Caching Policies over Unreliable Channels

Paulo Sena* , Igor Carvalho*, Antonio Abelem*, György Dán‡, Daniel Menasche§, Don Towsley¶
*Dept. of Computer Science, UFPA, Belém, Brazil, ‡Dept. of Computer Science, EECS, KTH, Stockholm, Sweden,
§Dept. of Computer Science, UFRJ, Rio de Janeiro, Brazil, ¶School of Information and Computer Sciences, UMass, Amherst

***Abstract*—Recently, there has been substantial progress in the formal understanding of how caching resources should be allocated when multiple caches each deploy the common LRU policy. Nonetheless, the role played by caching policies beyond LRU in a networked setting where content may be replicated across multiple caches and where channels are unreliable is still poorly understood. In this paper, we investigate this issue by first analyzing the cache miss rate in a system with two caches of unit size each, for the LRU, and the LFU caching policies, and their combination. Our analytical results show that joint use of the two policies outperforms LRU, while LFU outperforms all these policies whenever resource pooling is not optimal. We provide empirical results with larger caches to show that simple alternative policies, such as LFU, provide superior performance compared to LRU even if the space allocation is not fine tuned. We envision that fine tuning the cache space used by such policies may lead to promising additional gains.**

***Index Terms*—caching, networking, wireless**

I. Introduction

Caching is a fundamental technique for scaling the service capacity of networked systems. By bringing content closer to users, caches reduce service latency for end users and decrease network congestion and load at content servers, also known as content custodians. Users can access the caches either through reliable wired channels or through unreliable wireless channels, and a single user may be able to retrieve content from one or several caches, depending on the system topology and design choices.

Owing to to the importance of caching, a number of policies have been proposed in the past, with its own advantages. For example, the Least Recently Used (LRU) policy promotes file recency and optimizes performance under adversarial finite request sequences [1, Theorem 3.6]. Similarly, the Least Frequently Used (LFU) policy maximizes the hit ratio under the independent reference model (IRM) by promoting the contents with the highest request frequency [1, Table 3.1], while Time-To-Live (TTL) policies are policies that use timers to adjust the hit probability of each content, as well as being able to approximate other policies such as LRU and FIFO [2].

Each caching policy naturally allocates space to each of the flows arriving to the cache. We refer to such an allocation as *intra-cache* space partitioning, which may be dynamic, such as in LRU, LFU and TTL, or static, such as in Segmented LRU (SLRU) [3]. In a networking context, we refer to the space allocated to each cache as the *inter-cache* space partition. In wireless networks with unreliable channels, in particular, partitioning space across multiple caches can build robustness into the system, as the availability of each of the caches may vary over time [4].

The performance of a caching system relies on a number of factors, including (a) the policies used by the caches for content insertion and eviction, (b) the division of cache space among different contents or flows inside each cache and across caches, as well as by (c) networking aspects, such as network congestion and reliability. These factors have been individually analyzed in previous work, but their interplay is non-trivial and not yet well understood. In particular, the role of caching policies and cache partitioning over unreliable channels has not been previously investigated.

In this paper, we evaluate the role of caching policies and cache partitioning over unreliable channels, and find that caching policies and cache partitioning are intrinsically related, and that their impacts on the performance of caching systems over unreliable channels must be jointly taken into account. The study of the LRU cache replacement policy, which is often used in practice due to its low complexity and robustness, over unreliable channels, is an important first step towards understanding the interplay between networking and caching [4]. Nonetheless, we show that if system designers have the flexibility to choose the caching policy, other cache replacement policies may be preferable.

The rest of the paper is organized as follows. Section II outlines general rules for cache pooling vs. partitioning, discussing the relationship between partitioning and replacement policies. Section III introduces assumptions and provides analytical results for cache partitioning over unreliable channels. Numerical evaluation is reported in Section IV. Section V provides a discussion of the design space and related work, and Section VI concludes the paper.

II. Terminology and background

We consider a caching system which provides *caching as a service* to its clients, such as Amazon ElastiCache [5]. Given a caching budget, measured in bytes, the clients decide how this budget will be partitioned across caches installed in *physically distinct servers*, giving rise to the so called inter-cache space partitioning. In particular, the caching budget can be *pooled together in a single server or partitioned across multiple servers*. In addition, the space in each physical cache is also logically partitioned among multiple contending flows. The intra-cache space is governed by *local caching policies*. As for the physical allocation of resources, the *logical allocation* must determine logical pooling or partitioning of space.

TABLE I
INTRA-CACHE SPACE PARTITIONING STRATEGIES

Policy	Criterion for segmentation	Partitioning
ARC [7]	probatory vs protected	dynamic
SARC [8]	sequential vs random	dynamic
SLRU	probatory vs protected	static
LRU-K [1], [9]	meta-data vs data	static
memcached [10], [11]	object sizes	static
LFUDA [12], [13]	object sizes	dynamic

TABLE II
TABLE OF NOTATION

Variable	Description
M	Number of caches
N	Number of contents in catalog
x	Total cache space
b_i	Fraction of space allocated to cache i
p	Channel success probability
q	Popularity of most popular content (Section III)

A. Why cache space partitioning?

We start with briefly outlining general rules of thumb for deciding between pooling and partitioning of cache resources. Those rules of thumb are mostly derived from [6], and comprise four known characteristics favoring resource partitioning.

Time varying rates of requests for contents: it is better to partition cache resources if content requests are not time varying. Otherwise, partitioning may lead to resource underutilization when request flows pass through low demand periods and are directed to caches exclusively designated for them.

Sizes of objects vary across flows: a miss for a large sized object may lead to the eviction of many small sized objects. For that reason, when object sizes are heterogeneous, it may be beneficial to partition cache space for large size objects so that they do not cause churn among small sized ones.

Small overlap of content requests across flows: if flows are roughly independent, the benefits of resource pooling may be smaller [2, Theorem 2].

Miss rate per flow is a concern: if some flows need to be protected from others in terms of the experienced miss rate, resource partitioning is beneficial.

