Performance analysis methods for list-based caches with non-uniform access

Giuliano Casale, Nicolas Gast

Performance analysis methods for list-based caches with non-uniform access

Giuliano Casale, Nicolas Gast

***Abstract*—List-based caches can offer lower miss rates than single-list caches, but their analysis is challenging due to state space explosion. In this setting, we propose novel methods to analyze performance for a general class of list-based caches with tree structure, non-uniform access to items and lists, and random or first-in first-out replacement policies. Even though the underlying Markov process is shown to admit a product-form solution, this is difficult to exploit for large caches. Thus, we develop novel approximations for cache performance metrics, in particular by means of a singular perturbation method and a refined mean field approximation. We compare the accuracy of these approaches to simulations, finding that our new methods rapidly converge to the equilibrium distribution as the number of items and the cache capacity grow in a fixed ratio. We find that they are much more accurate than fixed point methods similar to prior work, with mean average errors typically below 1.5% even for very small caches. Our models are also generalized to account for synchronous requests, fetch latency, and item sizes, extending the applicability of approximations for list-based caches.**

I. Introduction

Replacement policies provide a strategy to select items to replace in a cache and are an important performance factor influencing the design of web systems [1], content delivery networks [2], edge networks [3], and peer-to-peer traffic [4], among others. The focus of this paper is on the theoretical analysis of randomized policies operating within a single cache, a problem that has attracted much attention over the years [5]–[13]. We consider *list-based* caches, which can deliver lower miss rates than single-list caches [14], [15], but due to state-space explosion issues are still not fully understood from a theoretical standpoint. We here focus in particular on a general class of list-based caches that admit a tree structure for the lists, rather than linear as in earlier work on these models [15], and where access to the cache is non-uniform, in the sense that items are promoted within the tree based on probabilities that depend on the item, its current list, and the stream requesting it. Splitting the cache into a tree structure can be used to differentiate the way a cache protects items from eviction. For example, items of one stream may be stored in a different list than items of another stream to avoid cache pollution and optimize hit ratios.

G. Casale is with the Department of Computing at Imperial College London, UK; e-mail: (g.casale@imperial.ac.uk).

N. Gast is with Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France; e-mail: (nicolas.gast@inria.fr).

The authors would like to thank Benny van Houdt for helpful comments during preparation of this manuscript. The work of G.C. was supported by the EU H2020 grants DICE (644869) and RADON (825040) and by a UK EPSRC grant (EP/L00738X/1, iBids). The work of N.G. was supported by the ANR project REFINO (ANR-19-CE23-0015). Research data is available at http://doi.org/10.5281/zenodo.1134886 (CC BY 4.0 license).

Our reference model consists of a cache partitioned into h lists, each having a capacity of $m_l \geq 1$ items, $l = 1, \ldots, h$. Let the capacity vector be $\boldsymbol{m} = (m_1, \ldots, m_h)$ so that the cache has total capacity $m = \sum_{l=1}^{h} m_l$. The total number of items is n and it is assumed to exceed the cache capacity ($n > m$) so that a replacement policy needs to decide which items to evict to make room for a new item. The challenge in designing effective replacement policies is that item popularity is not known beforehand, otherwise one could minimize miss rates by keeping the most frequently accessed items in the cache [16]. The focus is therefore on policies that self-organize item storage so to optimize miss rates.

This paper stems from the recent development in [15] of a framework to analyze *linear* list-based caches with the random replacement (RR), first-in first-out (FIFO), or CLIMB policies, under the independent reference model (IRM) [17], [18]. In the list-based setting, these policies generalize into RR($\boldsymbol{m}$), FIFO($\boldsymbol{m}$), CLIMB($\boldsymbol{m}$) [15], [19] and can deliver lower miss rates than single-list caches, even when the latter are equipped with the LRU policy. In the context of non-uniform access probabilities, we further evolve these policies into two policies called RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$), which may be seen a list-based extensions of policies for single-list caches where probabilities are used to capture access cost [16].

Our main contributions to the performance analysis of these caches are as follows. After showing that known product-form results generalize to the class of models we consider, we develop novel exact and approximate performance analysis methods to investigate RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$). These include a highly-accurate singular perturbation method, bridged from geometric optics and queueing theory [20], and a refined mean-field approximation [21], which can analyze the asymptotic behavior of the cache in both transient and equilibrium regimes.

