k x)\}.$$

The rest of the proof is consisted of two steps. In Step 1, we will estimate $\mathbb{P}[\cap_{k=1}^{m} \{\sigma_k > n_k\}]$. In Step 2, we show that $\mathbb{P}[\cap_{k=1}^{m} \{\sigma_k > T_k^{\leftarrow}(b_k x)\}]$ can be approximated by $\mathbb{P}[\cap_{k=1}^{m} \{\sigma_k > \bar{T}^{\leftarrow}(b_k x)\}]$, where

$$\bar{T}^{\leftarrow}(x) = \Gamma(1 - 1/\alpha)^{-\alpha} c^{-1} p^{-1} x^{\alpha}. \tag{5}$$

For $n_1 > n_2 > \cdots > n_m$ and $\forall d_i \in \mathcal{D}$, we have

$$\begin{aligned}
&\mathbb{P}[\cap_{k=1}^{m} \{\sigma_k > n_k\} | R_0 = d_i] \\
&= \mathbb{P}[\cap_{k=1}^{m} \{\sigma_k > n_m\} | R_0 = d_i] \\
&\qquad \cdot \mathbb{P}[\cap_{k=1}^{m-1} \{\sigma_k > n_k - n_m\} | R_0 = d_i] \\
&= \mathbb{P}[\cap_{k=1}^{m} \{\sigma_k > n_m\} | R_0 = d_i] \\
&\qquad \cdot \Pi_{j=1}^{m-1} \mathbb{P}[\cap_{k=1}^{j} \{\sigma_k > n_j - n_{j+1}\} | R_0 = d_i],
\end{aligned} \tag{6}$$

where the second equality holds because the requests arrive according to a Poisson process. For $\forall n \geq 1$, let $Y_i(n) = \sum_{j=1}^{n} \mathbf{1}(\{R_{-j} = d_i\})$, $1 \leq i \leq N$. We obtain, as $n \to \infty$,

$$\begin{aligned}
&\mathbb{P}[\sigma_1 > n, \sigma_2 > n, \cdots, \sigma_m > n | R_0 = d_i] \\
&= \sum_{k=0}^{n} \mathbb{P}[Y_i(n) = k] \mathbb{P}[\sigma_1 > n, \cdots, \sigma_m > n | Y_i(n) = k] \\
&= \sum_{k=0}^{n} \mathbb{P}[Y_i(n) = k](1-p)^{mk} \\
&= \mathbb{E}\left[\exp(n \log(1-p) Y_i(n))\right] = (1 - q_i + q_i(1-p)^m)^n \\
&\sim \exp\left(-q_i(1-(1-p)^m)n\right). \qquad (7)
\end{aligned}$$

For $\forall c_1 > c_2 > \cdots > c_{m-1}$ and $n_k = c_k n_m$, $1 \leq k \leq m-1$, as $n_m \to \infty$, we have, by combining (6) and (7) and defining $n_{m+1} = 0$,

$$\begin{aligned}
&\mathbb{P}[\sigma_1 > n_1, \sigma_2 > n_2, \cdots, \sigma_m > n_m] \\
&= \sum_{i=1}^{\infty} q_i \mathbb{P}[\sigma_1 > n_1, \sigma_2 > n_2, \cdots, \sigma_m > n_m | R_0 = d_i] \\
&= \sum_{i=1}^{\infty} q_i \Pi_{k=1}^{m} \left(1 - q_i + q_i(1-p)^k\right)^{n_k - n_{k+1}} \\
&\sim \sum_{i=1}^{\infty} q_i \exp\left(-q_i \sum_{k=1}^{m} (1-(1-p)^k)(n_k - n_{k+1})\right) \\
&= \sum_{i=1}^{\infty} q_i \exp\left(-q_i p \sum_{k=1}^{m} (1-p)^{k-1} n_k\right) \\
&\sim \frac{c^{1/\alpha}\Gamma(2-1/\alpha)}{\alpha - 1} \left(p \sum_{k=1}^{m} (1-p)^{k-1} n_k\right)^{-1+1/\alpha},
\end{aligned}$$

implying

$$\begin{aligned}
&\mathbb{P}[\text{Miss for } R_0 | \mathcal{J}_0 = \mathcal{C}, H] \\
&\sim \frac{c^{1/\alpha}\Gamma(2-1/\alpha)}{\alpha - 1} \left(p \sum_{k=1}^{m} (1-p)^{k-1} T_k^{\leftarrow}(b_k x)\right)^{-1+1/\alpha}.
\end{aligned}$$