B. Intra-cache vs inter-cache resource partitioning

A number of classic caching policies, such as ARC, SARC and LFUDA, already partition cache space based on request characteristics (see Table I). We refer to those policies as intra-cache space partition policies. Inter-cache space partition, in contrast, aims at partitioning space across physical distinct caches [4]. Combining insights from these two areas may lead to novel joint intra-cache and inter-cache space partitioning, and could be an interesting subject for future work. *In this paper, we consider the evaluation of lightweight intra-cache space partition policies together with static inter-cache space partition policies.*

C. Which features to use for partitioning cache space?

An essential aspect to determining how to partition space across caches is to establish which features to use for the partitioning. If request flows are already divided into clusters, with little overlap, a natural choice is to allocate a separate cache to each of the flows. Alternatively, flows may exhibit different characteristics, e.g., some flows may correspond to sequential accesses, and others to random accesses. If flows are not tagged with information about their characteristics, a challenge is to detect them in an online fashion. Sub-flows inside a flow may additionally correspond to requests for objects with different properties, such as content sizes, further motivating additional partitions.

D. When to adjust partitioning? Static vs dynamic allocation

Space allocation may be adjusted in a static fashion, e.g., through parameters that determine the amount of space used by each cache segment [3]. Alternatively, dynamic space allocation is typically implemented in a non-parametric fashion, with space being adjusted according to the evolution of request patterns, e.g., as implemented by SARC or LFUDA. In this work we focus on static inter-cache space allocation, and leave dynamic strategies as subject for future work.

III. ANALYTICAL MODEL

In this section, we develop an analytical model of the cache miss probability for LRU and LFU caching, under three main assumptions. *(A1) Zero download delay (ZDD):* the time taken by users to download a content after a hit, as well as the the time to retrieve the content from a custodian after a miss, are both negligible; *(A2) Broadcast channel:* each request is sent to all caches through an unreliable broadcast channel. The probability that a request issued from a user reaches each of the caches equals p; *(A3) Independent reference model (IRM):* at each slot a single request is issued according to the IRM model [14].

In what follows we consider a simple setting comprising of two ($M = 2$) unit size caches, i.e., the total cache size is $B = 2$ and a fraction $b_i = 0.5$ of the cache space is allocated to cache i, $i \in \{1, 2\}$. There are two ($N = 2$) contents in the catalog, A and B. Requests are made for contents A and B with probabilities q and $1-q$, respectively. Each unreliable link produces a successful transmission with probability p, independent of the other. Although the setting is admittedly simple, it allows us to appreciate some of the issues involved in the choice between policies, and in the choice of policy whether or ont cache storage resources are pooled or partitioned (see Figure 1).

Let us first consider pooling, i.e., there is a single cache storing both contents, and denote by $\mathbb{P}^P_{miss}$ the cache miss probability under cache pooling. Clearly,

$$\mathbb{P}^P_{miss} = 1 - p, \tag{1}$$

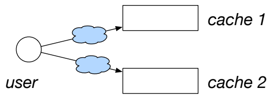


Fig. 1. System setup with $M = 2$ caches and $b_1 = b_2 = 0.5$.

as a miss only occurs if the transmission is not succesful. Otherwise, since the two contents are always stored in the cache, the request produces a cache hit.

Next, we consider resource partitioning. Let $\mathbb{P}_{miss}^{LFU,LFU}$, $\mathbb{P}_{miss}^{LRU,LRU}$ and $\mathbb{P}_{miss}^{LRU,LFU}$ be the miss probabilities when the caches adopt (LFU, LFU), (LRU, LRU) and (LRU, LFU), respectively. Furthermore, we denote by $\mathbb{P}_{miss}^{D}$ the miss probability when the two caches choose to maximize diversity, i.e., one cache stores content A and the other stores content B.

In addition, we approximate $\mathbb{P}_{miss}^{LRU,LRU}$ by $\tilde{\mathbb{P}}_{miss}^{LRU,LRU}$, which is obtained under the assumption that the two caches are independent. One of our goals with this approximation is to determine when and if the dependence between caches may improve system performance, i.e., whether $\mathbb{P}_{miss}^{LRU,LRU} < \tilde{\mathbb{P}}_{miss}^{LRU,LRU}$ could hold. Then, assuming $q \geq 0.5$,

$$\mathbb{P}_{miss}^{LFU,LFU} = 1 - q + q(1-p)^2 = 1 - 2pq + p^2 q \quad (2)$$

$$\tilde{\mathbb{P}}_{miss}^{LRU,LRU} = (1-q)((1-p(1-q))^2) + q(1-pq)^2 \quad (3)$$

$$\mathbb{P}_{miss}^{LRU,LFU} = (1-q)(1-p(1-q)) + q(1-pq)(1-p) \quad (4)$$

The rationale behind the above equations is as follows. Consider an LFU system (eq. (2)). *Under LFU, we assume that at steady state the most popular content will be statically stored in both caches.* In particular, under LFU we assume that a request for less popular content not in the cache does not result in an eviction. The relationship between LFU and static policies is further discussed in [1, Theorem 3.8] and [15]. Then, in steady state a miss occurs if the user requests the least popular content, which occurs with probability $1-q$, or if the user requests the most popular content but the request fails to reach both caches. Consider now an LRU system with two independent caches (eq. (3)). A miss occurs if (i) a request is issued towards the least (resp., most) popular content and (ii) it is not the case that the request reaches a cache wherein the content is stored. Event (ii) occurs with probability $(1-p(1-q))^2$ (resp., $(1-pq)^2$). The rationale behind (4) is similar.