The paper also delivers a number of extensions to the reference models as follows. Often, caching systems are accessed by applications with a finite pool size, in which the arrival time of the next request from a stream depends on the fetch latency for the previous request. Stemming from the observation that this cache dynamics behaves qualitatively as a closed system, we combine our results with algorithms for closed queueing network theory, developing an approximate solution technique for models with synchronous requests. Our approximation is fairly general and it is shown to be able to accurately account for queueing due to miss and fetch latency.

Moreover, we show that our approach can be generalized to impose hard bounds on costs for the stored items. For example, we illustrate that it is possible to analyze our reference model in the case where one accounts for item sizes and limited size

for individual lists, or globally for the cache as a whole.

The paper is organized as follows. Section I-A overviews related work. The reference model is introduced in Section II and its equilibrium analyzed in Section III. An exact analysis is developed in Section IV. Section V approximates the normalizing constant. Iterative analysis and bounds are introduced in Section VI. Numerical results are reported in Section VII. An approximation for models with synchronous requests and fetch latencies is presented in Section VIII, followed by a cost analysis method in Section IX and the conclusion. Proofs are given in the Appendix. Additional material is included within an online supplement. The present work delivers an extension of a preliminary paper that analyzes non-uniform access probabilities in a less general family of list-based caches with linear structure [22].

A. Related work

Recently, most of the research literature has focused on time-to-live (TTL) caches, which can be analyzed either by exact methods [23] or by characteristic time approximation [24]. The later approach has also been shown to be valid for randomized policies [25]. However, there is still a limited body of work on using this method to study list-based caches [19] and therefore our paper strikes a contribution towards performance analysis of this particular family of stochastic systems.

This paper is related to works that consider the independent reference model (IRM), which include [18], [7], [5], [6], [26], and references therein. Although real systems deviate from the IRM assumptions, IRM theoretical results provide useful insights, such as the characterization of the miss ratio of LRU [9], and formulas for the approximate analysis of cache networks [27]. The work in [28] further adopts an IRM model to help designers compare trade-offs in hybrid memory systems, first extending the linear model in [15] to a parallel structure. Recently, some frameworks that simplify the mathematical analysis of caches have been investigated [14], [15]. This paper generalizes some of the ideas in [15], which provides a mathematical framework to study list-based caches. [28] offers another example of generalization of [15], but to list-based caches with flat and layered list topologies. Compared to the above works, our results apply to general tree-structured caches and to non-uniform access.

In the literature, [16] is among the first works to analyze the least-recently used (LRU) and CLIMB policies when an item has an access probability. The authors in [29] study optimal policies for a single-list with non-uniform access. The authors use a randomization approach to model costs as probabilities, so that items are promoted to a deeper list as a result of a cache hit. This definition is rather flexible, for example it can be extended to take into account item sizes through a notion of density [16]. However, models with non-uniform accesses are generally harder to deal with, since the access probability need not be monotonic through the list hierarchy. Other probabilistic algorithms are surveyed in [1].

II. Preliminaries

A summary of the main notation used in the paper is given in Table 1. We initially take the standard assumption that items have equal sizes; an analysis of the case where items have variable sizes is given in Section IX. In our cache model, h lists are arranged into a fairly general tree topology in which every subtree is disjoint. Special cases of tree topologies, such as single or linear structures, are allowed within this definition. Some possible instances of the list-based caches under consideration are given in Figure 1. We take the convention that a virtual list with index $l = 0$, called the root list, contains all the items outside the cache. In this way, cache misses are simply modeled as standard accesses to list 0.