Next, we will show that $\mathbb{P}[\text{Miss for } R_0 | \mathcal{J}_0 = \mathcal{C}, H]$ can be accurately approximated by replacing $T_k^{\leftarrow}(b_k x)$ with a constant $\bar{T}^{\leftarrow}(b_k x)$. Define

$$\bar{T}(n) \triangleq \sum_{i=1}^{N} \left(1 - (1 - q_i + q_i(1-p))^n\right).$$

We have, as $n \to \infty$

$$\bar{T}(n) \sim \sum_{i=1}^{N} (1 - \exp(-q_i p n)) \sim \Gamma\left(1 - \frac{1}{\alpha}\right) c^{1/\alpha} p^{1/\alpha} n^{1/\alpha},$$

and therefore, $\bar{T}^{\leftarrow}(x) = \Gamma(1-1/\alpha)^{-\alpha} c^{-1} p^{-1} x^{\alpha}$. Applying Lemma 7.1 in [10], we can prove that for $\forall \epsilon > 0$,

$$\begin{aligned}
&\mathbb{P}\left[T_k^{\leftarrow}(x) < \bar{T}^{\leftarrow}\left(\frac{x}{1+\epsilon}\right)\right] \leq \exp\left(-\frac{\epsilon^2 x}{4(1+\epsilon)}\right) \\
&\mathbb{P}\left[T_k^{\leftarrow}(x) > \bar{T}^{\leftarrow}\left(\frac{x}{1-\epsilon}\right)\right] \leq \exp\left(-\frac{\epsilon^2 x}{4(1-\epsilon)}\right).
\end{aligned}$$

Therefore, for $\forall \epsilon > 0$, there exists an x_0 such that, for $x \geq x_0$

$$\begin{aligned}
&\mathbb{P}[\text{Miss for } R_0 | \mathcal{I}_0 = \mathcal{J}_0 = \{C_1, \cdots, C_m\}, H] \\
&\leq \mathbb{P}[\sigma_1 > \bar{T}^{\leftarrow}(b_1 x/(1+\epsilon)), \cdots, \sigma_m > \bar{T}^{\leftarrow}(b_m x/(1+\epsilon))] \\
&\quad + \sum_{k=1}^{m} \mathbb{P}[\bar{T}^{\leftarrow}(b_k x/(1+\epsilon))] \\
&\leq Q\left(\left(\sum_{k=1}^{m} (1-p)^{k-1} {b_k}^{\alpha}\right)^{1/\alpha} \frac{x}{1+\epsilon}\right) \\
&\quad + \sum_{k=1}^{m} \exp\left(-\frac{\epsilon^2 b_k x}{4(1+\epsilon)}\right) \\
&= Q\left(\bar{x}/(1+\epsilon)\right) + o(Q(\bar{x})), \qquad (8)
\end{aligned}$$

where $\bar{x} = \left(\sum_{k=1}^{m} (1-p)^{k-1} {b_k}^{\alpha}\right)^{1/\alpha} x$. Similarly, we can prove for large x

$$\begin{aligned}
&\mathbb{P}[\text{Miss for } R_0 | \mathcal{I}_0 = \mathcal{J}_0 = \{C_1, \cdots, C_m\}, H] \\
&\geq Q\left(\bar{x}/(1-\epsilon)\right) - o(Q(\bar{x})). \qquad (9)
\end{aligned}$$

Combining (8) and (9) then letting $x \to \infty$ finish the proof. □

Lemma 4. *Data items that are hashed to the cache set $\mathcal{S}_k^{(m)}$ asymptotically follow a Zipf's distribution $c{\mu_m}^{\alpha} \Big/ \left(i W_k^{(m)} \binom{M}{m}\right)^{\alpha}, i \geq 1$ almost surely for all H.*

Proof. Let $X_{i,k}^{(m)} = 1$ indicate that $H(d_i) = \mathcal{S}_k^{(m)}$. Otherwise, $X_{i,k}^{(m)} = 0$. Define $I_{n,k}^{(m)} = \sum_{i=1}^{n} X_{i,k}^{(m)}$. Applying the Bernstein's inequality, we can prove that for $\forall 0 < \epsilon < 1$, there exists constants c and n_1 such that for $\forall n > n_1$,

$$\begin{aligned}
&\mathbb{P}\left[\left|\frac{I_{n,k}^{(m)}}{n\mu_m / \binom{M}{m}} - 1\right| > \epsilon\right] \leq \exp\left(-\frac{n^2 {\mu_m}^2 \epsilon^2 / \binom{M}{m}^2}{2n\mu_m / \binom{M}{m} + 2\epsilon/3}\right) \\
&\leq \exp\left(-cn^{1/2}\right).
\end{aligned}$$