For diversity maximizing caches we have

$$\mathbb{P}_{miss}^{D} = (1-q)(1-p) + q(1-p) = 1-p, \quad (5)$$

as if each cache stores a different content, with probability $(1-p)^2$ we are unable to access any of the caches, and with probabilities $(1-p)pq$ and $(1-p)p(1-q)$ we are able to access exactly one of the caches but produce a miss. Observe that we thus obtain the same miss rate as under pooling,

$$\mathbb{P}_{miss}^{D} = \mathbb{P}_{miss}^{P}. \quad (6)$$

Next, we account for dependencies between caches, which is relevant under the (LRU,LRU) setup, as they do not play a role in the other setups. To that aim, we model the caches as a Markov chain (MC) with states (A,A), (A,B) and (B,B), shown in Figure 2, with

$$p_{00}(r) = r + (1-r)(1-p)^2, \quad (7)$$

$$p_{11}(r) = (1-r)p^2, \quad (8)$$

$$p_{01}(r) = (1-r)2p(1-p), \quad (9)$$

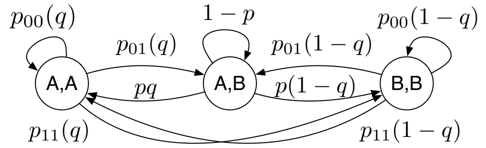


Fig. 2. Markov chain for the LRU/LRU system

obtaining the steady-state probabilities

$$\pi_{AA} = (2-p)^{-1}(p + 2q - 2pq)q \quad (10)$$

$$\pi_{AB} = (2-p)^{-1}(1 + pq - q - p)4q \quad (11)$$

$$\pi_{BB} = (2-p)^{-1}(-p - 4q + 3pq - 2pq^2 + 2q^2 + 2). \quad (12)$$

Thus, the cache miss probability in the (LRU,LRU) setup, accounting for dependencies, is given by

$$\begin{aligned}\mathbb{P}_{miss}^{LRU,LRU} &= \pi_{AA}((1-q) + q(1-p)^2) + \\ &+ \pi_{AB}(1-p) + \pi_{BB}(q + (1-q)(1-p)^2) = \quad (13)\\ &= \frac{(4q^2 - 4q + 1)p^3}{p-2} + \\ &+ \frac{(-10q^2 + 10q - 4)p^2 + (8q^2 - 8q + 5)p - 2}{p-2} \quad (14)\end{aligned}$$

Let R denote the number of requests that reach a cache. Then,

$$\mathbb{P}(R > 0) = 1 - (1-p)^2 \quad (15)$$

and

$$\begin{aligned}\mathbb{P}_{miss}^{LRU,LRU} &= 1 - \mathbb{P}(R>0)\left(1 - \mathbb{P}_{miss|R>0}^{LRU,LRU}\right) \quad (16)\\ &= 1 - (1-(1-p)^2)\left(1 - \frac{\left(4p^2 - 10p + 8\right)q(1-q)}{(2-p)^2}\right) \quad (17)\end{aligned}$$

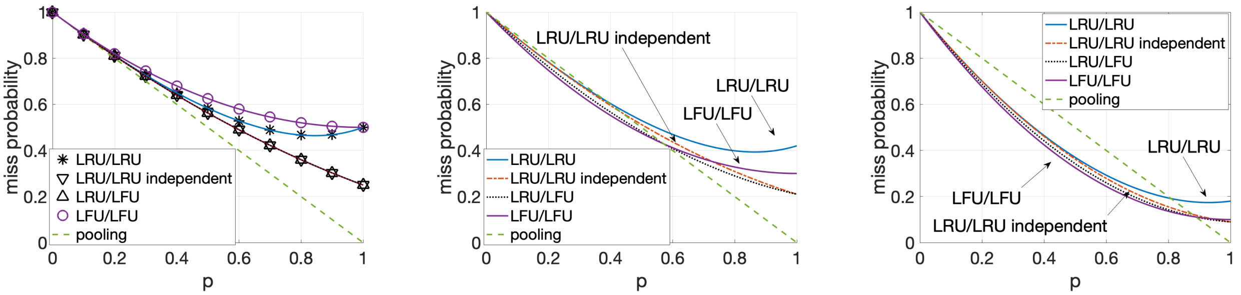
where $\mathbb{P}_{miss|R>0}^{LRU,LRU}$ is the probability of a miss conditional on that at least one request reaches a cache. While $\mathbb{P}(R>0)$ is a strictly increasing function of p, $1 - \mathbb{P}_{miss|R>0}^{LRU,LRU}$ is concave and decreasing with respect to p for $p > 2/3$, for any $q < 1$. The latter occurs because for $p > 2/3$ a reduction in channel reliability (p) favors more diversity between contents across caches [16], hence larger conditional hit probabilities ($1 - \mathbb{P}_{miss|R>0}^{LRU,LRU}$). Together, the joint effect of p on $\mathbb{P}(R>0)$ and $\mathbb{P}_{miss|R>0}^{LRU,LRU}$ may cause the miss probability to increase as p grows (see Section IV).

The error due to assuming independence is given by

$$\Delta_{miss} = \mathbb{P}_{miss}^{LRU,LRU} - \tilde{\mathbb{P}}_{miss}^{LRU,LRU} = \frac{p^3 q(1-q)}{2-p}. \quad (18)$$

Proposition 1 (Independence is beneficial)**.** *The miss probability of a system with two LRU caches is greater than the miss probability of the corresponding system wherein caches are assumed to be independent, i.e.,* $\Delta_{miss} \geq 0$. *In addition, the error in the miss probability estimate due to assuming independence of two LRU caches is bounded,* $\Delta_{miss} \leq 0.25$.

Proof. It follows from (18) that $0 \leq \Delta_{miss} \leq 0.25$ for $0 \leq p \leq 1$, and that Δ_{miss} is a monotonically increasing



(a) $q = 0.5$ (b) $q = 0.7$ (c) $q = 0.9$

Fig. 3. Simple scenario with two caches of size 1 each. If $q = 1$ (not shown above), the miss probability equals $(1-p)^2$ under cache partitioning.