TABLE I: Summary of main notation

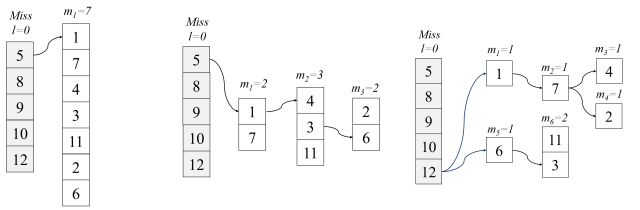
1_j	unit vector in direction j
$c_{vk}(l,j)$	cost (probability) for item k in list l to access list j when requested by stream v
$E(\boldsymbol{m})$	normalizing constant of the equilibrium state probabilities
$E_k(\boldsymbol{m})$	normalizing constant for a model without item k
γ_{kl}	access factor for item k to access list l
h	number of lists
$\lambda_{vk}(l)$	stream-v request rate for an item k residing in list l
m	cache capacity
$\boldsymbol{m}$	cache capacity vector
m_j	capacity of list j
$M(\boldsymbol{m})$	miss rate
$M_k(\boldsymbol{m})$	item-k miss rate
$M_{vk}(\boldsymbol{m})$	item-k miss rate relatively to requests from stream v
n	number of items
π_{kl}	probability that item k resides in list l
$\boldsymbol{s}$	state vector
$s(i,k)$	index of the item in position i of list k
u	number of streams

The set of cached items changes over time. The time required to replace an item is assumed to be negligible compared to inter-request time, and thus treated as instantaneous. We assume that, for each list $l = 1, \ldots, h$ there exists one and only one parent list $p(l)$ from which cache items can be promoted into list l, simultaneously demoting an item from l into $p(l)$. A list l can be parent list for an arbitrary number of children lists and there is no requirement for the number of children to be constant. A list j without children is called a *leaf list*. To require that the lists are connected and rooted in list 0, we ask that recursive application of the function $p(l)$ eventually returns the index of the root list 0, i.e., $p(p(\cdots p(l))) = 0$.

The cache serves u independent streams, each issuing requests for item k in list l according to an exogenous Poisson arrival process with rate $\lambda_{vk}(l) \geq 0$, $v = 1, \ldots, u$, $k = 1, \ldots, n$, $l = 0, \ldots, h$. The rates $\lambda_{vk}(0)$ for items in the virtual list 0 represent the arrival rates of requests for items that are currently out of the cache and which therefore produce cache misses.

A. Non-uniform access

Let the *access probability* $c_{vk}(l,j)$ be the probability that item k, currently residing in list l, is moved by the replacement policy to list j after a hit from a request issued by stream v. Matrix $C_{vk} = [c_{vk}(l,j)]$, $l, j = 0, \ldots, h$, may be seen as a discrete-time Markov chain regulating probabilistically the access to lists. In particular, the tree structure of the cache requires that $c_{vk}(l,j) > 0$ whenever $l = p(j)$ and $c_{vk}(l,j) = 0$ otherwise. According to these definitions, note that $c_{vk}(l,l) =$



(a) Single list (b) Linear structure (c) Tree structure

Fig. 1: Examples of list-based caches with different internal structure, total capacity $m = 7$, and $n = 12$ items.

Algorithm 1 RR-C($\boldsymbol{m}$) policy

Input: requested item i, requesting stream v
list(i) = current list of item i
if list(i) is not a leaf list **then**
  Pick newlist(i) with probability $c_{vi}(\text{list}(i), \text{newlist}(i))$
  if newlist(i) $\neq$ list(i) **then**
    Pick uniformly at random an item k in newlist(i)
    Swap the positions of i and k
    return *// Commit cache state change*
  end if
end if
return *// No cache state change*

$1 - \sum_{j=1, j\neq l}^{h} c_{vk}(l, j)$ is interpreted as the probability that the item is served and remains in its current list l in the same position.

As mentioned in the introduction, and for consistency with earlier work that interprets probabilities as access costs [16], the variants of RR($\boldsymbol{m}$) and FIFO($\boldsymbol{m}$) that allow for non-uniform access are called RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$). Based on the above definitions, we define RR-C($\boldsymbol{m}$), FIFO-C($\boldsymbol{m}$), as shown in Algorithm 1 and 2. CLIMB-C($\boldsymbol{m}$) follows from the above definitions as the extreme case of FIFO-C($\boldsymbol{m}$) and RR-C($\boldsymbol{m}$) when $\boldsymbol{m} = (1, \ldots, 1)$ [8].