Therefore, applying the union bound, we have

$$\begin{aligned}
&\mathbb{P}\left[\bigcap_{i \geq n} \left\{\left|\frac{I_{i,k}^{(m)}}{n\mu_m / \binom{M}{m}} - 1\right| \leq \epsilon\right\}\right] \\
&\geq 1 - \sum_{i \geq n} \exp\left(-ci^{1/2}\right) \geq 1 - \int_{n-1}^{\infty} \exp\left(-ct^{1/2}\right) dt. \\
&= 1 - \frac{2}{c^2}(c(n-1)^{1/2} - 1)\exp(-c(n-1)).
\end{aligned}$$

Since $\sum_{n=1}^{\infty} c^2 (c(n-1)^{1/2} - 1)/2 \exp(-c(n-1)) < \infty$, by the Borel-Contelli lemma, for any ϵ, there always exists an integer n_1 such that for $\forall n > n_1$, $\cap_{i \geq n} \left\{\left|I_{i,k}^{(m)} \binom{M}{m} / (i\mu_m) - 1\right| \leq \epsilon\right\}$ holds almost surely for any H.

Let $q_{i,k}^{(m)}$ denote q_j with the index $j = I_{i,k}^{(m)}$. We have almost surely for all H, $q_{i,k}^{(m)} \sim c \Big/ \left(i \binom{M}{m} \Big/ \mu_m\right)^{\alpha}$. Then normalizing $q_{i,k}^{(m)}$ by $\mathbb{P}[\mathcal{I}_0 = \mathcal{S}_k^{(m)} | H] = W_k^{(m)}$ finishes the proof. □

Lemma 5 (Theorem 4.1 of [10])**.** *Consider K flows of independent data requests sharing a LRU cache. For each flow k, $1 \le k \le K$, the data popularity follows a Zipf's distribution c_k/i^α, $\alpha > 1$, $i \ge 1$ asymptotically. Assume the size of each data item is 1. Let μ_k denote the probability that a request is from flow k, then as the cache space x goes to infinity, we have*

$$\mathbb{P}[\textit{Miss for } R_0 | R_0 \textit{ is from flow } k] \sim Q\left(\frac{(c_k\mu_k)^{1/\alpha}x}{\sum_{i=1}^K (c_i\mu_i)^{1/\alpha}}\right).$$

With the established lemmas, now we prove Theorem 1.

Proof of Theorem 1. Under the model described in Section II, we have,

$$\begin{aligned}
&\mathbb{P}[\text{Miss for } R_0|H] \\
&= \sum_{m=1}^{M}\sum_{k=1}^{\binom{M}{m}} \mathbb{P}[\mathcal{I}_0 = \mathcal{S}_k^{(m)}]\mathbb{P}[\text{Miss for } R_0|\mathcal{I}_0 = \mathcal{S}_k^{(m)}, H] \\
&= \sum_{m=1}^{M}\sum_{k=1}^{\binom{M}{m}} \mathbb{P}[\mathcal{I}_0 = \mathcal{S}_k^{(m)}]\Big((1-p)^m \\
&\quad + \sum_{(i,j):\mathcal{S}_i^{(j)} \subseteq \mathcal{S}_k^{(m)}} \Big(\mathbb{P}[\mathcal{J}_0 = \mathcal{S}_i^{(j)} | \mathcal{I}_0 = \mathcal{S}_k^{(m)}, H] \\
&\qquad \cdot \mathbb{P}[\text{Miss for } R_0 | \mathcal{J}_0 = \mathcal{S}_i^{(j)}, \mathcal{I}_0 = \mathcal{S}_k^{(m)}, H]\Big)\Big). \quad (10)
\end{aligned}$$

Combining Lemma 3, Lemma 4 and Lemma 5 yields

$$\mathbb{P}[\text{Miss for } R_0 | \mathcal{J}_0 = \mathcal{S}_i^{(j)}, \mathcal{I}_0 = \mathcal{S}_k^{(m)}, H] \sim Q\left(\bar{x}_i^{(j)}\right) \quad (11)$$

where

$$\bar{x}_i^{(j)} = \left(\sum_{l=1}^{j} (1-p)^{l-1} {b_{I_{i,l}^{(j)}}}^\alpha\right)^{1/\alpha} \frac{M\mu_m x}{\binom{M}{m}\sum_{i=1}^M i\mu_i}.$$

Plugging $\mathbb{P}[\mathcal{I}_0 = \mathcal{S}_k^{(m)}|H] = 1/W_k^{(m)}$ and (11) into (10) finishes the proof. □