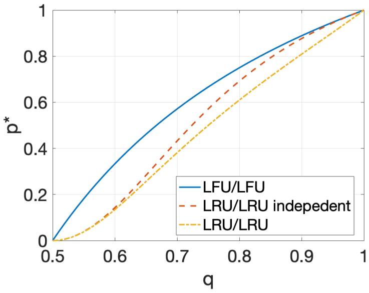


Fig. 4. Threshold to switch from partitioning to pooling: if $p < p^\star$ use partitioning; otherwise, pooling.

convex function of p, concave decreasing in q, and it attains its maximum for $p = 1$ and $q = 0.5$ (c.f., Figure 3(a)). □

Thus, interestingly, the miss rate under the independence assumption is never larger than when the two caches are dependent. This surprising result can be explained by that the probability that two independent LRU caches contain different contents is $2q(1-q)$, which is larger than π_{AB} for $p > 0$.

A. Independent LRU Caching by Design

Motivated by Proposition 1, in what follows we propose a scheme similar to multi-LRU [17]–[19] to partition LRU caches in which caches are independent by design. To achieve independence of the caches, we ensure that at most one of the caches is updated upon each request, chosen at random. In the proposed design the sender sends a bit together with the request, equal to 0 (resp., 1) if the first (resp., second) cache is to be updated in response to the request. The state of the cache for which the bit is 0 does not change in response to the request. The bit is set to 0 or 1 with probability $1/2$.

In order to analyze the cache miss rate for the proposed randomized solution, observe that the state space and the state transitions of the resulting Markov chain (MC) are as shown in Figure 2, with transition probabilities

$$\tilde{p}_{00}(r) = r + (1-r)((1-p)^2 + p(1-p)), \tag{19}$$

$$\tilde{p}_{11}(r) = 0, \tag{20}$$

$$\tilde{p}_{01}(r) = (1-r)(p(1-p) + p^2). \tag{21}$$

This MC has stationary distribution

$$\tilde{\pi}_{AA} = q^2, \quad \tilde{\pi}_{AB} = 2q(1-q), \quad \tilde{\pi}_{BB} = (1-q)^2. \tag{22}$$

We denote by $\hat{\mathbb{P}}^{LRU,LRU}_{miss}$ the cache miss probability. Then,

$$\begin{aligned}\hat{\mathbb{P}}^{LRU,LRU}_{miss} &= \tilde{\pi}_{AA}((1-q) + q(1-p)^2) + \\ &+ \tilde{\pi}_{BB}(q + (1-q)(1-p)^2) + \tilde{\pi}_{AB}(1-p),\end{aligned} \tag{23}$$

which equals (3), $\hat{\mathbb{P}}^{LRU,LRU}_{miss} = \tilde{\mathbb{P}}^{LRU,LRU}_{miss}$, in agreement with the fact that the two LRU caches are independent.

B. When is Pooling Better?

In order to further delve into the benefit of resource pooling vs. partitioning, we now analyze the optimality of pooling as a function of channel reliability.

Proposition 2 (Thresholds)**.** *For each of the caching policies (LRU,LRU), (LRU,LFU), (LFU,LFU) and independent (LRU,LRU) there is a unique threshold $p^\star$ such that if $p > p^\star$ pooling outperforms partitioning. The thresholds for $q > 0.5$ are*

$$p^\star_{LRU} = \frac{3 - 10q(1-q) - \sqrt{-28q^4 + 56q^3 - 32q^2 + 4q + 1}}{2(2q-1)^2}, \tag{24}$$

$$p^\star_{LFU} = p^\star_{LRU,LFU} = (2q-1)/q, \tag{25}$$

$$\tilde{p}^\star_{LRU} = (2q-1)^2/(3q^2 - 3q + 1). \tag{26}$$

Proof. We first prove the result for (LRU.LRU). The cache miss rate $\mathbb{P}^{LRU,LRU}_{miss}$ is a convex function of p. In addition, $\mathbb{P}^{LRU,LRU}_{miss} = 1$ for $p = 0$, $\mathbb{P}^{LRU,LRU}_{miss} \geq 0$ for $p = 1$ and for $q > 0.5$, we have $\frac{\partial \mathbb{P}^{LRU,LRU}_{miss}}{\partial p}\Big|_{p=0} = -4q^2 + 4q - 2 < -1$. Thus, for $q > 0.5$ there exists a unique threshold $p^\star_{LRU} > 0$ below which partitioning is optimal for LRU, and above which pooling is optimal. The unique threshold $p^\star_{LRU}$ is the solution to $\mathbb{P}^{LRU,LRU}_{miss} = 1 - p$, which is given by (24).

The expressions for the cache miss rates $\mathbb{P}^{LFU,LFU}_{miss}$, $\mathbb{P}^{LRU,LFU}_{miss}$ and $\tilde{\mathbb{P}}^{LRU,LRU}$ behave similarly, and the thresholds $p^\star_{LFU}$, $p^\star_{LRU,LFU}$ and $\tilde{p}^\star_{LRU}$ are obtained solving the corresponding equations, which yield (25) and (26). This concludes the proof. □

Figure 4 shows the pooling thresholds based on the above equations and shows two interesting properties. First, the pooling threshold is highest for LFU. Second, the pooling threshold of the independent (LRU,LRU) is close to that of (LRU,LRU) when contents are nearly equally popular, but it approaches that of (LFU,LFU) when content popularity is very skewed.

C. *Comparison of Caching Policies*

As shown in Figure 3(a), LFU may be the worst policy over unreliable channels in the case of resource partitioning. This is in contrast to classical results for single caches over reliable channels, for which it is well known that LFU is the stationary optimal policy [1, Theorem 3.8].

Proposition 3 (LFU may be suboptimal)**.** *Under resource partitioning, two LFU caches can be worse than two LRU caches.*

Proof. See Figure 3(a). We observed that LFU remains the worst policy given partitioned caches under a number of other scenarios, e.g., when $(p, q) \in [0.5, 0.55] \times [0.5, 0.55]$. □

Although in the case of resource partitioning LFU may be the worst option under certain conditions, as we show next, choosing the better between LFU with pooling and LFU with resource partitioning minimizes the cache miss probability.