B. Motivating Example

We use a small example to illustrate the benefits of tree structures and non-uniform access. We consider $n = 10$ items accessed by $u = 2$ streams. Stream 1 can only request the first five items at rate $\lambda_{1k}(j) = 0.9$, $\forall j$, $k = 1, \ldots, n/2$. Stream 2 can instead access the other items at rate $\lambda_{2k}(j) = 1.0$, $\forall j$, $k = n/2+1, \ldots, n$. We compare several caches with identical total capacity, but with different internal structures and access probabilities. In particular, we show the effect of reserving space for Stream 2 using non-uniform access. All caches use random replacement or CLIMB and are as follows:

1) Single cache ($h = 1$, $m = 6$)
2) Uniform access linear cache ($h = 4$, $\boldsymbol{m} = (2, 1, 1, 2)$)
3) Uniform access linear cache with CLIMB-($\boldsymbol{m}$) ($h = 6$, $\boldsymbol{m} = (1, 1, 1, 1, 1, 1)$)
4) Non-uniform access linear cache ($h = 4$, $\boldsymbol{m} = (2, 1, 1, 2)$). Stream 1 items cannot access lists 3 and 4.
5) Non-uniform access linear cache with CLIMB-($\boldsymbol{m}$) ($h = 6$, $\boldsymbol{m} = (1, 1, 1, 1, 1, 1)$). Stream 1 items cannot access lists 4, 5, and 6.

Algorithm 2 FIFO-C($\boldsymbol{m}$) policy

Input: requested item i, requesting stream v
list(i) = current list of item i
pos(i) = position of i in list(i) *// head = 1, tail =* $m_{list(i)}$
if list(i) is not a leaf list **then**
  Pick newlist(i) with probability $c_{vi}(\text{list}(i), \text{newlist}(i))$
  if newlist(i) $\neq$ list(i) **then**
    Move the item currently at the tail of newlist(i) into list(i) at position pos(i)
    for position $j = m_{\text{newlist}(i)} - 1, \ldots, 1$ **do**
      Move the item at position j of newlist(i) into position $j + 1$ within the same list
    end for
    Place item i at the head of newlist(i)
    return *// Commit cache state change*
  end if
end if
return *// No cache state change*

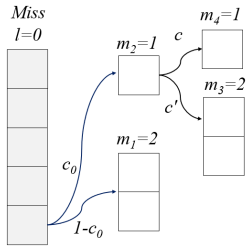


Fig. 2: Motivating example: a tree cache

6) Uniform access tree cache as in Figure 2 ($h = 4$, $\boldsymbol{m} = (2, 1, 1, 2)$). Both streams have $c_0 = 0.5$ and $c = c' = 0.5$.
7) Non-uniform access tree cache as in Figure 2 ($h = 4$, $\boldsymbol{m} = (2, 1, 1, 2)$). Stream 1 has $c_0 = 0.025$, $c = c' = 0$. Stream 2 has $c_0 = 1.0$ and $c = c' = 0.5$. Stream 1 items cannot access lists 3 and 4.

Performance analysis results are given in Table II, showing the miss rates, total and per-stream. The results indicate that, while all methods are relatively close to each other in terms of miss rate, the combination of non-uniform access probabilities and tree structure delivers the best performance. Further, the tree with uniform access is instead performing best in terms of fairness by balancing the miss rates of the two streams. Overall, this small scale example illustrates the principle that the extra flexibility that comes with non-uniform access and non-linear cache structures can be both beneficial to tune cache performance and to differentiate performance across streams (i.e., users). Such flexibility does not exist in models with uniform access, and justify our interest for this class of models. However, the large range of possible structures and non-uniform access probabilities motivates investigating efficient performance analysis methods, to guide in the selection of the best structure for the cache in reference scenarios.

TABLE II: Miss rates in the motivating example

		Miss rate		
Model	Cache structure	Total	Stream 1	Stream 2
1	Single	3.7930	1.8632	1.9298
2	Linear	3.7825	1.9575	1.8251
3	CLIMB	3.7756	2.0197	1.7559
4	Tree	3.7895	1.8947	1.8947
5	Linear non-uniform	3.7085	2.6236	1.0849
6	CLIMB non-uniform	3.7055	2.6501	1.0554
7	Tree non-uniform	3.7033	2.6706	1.0326

III. Analytical model

A. Markov process

Define the cache state vector $\boldsymbol{s} = [s(i,j)]$, such that $s(i,j) \in [1,n]$ is the index of the item in position i of list j, and let $\mathcal{S}$ be the space of cache states that are reachable from the initial state. We take the convention that $m_0 = n - m$ represents the number of items outside the cache, i.e., the capacity of list 0, and that $s(i,0)$ indicates the item with the ith largest index among those outside the cache.