C. Proof of Theorem 2

Proof. Recall that

$$\begin{aligned}
P\left(x;\vec{\mu},\vec{b}\right) &= \sum_{m=1}^{M}(1-p)^m\mu_m \\
&\quad + \left(\sum_{m=1}^{M} L(m,\vec{b})\mu_m\right)\left(\sum_{m=1}^{M} m\mu_m\right)^{\alpha-1} Q(x),
\end{aligned}$$

where

$$L(m,\vec{b}) = \sum_{i=1}^{m}\binom{m}{i} p^i(1-p)^{m-i}\left(\frac{p}{1-(1-p)^i}\right)^{1-1/\alpha}.$$

Since $\lim_{x\to\infty} Q(x) = 0$, we have

$$\lim_{x\to\infty} P\left(x;\vec{\mu},\vec{b}\right) = \sum_{m=1}^{M}(1-p)^m\mu_m,$$

which implies

$$\begin{aligned}
\lim_{x\to\infty}\mu_m^*(x) &= 0 \quad \text{for } 1 \le m \le M-1, \\
\lim_{x\to\infty}\mu_M^*(x) &= 1,
\end{aligned}$$

and

$$\lim_{x\to\infty}\mathbb{P}_{\text{miss}}^{\text{RP}} = 1-p, \qquad \lim_{x\to\infty}\mathbb{P}_{\text{miss}}^{\text{EA}} = (1-p)^M.$$

□

D. Proof of Theorem 3

Proof. Assuming the data items are hashed to all caches, we have $\mathcal{I}_0 = \mathcal{C}$, which is no longer a random variable. Therefore, Theorem 1 as well as Corollary 1 still holds even when M scales with x as long as $\lim_{x\to\infty} x/M = \infty$. Corollary 1 yields $\mathbb{P}[\text{Miss for } R_0] \sim P\left(x;\vec{\mu},\vec{b}\right) = (1-p)^M + L(M,\vec{b})M^{\alpha-1}Q(x)$ where

$$L(M,\vec{b}) = \sum_{i=1}^{M}\binom{M}{i} p^i(1-p)^{M-i}\left(\frac{p}{1-(1-p)^i}\right)^{1-1/\alpha}.$$

Letting $\partial P(x;\vec{\mu},\vec{b})/\partial M = 0$ and using $\lim_{M\to\infty} L(M,\vec{b}) = p^{1-1/\alpha}$ finish the proof. □

E. Proof of Theorem 4

Proof. Suppose towards contradictions that there exists $1 \le i < j \le M$ such that $b_i^* = b_j^*$. Then, b_i^* and b_j^* should be the optimal solution to the following problem

$$\begin{aligned}
\min_{b_i,b_j} \quad & P(x;\vec{\mu},\vec{b}) \\
\text{subject to} \quad & \sum_{m=1}^{M} b_m = 1, \quad (12) \\
& b_m = b_m^*, \qquad 1 \le m \le M, m \ne i,j, \\
& 0 \le b_m \le 1, \qquad 1 \le m \le M.
\end{aligned}$$

The necessary condition for the optimal solution is

$$\partial P(x;\vec{\mu},\vec{b})/\partial b_i\Big|_{b_i=b_i^*} = \partial P(x;\vec{\mu},\vec{b})/\partial b_j\Big|_{b_j=b_j^*}.$$

It is easy to verify that $b_i = b_j = (b_i^* + b_j^*)/2$ does not satisfy this condition and therefore is not the optimal solution. Furthermore, if we assume $b_i^* < b_j^*$, then simply switching these two values can achieve a lower function value. Therefore, we have $b_i^* > b_j^*$ for $\forall 1 \le i < j \le M$, which finishes the proof. □

F. Proof of Theorem 5

Proof. Recalling the EA and UA policies and Theorem 1, we have

$$\begin{aligned}
&\mathbb{P}_{\text{miss}}^{\text{EA}} \sim (1-p)^M + L\left(M, \left(\frac{1}{M},\cdots,\frac{1}{M}\right)\right)M^{\alpha-1}Q(x), \\
&\mathbb{P}_{\text{miss}}^{\text{UA}} \sim (1-p)^M + L\left(M,\vec{b}^\circ(x)\right)M^{\alpha-1}Q(x) \ \text{ for } p > p_{\text{th}}, \\
&\mathbb{P}_{\text{miss}}^{\text{UA}} = \mathbb{P}_{\text{miss}}^{\text{EA}} \ \text{ for } p \le p_{\text{th}}.
\end{aligned}$$

Through direct computation, we can prove Theorem 5. □

REFERENCES

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.*, vol. 1. IEEE, 1999, pp. 126–134.