Proposition 4 (LFU and optimal partitioning suffices)**.** *The cache miss probabilities satisfy*

$$\min\left(\mathbb{P}^{P}_{miss}, \mathbb{P}^{LFU/LFU}_{miss}\right) \leq \\ \leq \mathbb{P}^{LRU,LFU}_{miss} \leq \tilde{\mathbb{P}}^{LRU,LRU}_{miss} \leq \mathbb{P}^{LRU,LRU}_{miss}. \qquad (27)$$

Proof. To show (27), note that for $q \geq 0.5$

$$\tilde{\mathbb{P}}^{LRU,LRU}_{miss} - \mathbb{P}^{LRU,LFU}_{miss} = p(p-1)(q-1)(2q-1) \geq 0. \quad (28)$$

Combining the above result with (18), it remains to show that

$$\min\left(\mathbb{P}^{P}_{miss}, \mathbb{P}^{LFU,LFU}_{miss}\right) \leq \mathbb{P}^{LRU,LFU}_{miss}. \qquad (29)$$

The inequality above follows from $\mathbb{P}^{LRU,LFU}_{miss} - \mathbb{P}^{LFU,LFU}_{miss} = p(pq - 2q + 1)(q - 1)$, which is strictly positive if and only if $0 < p < p^{\star}_{LFU} = p^{\star}_{LRU,LFU}$. □

IV. Numerical Results

In what follows we provide numerical results obtained through simulation, so as to validate the analytical results for two caches (Section III), and to investigate whether they extend to more cache space, more contents, and many caches. We also aim at providing insight into the sensitivity of LFU to cache partitioning.[1]

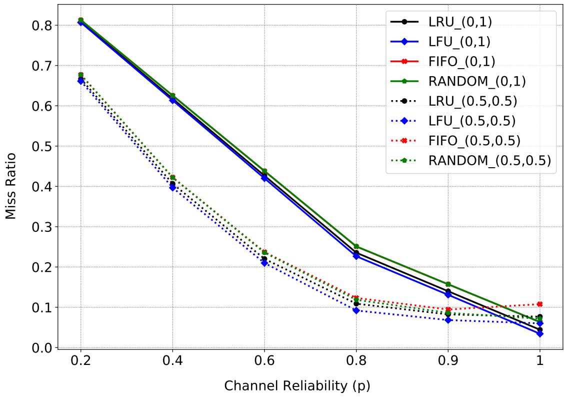


Fig. 5. Miss ratio vs. channel reliability p for LFU and LRU caching policies, with pooling (0,1) and with partitioning (0.5,0.5), $x = 50$.

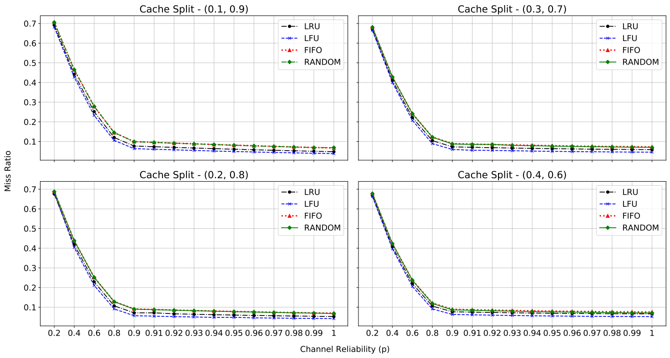


Fig. 6. Miss ratio vs. channel reliability p for LFU, LRU and FIFO and Random, $M = 2$, $x = 50$.

A. *Simulation Methodology*

Our evaluation considers that there are M caches in the system, the total cache size x varies from 10 to 8500, and a fraction b_i of the total cache size is allocated to cache $1 \leq i \leq M$. The content catalog includes 10 million data items, and the content popularity distribution follows Zipf's law with exponent $\alpha = 1.8$. Each content has unit size. The request stream is sent to the caches using a broadcast channel, i.e., each request is sent to all caches. We considered channel reliability values $p \in \{0.2, 0.4, 0.6, 0.8, 1\}$, where p = 1 corresponds to a reliable channel. Initially all caches are assumed to be empty, noting that such initial condition may impact system performance [14], [20].

B. *The Case of Two Caches*

We start with validating the observations made based on the analytical model for a system with $M = 2$ caches and equal partitions $b_i = 0.5$, compared with pooling. Figure 5 shows the cache miss ratio as a function of the channel reliability for the LFU and the LRU caching policies, with and without pooling. The figure confirms all the observations made based on the analytical model, i.e., there is a threshold p^* for LRU and for LFU above which pooling is optimal (Proposition 2), and LFU or pooling achieve a lower miss ratio than LRU (Proposition 4).

[1] Source code available at https://tinyurl.com/unrcache

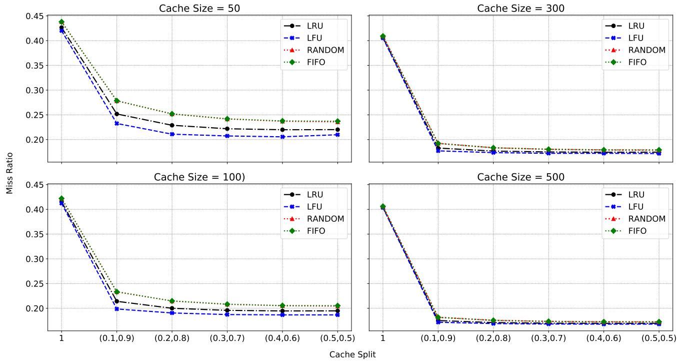


Fig. 7. Miss ratio for various cache partitioning schemes for LFU, LRU, and FIFO and Random caching. $M = 1$ and $M = 2$ caches, channel reliability $p = 0.6$, various cache sizes.