Let $\pi(\boldsymbol{s})$ be the equilibrium probability of the continuous-time Markov chain (CTMC) modeling the chosen replacement policy. From now on, we restrict our attention to recurrent states and assume the CTMC to be irreducible. We also require that for each list l there exists at least m_l items that can reach it under requests from one or more streams.

The following product-form result for $\pi(\boldsymbol{s})$ generalizes similar expression in earlier work [15] by showing the relationship between the equilibrium distribution of the cache and the access probabilities.

Theorem 1. *RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$) admit the product-form equilibrium distribution*

$$\pi(\boldsymbol{s}) = \frac{\prod_{j=0}^{h}\prod_{i=1}^{m_j}\gamma_{s(i,j)j}}{E(\boldsymbol{m})} \tag{1}$$

for all recurrent states $\boldsymbol{s} \in \mathcal{S}$, where $E(\boldsymbol{m})$ is a normalizing constant and define $\gamma_{i,0} = 1$ and

$$\gamma_{ij} = \gamma_{ip(j)}\sum_{v=1}^{u}\lambda_{vi}(p(j))c_{vi}(p(j),j)$$

for all $i = 1,\dots,n$ and $j = 1,\dots,h$.

The proof is given in Appendix A. The result generalizes prior work on list-based caches with a linear structure to tree structures and non-uniform access probabilities.

We refer to γ_{ij} as the *access factor* for item i to access list j. In prior work [15], this term specializes to $\gamma_{ij} = r_i^j$ in discrete-time, r_i being the popularity of item i. The significance of the extension is that we allow for list- and item-dependent behaviors, which are required to account for non-uniform access probabilities. Items can behave differently inside the cache, and be requested at varying rates depending on the current list, e.g., due to different fetch latency as we discuss in Section VIII, or produce different impacts on different streams, as we have shown in Table II. Compared to models such as those in [15], monotonicity properties also no longer apply, in the sense that reaching a deeper position in the list does not necessarily translate into a lower miss rate for the item, as this depends on the particular tree structure and also on the access patterns of the other items. As such, intuition on these models needs to be supported by quantitative methods to avoid pitfalls, for example in assigning list capacities. This motivates the development of specialized performance analysis techniques in the following sections.

B. Performance measures

We now give some standard definitions for the performance measures of interest. These definitions are consistent with the ones used in prior work. Let $\pi_{kl}(\boldsymbol{m})$ be the (marginal) probability that item k is in list l, and let $\pi_{k0}(\boldsymbol{m})$ denote the miss ratio for item k. By definition we have $\sum_{l=0}^{h}\pi_{kl}(\boldsymbol{m}) = 1$ and since at all times list l contains exactly m_l items it is also $\sum_{k=1}^{n}\pi_{kl}(\boldsymbol{m}) = m_l$. Further, using the product-form result (1), we can write

$$\pi_{kl}(\boldsymbol{m}) = m_l\gamma_{kl}\frac{E_k(\boldsymbol{m}-1_l)}{E(\boldsymbol{m})} \tag{2}$$

where 1_l is the unit vector in direction l if $l > 0$ or $1_0 = 0$ otherwise. $E_k(\boldsymbol{m})$ is the normalizing constant of a model without item k. Note that, in particular, the formula implies that the miss ratio for item k is $\pi_{k0}(\boldsymbol{m}) = E_k(\boldsymbol{m})/E(\boldsymbol{m})$.

From the definitions, the miss rate for stream v and item k may be written as $M_{vk}(\boldsymbol{m}) = \pi_{k0}(\boldsymbol{m})\lambda_{vk}(0)(1 - c_{vk}(0,0))$, where $1 - c_{vk}(0,0)$ is the probability that an item outside the cache is cached after a cache miss. The miss rate of item k is then $M_k(\boldsymbol{m}) = \sum_v M_{vk}(\boldsymbol{m})$ and the cache miss rate is $M(\boldsymbol{m}) = \sum_{v,k} M_{vk}(\boldsymbol{m})$.