[2] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's highly available key-value store," in *Proceedings of Twenty-first ACM SIGOPS Symposium on Operating Systems Principles*, ser. SOSP '07. ACM, 2007, pp. 205–220.

[3] "Memcached," http://memcached.org/.

[4] "Redis," http://redis.io/.

[5] "Aerospike," http://www.aerospike.com/.

[6] N. Megiddo and D. S. Modha, "ARC: A self-tuning, low overhead replacement cache." in *FAST*, vol. 3, no. 2003, 2003, pp. 115–130.

[7] S. Jiang and X. Zhang, "LIRS: an efficient low inter-reference recency set replacement policy to improve buffer cache performance," *ACM SIGMETRICS Performance Evaluation Review*, vol. 30, no. 1, pp. 31–42, 2002.

[8] A. S. Tanenbaum, *Modern Operating Systems*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2001.

[9] "MySQL," https://www.mysql.com/.

[10] J. Tan, G. Quan, K. Ji, and N. Shroff, "On resource pooling and separation for LRU caching," in *Proceedings of the 2018 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*. ACM, 2018.

[11] W. Chu, M. Dehghan, D. Towsley, and Z.-L. Zhang, "On allocating cache resources to content providers," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. ACM, 2016, pp. 154–159.

[12] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 1, pp. 53–64, Jun. 2012.

[13] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab *et al.*, "Scaling Memcache at Facebook." in *nsdi*, vol. 13, 2013, pp. 385–398.

[14] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, "Fog computing: Focusing on mobile users at the edge," *arXiv preprint arXiv:1502.01815*, 2015.

[15] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017.

[16] K. Shanmugam, N. Golrezaei, A. G. Dimakis, A. F. Molisch, and G. Caire, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[17] X. Wang, M. Chen, T. Taleb, A. Ksentini, and V. Leung, "Cache in the air: exploiting content caching and delivery techniques for 5G systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 131–139, 2014.

[18] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web," in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*. ACM, 1997, pp. 654–663.

[19] R. Fagin, "Asymptotic miss ratios over independent references," *Journal of Computer and System Sciences*, vol. 14, no. 2, pp. 222–250, 1977.

[20] P. R. Jelenković, "Asymptotic approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities," *The Annals of Applied Probability*, no. 2, pp. 430–464, 1999.

[21] P. R. Jelenković and X. Kang, "LRU caching with moderately heavy request distributions," in *2007 Proceedings of the Fourth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*. SIAM, 2007, pp. 212–222.

[22] G. Quan, K. Ji, and J. Tan, "LRU caching with dependent competing requests," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications (INFOCOM 2018)*, Honolulu, USA, Apr. 2018.

[23] R. Hirade and T. Osogami, "Analysis of page replacement policies in the fluid limit," *Operations research*, vol. 58, no. 4-part-1, pp. 971–984, 2010.

[24] N. Gast and B. Van Houdt, "Transient and steady-state regime of a family of list-based cache replacement algorithms," *ACM SIGMETRICS Performance Evaluation Review*, vol. 43, no. 1, pp. 123–136, 2015.

[25] K. Ji, G. Quan, and J. Tan, "Asymptotic miss ratio of LRU caching with consistent hashing," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications (INFOCOM 2018)*, Honolulu, USA, Apr. 2018.

[26] J. Li, T. K. Phan, W. Chai, D. Tuncer, G. Pavlou, D. Griffin, and M. Rio, "Dr-cache: Distributed resilient caching with latency guarantees," in *IEEE INFOCOM*. IEEE, 2018.

[27] M. Dehghan, A. Seetharam, B. Jiang, T. He, T. Salonidis, J. Kurose, D. Towsley, and R. Sitaraman, "On the complexity of optimal routing and content caching in heterogeneous networks," in *IEEE INFOCOM*. IEEE, 2015, pp. 936–944.

[28] S. Ioannidis and E. Yeh, "Adaptive caching networks with optimality guarantees," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 1. ACM, 2016, pp. 113–124.

[29] D. S. Berger, R. K. Sitaraman, and M. Harchol-Balter, "Adaptsize: Orchestrating the hot object memory cache in a content delivery network." in *NSDI*, 2017, pp. 483–498.

[30] D. S. Berger, N. Beckmann, and M. Harchol-Balter, "Practical bounds on optimal caching with variable object sizes," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 2, p. 32, 2018.