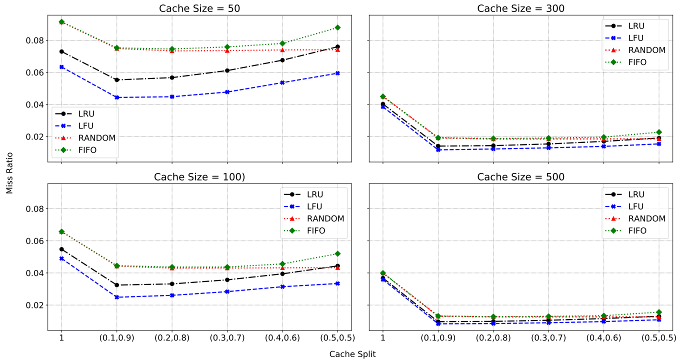


Fig. 8. Miss ratio for various cache partitioning schemes for LFU, LRU, FIFO and Random caching. $M = 1$ and $M = 2$ caches, channel reliability $p = 0.97$, various cache sizes.

In order to evaluate the potential impact of the cache partitioning on the results, Figure 6 shows the cache miss ratio as a function of the channel reliability for various unequal partitions of the cache space. The figure shows that LFU outperforms LRU for all considered cache partitioning schemes. The figure also shows that all four caching policies, including FIFO and Random, not considered in the analytical model, consistently perform close to each other. Furthermore, comparing with the results for pooling shown in Figure 5, we can conclude that the channel reliability threshold above which pooling is optimal exists in the case of unequal partitioning as well.

C. *Small Sensitivity of LFU to Cache Partitioning*

Before turning to more than $M = 2$ caches, we further investigate what happens in the case of unequal cache space allocation. For this we consider $M = 2$ caches and investigate how the optimal way of partitioning the cache space depends on the algorithm used (LFU, LRU, FIFO, Random). Recall that, for LRU, partitioning the cache has an impact on the miss ratio, and there is an optimal partitioning that depends on the content popularity distribution and the channel reliability [4]. Figure 7 shows the cache miss ratio for various cache partitioning schemes, from $(0, 1)$ (i.e., pooling), to $(0.5, 0.5)$ (i.e., equal partitions) for a channel reliability of $p = 0.6$. The subfigures show results for caches size $x \in \{50, 100, 300, 500\}$. The figure allows us to make two important observations. First, it confirms that LFU always outperforms LRU, independent of the cache partitioning scheme. Second, it shows that for the LFU caching policy the cache partitioning has rather little effect on the cache miss ratio.

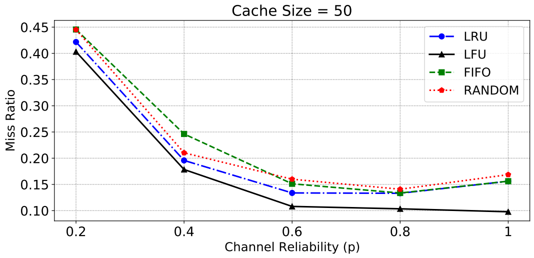


Fig. 9. Miss ratio vs. channel reliability p for LFU, LRU, TTL, FIFO and RANDOM caching policies, $M = 5$, $x = 50$.

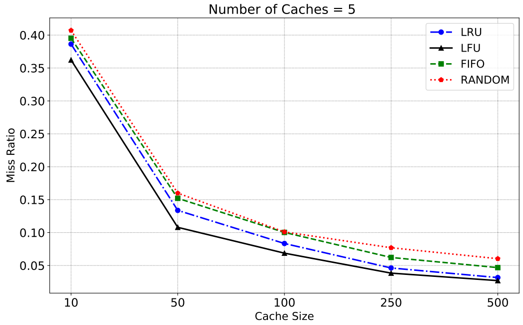


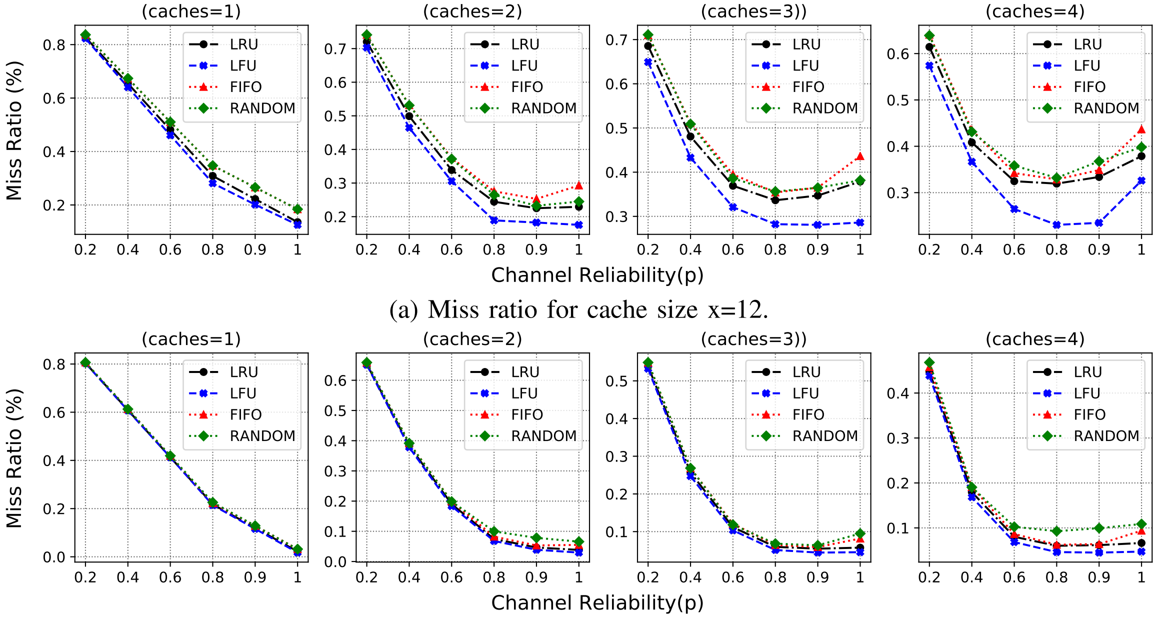
Fig. 10. Miss ratio vs. cache size x for LFU, LRU, TTL and Random caching, $M = 5$, channel reliability $p = 0.6$.