IV. Exact analysis

In this section, we derive recurrence relations to calculate the performance measures. The algorithms we derive are exact and require polynomial time and space in n and m for constant h. Thus, they ease the problem that the state space $\mathcal{S}$ grows exponentially in size with n and m, however their cost still make them applicable only in small models. This is addressed with the approximate results in the next sections.

A. Recurrence relation for the normalizing constant

We begin by conditioning on the list containing item k, so that we can recursively express the normalizing constant as

$$E(\boldsymbol{m}) = E_k(\boldsymbol{m}) + \sum_{j=1}^{h} m_j\gamma_{kj}E_k(\boldsymbol{m}-1_j) \tag{3}$$

for an arbitrary $k = 1,\dots,n$. Here, m_j accounts for the permutations of the positions of item k within list j. This is a slight generalization of a similar expression obtained in [15] for uniform access models. The recurrence relation requires the boundary conditions $E(\boldsymbol{0}) = 1$, and $E(\boldsymbol{m}) = 0$ if either $\sum_i m_i > n$ or $m_j < 0$ for any j.

Using (3), $E(\boldsymbol{m})$ can be computed in $\mathcal{O}(n\prod_{i=1}^{h}(m_i+1))$ time and space. Standard dynamic scaling methods may be used to alleviate floating-point range exceptions in large models [30]. Additional recurrence relations related to (3) are developed in Appendix E in the supplemental material.

B. Exact analysis of marginal probabilities

Let $\pi_k(\boldsymbol{m}) = \sum_{l=1}^{h} \pi_{kl}(\boldsymbol{m})$ be the probability that item k is cached. We begin by deriving the following relation for marginal state probabilities

Theorem 2. *The probability of finding item k in list l is*

$$\pi_{kl}(\boldsymbol{m}) = \gamma_{kl}\xi_l(\boldsymbol{m})(1 - \pi_k(\boldsymbol{m} - 1_l)) \tag{4}$$

for all items $k = 1, \ldots, n$ and lists $l = 1, \ldots, j$, and where

$$\xi_l(\boldsymbol{m}) = m_l \frac{E(\boldsymbol{m} - 1_l)}{E(\boldsymbol{m})} \tag{5}$$

The proof is given in Appendix B. It is interesting to observe that (4) is similar to the arrival theorem for queueing networks [31]. It may be seen as relating the equilibrium distribution of the cache with the conditional distribution of states seen by item k while residing in list l. A similar argument holds also for the arrival theorem, which is sometimes seen as relating the equilibrium distribution of a closed product-form queueing network with the conditional distribution while a job resides at a given station [32]. We exploit this connection to develop a recurrence relation for the marginal probabilities that is similar in structure to the mean-value analysis algorithm [31].

Since $\sum_{k=1}^{n} \pi_{kl}(\boldsymbol{m}) = m_l$, using (4) and solving for $\xi_l(\boldsymbol{m})$ we find

$$\xi_l(\boldsymbol{m}) = \frac{m_l}{\sum_{k=1}^{n} \gamma_{kl}(1 - \pi_k(\boldsymbol{m} - 1_l))} \tag{6}$$

Finally, plugging (6) into (4), and using that $\pi_k(\boldsymbol{m}) = \sum_{l=1}^{h} \pi_{kl}(\boldsymbol{m})$, we get the exact recurrence relation

$$\pi_{il}(\boldsymbol{m}) = \frac{m_l \gamma_{il}(1 - \sum_{j=1}^{h} \pi_{ij}(\boldsymbol{m} - 1_l))}{\sum_{k=1}^{n} \gamma_{kl}(1 - \sum_{j=1}^{h} \pi_{kj}(\boldsymbol{m} - 1_l))} \tag{7}$$

for all $i = 1, \ldots, n$ and $l = 1, \ldots, h$. The recursion has termination conditions $\pi_{il}(\boldsymbol{m}) = 0$ if either $\boldsymbol{m} = \mathbf{0}$ or $\min_j m_j < 0$. Solving (7) recursively requires $\mathcal{O}(nh \prod_{i=1}^{h}(m_i + 1))$ time and $\mathcal{O}(n \prod_{i=1}^{h}(m_i + 1))$ space. Item miss rates may be easily computed by the expressions in Section III-B.