Figure 8 shows corresponding results for a channel reliability of $p = 0.97$ for the same cache sizes for cache partitioning scheme (b_1, b_2). The figure confirms that both observations hold in the case of a highly reliable channel, i.e., LFU outperforms LRU irrespective of the cache partitioning scheme, and it is less sensitive to the cache partitioning.

D. *Multiple Caches*

Figure 9 shows the cache miss ratio as a function of the channel reliability p for the caching policies LRU, LFU, FIFO and Random for a cache size of $x = 50$ and $M = 5$ caches. The figure shows that LFU outperforms all other policies in terms of miss ratio for all channel reliability values. In addition, it shows that increasing the channel reliability can increase the miss rate for LRU. Figure 10 shows the cache miss ratio for various caches sizes, for $p = 0.6$ for the same caching policies. The results are consistent, and show no impact of the cache size on the relative miss rates of the policies.

To further investigate the impact of cache partitioning, Figure 11(a) shows the cache miss ratio as a function of the channel reliability p for the LRU, LFU, FIFO and Random policies, for $M \in \{1, 2, 3, 4\}$ caches with a total size of $x = 12$. The figure shows that LFU outperforms the other



(a) Miss ratio for cache size x=12.

(b) Miss ratio for cache size x=120.

Fig. 11. Miss ratio vs. channel reliability p for a cache size of (a) 12 and (b) 120, for LFU, LRU, and TTL, 1 to 4 caches.

policies irrespective of the number of caches and the channel reliability, with a significant gain. It is also noteworthy to observe that the cache miss ratio is a convex function for all scenarios, similar to the analytical results, and is not monotonically decreasing in p for $M = 3$ and $M = 4$ caches.

Figure 11(b) show corresponding results for a total cache size of $x = 120$. The figure allows us to draw similar conclusions, even though the difference in terms of miss rates is almost negligible, due to the low aggregate popularity of the tail of the content popularity distribution. We can conclude that LFU works significantly better than LRU, FIFO and Random when cache storage is scarce, regardless of the number of partitions. In addition, LFU seems to be less sensitive to the cache partitioning than LRU. We believe that these results reinforce the adoption of LFU for unreliable channels and stationary workloads more than LRU, FIFO or Random.

V. Related work and discussion

There is a vast literature on caching systems [1], [21], [22], accounting for utility maximization [2], caching networks [23], [24] and cache replication over unreliable channels [4].

The literature on cache partitioning is mostly comprised of works focusing on its practical aspects [10] and on partitioning among multiple flows [2], [25]. Recently, there has been significant progress on the formal analysis of strategies for cache partitioning [4], [6], [26]. While the former works indicated clear advantages of cache partitioning explored by policies such as TTL, ARC, SLRU and SARC, the latter is focused on LRU systems. The purpose of this work is to bridge that gap, showing preliminary empirical results about how inter-cache partitioning impact performance under policies beyond LRU.

Caching over wireless networks has been considered under the framework of femtocaching [27], wherein base stations are equipped with caching capabilities. Alternatively, set-top boxes equipped with caches close to users can be accessed through wifi, being managed partially by the network providers [28]. In both cases, the allocation and partitioning of cache space among caches strategically placed close to users, as considered in this paper, are key elements.

A. Cache partitioning

There has been significant effort to formally derive optimality conditions for pooling and partitioning of cache resources, under the assumption of LRU caches. Much less work has been conducted on alternative policies, such as ARC, SLRU and LRU-K. Nonetheless, all those policies intrinsically account for caching resource partitioning inside each cache. For practical purposes, we envision that a better understanding of the role of pooling and partitioning, as well as of unreliable channels, accounting for those alternative policies, is key for effective deployment of caching systems.

Among the open questions related to this rich problem space, we point out the following: *(i)* assuming a system with legacy caches, some of which implementing LRU, to what extent resource partitioning can compensate for the sub-optimality of the cache policies? *(ii)* assuming a fully tunable system, wherein both caching policies and caching resource partitioning can be configured, how to configure caching policies and resource partitioning strategies? *(iii)* accounting for the fact that certain policies, such as ARC and SLRU, intrinsically account for resource partitioning at the intra-cache level, how to optimally tune the intra-cache space and the inter-cache space in a joint fashion so as to minimize the overall system wide cache miss? *(iv)* how to account for distinct costs of cache misses for different contents?

B. Cache networks and redundant caches

Modern cache systems are essentially interconnected cache networks. Therefore, the problem of caching in its entirety includes decisions about cache dimensioning, content placement, content routing, server placement, dimensioning the links and the ability to serve the cache. Following [4], in this paper we considered a single user submitting requests to a set of redundant caches through an unreliable network. We envision a number of possible extensions, including *(i) user interaction:* interaction between users can be considered in a setup wherein multiple users share a given network to access a set of caches; *(ii) broadcast channel:* we consider requests broadcasted through wireless channels. In practice, channel capacity also plays a role, requiring adjustments to the network model.

In this work we neglected the service capacity of caches and channel delays. Nonetheless, issuing multiple parallel requests to multiple caches is a natural strategy to deal with variable delays. Optimally determining load balancing strategies is out of the scope of this paper, but has been thoroughly investigated in the context of redundant queueing systems [29]. In particular, one of the key challenges involved in the analysis of such redundant systems involves accounting for the dependencies that naturally exist across multiple servers. Therefore, the design of redundant caching systems whose caches are independent by design may be beneficial for the system, as indicated in Section III-A, and to simplify its analysis.

VI. Conclusion

Caching is a simple strategy to significantly improve the performance of networked systems. At virtually zero cost, caching solutions can be tuned to reduce delays and the load at custodians. For those reasons, it is no surprise that a significant effort has been devoted to better understand how caching policies and cache allocation impact miss ratios, both under reliable and unreliable network settings.

In this paper we argued that although there has been substantial progress in the understanding of how caching resources should be allocated, there is still a vast space to explore and room for improvement. In particular, although most of the previous analytical work on cache allocation has focused on LRU, alternative policies may outperform LRU even if the space allocation is not fine tuned. We envision that fine tuning the cache space used by such policies may lead to additional gains, but leave the topic as subject for future work. In addition, we also envision the joint intra and inter-cache space segmentation as a fruitful direction for future research.

Acknowledgment: The work was supported in part by CAPES, CNPq, FAPERJ, by the NSF under grant CNS-1617437, and by Sweden's Innovation Agency (Vinnova) through the TECoSA project.

References

[1] G. Paschos, G. Iosifidis, and G. Caire, "Cache optimization models and algorithms," *arXiv preprint arXiv:1912.12339*, 2019.

[2] M. Dehghan, W. Chu, P. Nain, D. Towsley, and Z.-L. Zhang, "Sharing cache resources among content providers: A utility-based approach," *IEEE/ACM Trans. on Networking*, vol. 27, no. 2, pp. 477–490, 2019.

[3] H. Gao and C. Wilkerson, "A dueling segmented LRU replacement algorithm with adaptive bypassing," in *JWAC*, 2010.

[4] G. Quan, J. Tan, and A. Eryilmaz, "Counterintuitive characteristics of optimal distributed LRU caching over unreliable channels," in *Proc. of IEEE INFOCOM*, 2019, pp. 694–702.

[5] D. Carra, G. Neglia, and P. Michiardi, "Elastic provisioning of cloud caches: A cost-aware TTL approach," *IEEE/ACM Transactions on Networking*, 2020.

[6] J. Tan, G. Quan, K. Ji, and N. Shroff, "On resource pooling and separation for LRU caching," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, pp. 1–31, 2018.

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

[8] B. S. Gill and D. S. Modha, "SARC: Sequential prefetching in adaptive replacement cache." in *Proc. of USENIX ATC*, 2005, pp. 293–308.

[9] E. J. O'Neil, P. E. O'Neil, and G. Weikum, "An optimality proof of the LRU-K page replacement algorithm," *Journal of the ACM*, vol. 46, no. 1, pp. 92–112, 1999.

[10] M. Perham, "Slabs, pages, chunks and memcached," 2009, https://www.mikeperham.com/2009/06/22/slabs-pages-chunks-and-memcached/.

[11] M. Arlitt, R. Friedrich, and T. Jin, "Performance evaluation of web proxy cache replacement policies," *Performance Evaluation*, vol. 39, no. 1-4, pp. 149–164, 2000.

[12] Q. Yang and H. H. Zhang, "Web-log mining for predictive web caching," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 4, pp. 1050–1053, 2003.

[13] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, "Evaluating content management techniques for web proxy caches," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, pp. 3–11, 2000.

[14] E. J. Rosensweig, D. S. Menasche, and J. Kurose, "On the steady-state of cache networks," in *INFOCOM*. IEEE, 2013, pp. 863–871.

[15] Z. Liu, P. Nain, N. Niclausse, and D. Towsley, "Static caching of web servers," in *Multimedia Computing and Networking 1998*, vol. 3310. International Society for Optics and Photonics, 1997, pp. 179–190.

[16] W. Caarls, E. Hargreaves, and D. S. Menasché, "Q-caching: an integrated reinforcement-learning approach for caching and routing in information-centric networks," *arXiv preprint arXiv:1512.08469*, 2015.

[17] E. Leonardi and G. Neglia, "Implicit coordination of caches in small cell networks under unknown popularity profiles," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1276–1285, 2018.

[18] A. Giovanidis and A. Avranas, "Spatial multi-lru caching for wireless networks with coverage overlaps," *ACM SIGMETRICS Performance Evaluation Review*, vol. 44, no. 1, pp. 403–405, 2016.

[19] K. Avrachenkov, J. Goseling, and B. Serbetci, "A low-complexity approach to distributed cooperative caching with geographic constraints," *POMACS*, vol. 1, no. 1, pp. 1–25, 2017.

[20] J. Li, S. Shakkottai, J. C. Lui, and V. Subramanian, "Accurate learning or fast mixing? dynamic adaptability of caching algorithms," *IEEE JSAC*, vol. 36, no. 6, pp. 1314–1330, 2018.

[21] N. Gast and B. Van Houdt, "TTL approximations of the cache replacement algorithms LRU(m) and h-LRU," *Performance Evaluation*, vol. 117, pp. 33–57, 2017.

[22] M. A. Maddah-Ali and U. Niesen, "Coding for caching: Fundamental limits and practical challenges," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 23–29, 2016.

[23] V. Pacifici and G. Dán, "Coordinated selfish distributed caching for peering content-centric networks," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3690–3701, 2016.

[24] M. Dehghan, A. Seetharam, B. Jiang, T. He *et al.*, "On the complexity of optimal routing and content caching in heterogeneous networks," in *INFOCOM*, 2015, pp. 936–944.

[25] A. Araldo, G. Dán, and D. Rossi, "Caching encrypted content via stochastic cache partitioning," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 548–561, Feb. 2018.

[26] K. Ji, G. Quan, and J. Tan, "Asymptotic miss ratio of LRU caching with consistent hashing," in *Proc. of IEEE INFOCOM*, 2018, pp. 450–458.

[27] K. Shanmugam, N. Golrezaei, Dimakis *et al.*, "Femtocaching: Wireless content delivery through distributed caching helpers," *IEEE Trans. on Information Theory*, vol. 59, no. 12, pp. 8402–8413, 2013.

[28] K. Suh, C. Diot, J. Kurose, L. Massoulie, C. Neumann, D. Towsley, and M. Varvello, "Push-to-peer video-on-demand system: Design and evaluation," *IEEE JSAC*, vol. 25, no. 9, pp. 1706–1716, 2007.

[29] G. Joshi, "Synergy via redundancy: Boosting service capacity with adaptive replication," *ACM SIGMETRICS PER*, vol. 45, no. 3, 2018.