V. Normalizing constant approximation

Exact recurrence relations are too expensive to apply to models of medium and large caches, because as shown in the last section their cost grows exponentially in the number of lists h. To address this issue, we study the asymptotic limit for $E(\boldsymbol{m})$ to approximate performance measures. The main contribution is to develop, under mild assumptions, the leading term in the asymptotic expansion for $E(\boldsymbol{m})$ when $n \to \infty$ and $m \to \infty$, with $n/m \sim O(1)$. To this aim, we use the singular perturbation method [20], which can determine the asymptotic solution of multivariate recurrence relations by means of tractable partial differential equations (PDEs).

A. Overview

Before giving the formal derivation of the asymptotics, we sketch the solution paradigm of the singular perturbation method proposed in [20] once applied in a similar fashion to our product-form solution, which presents structural similarities with the one characterizing queueing network models considered in the original paper.. We point the interested reader to the tutorial in [33] for more details.

The singular perturbation method may be used to obtain approximations to multivariate recurrence relations by studying their behavior under perturbations in their parameters. In our context, without loss of generality, we focus on relation (3) with $k = n$ and set $H(\boldsymbol{m}) = E(\boldsymbol{m}) / \prod_{j=1}^{h} m_j!$ and $H_k(\boldsymbol{m}) = E_k(\boldsymbol{m}) / \prod_{j=1}^{h} m_j!$, so that it may be rewritten as

$$H(\boldsymbol{m}) = H_n(\boldsymbol{m}) + \sum_{j=1}^{h} \gamma_{nj} H_n(\boldsymbol{m} - 1_j) \tag{8}$$

We now scale the recurrence parameters as $\boldsymbol{x} = \boldsymbol{m}\varepsilon$ and $y = n\varepsilon$, for some small perturbation ε, and rewrite it as

$$h(\boldsymbol{x}, y) = h(\boldsymbol{x}, y - \varepsilon) + \sum_{j=1}^{h} g_j(y) h(\boldsymbol{x} - \varepsilon 1_j, y - \varepsilon) \tag{9}$$

where $x = \sum_j x_j$ and we define $h(\boldsymbol{x}, y) = H_{y/\varepsilon}(\boldsymbol{x}/\varepsilon)$, which is a scaled version of $H(\boldsymbol{m})$ in which $\boldsymbol{x}$ corresponds to list capacities and y to the number of items, and in which access factors are modeled by the $g_j(y)$ functions. To match the coefficients of the initial expression, we have assumed that we can replace γ_{nj} with a smooth function $g_j(y)$ defined such that $g_j(k\varepsilon) = \gamma_{kj}$ and $g_j(0\varepsilon) = g_j(n\varepsilon)$, $\forall j$.

1) Analytical form: Our goal is to approximate asymptotically the function $h(\boldsymbol{x}, y)$. As an example, consider the case where $\gamma_{kj} = g_j(k) = \gamma_j$, $\forall k, j$, for arbitrary γ and denote the normalizing constant for this model by $H_0(\boldsymbol{m}, n)$. Since all the items have identical access factors to each list, it is possible to show that

$$H_0(\boldsymbol{m}, n) = \frac{n!}{(n-m)! m_1! \cdots m_h!} \prod_{j=1}^{h} \gamma_j^{m_j}$$

Using Stirling approximation $n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$, and scaling the variables in $H_0(\boldsymbol{m}, n)$ we obtain an asymptotically exact form for the normalizing constant given by

$$h_0(\boldsymbol{x}, y) \sim \frac{(2\pi/\varepsilon)^{-h/2}}{\sqrt{\prod_j x_j}} \sqrt{\frac{y}{y - x}} e^{\phi_0(\boldsymbol{x}, y)/\varepsilon} \tag{10}$$

where $\phi_0(\boldsymbol{x}, y) = y \log y - (y - x) \log(y - x) - \sum_j x_j \log x_j + \sum_j x_j \log \gamma_j$ is a function independent of ε.

As (10) is one particular example of asymptotic solution, the general case needs to include an exponential dependence on $1/\varepsilon$ to match it. As such, we look for solutions matching the quite general form

$$h(\boldsymbol{x}, y) \sim D(\varepsilon) B(\boldsymbol{x}, y) e^{\phi(\boldsymbol{x}, y)/\varepsilon} (1 + O(1)) \tag{11}$$

for some arbitrary functions $\phi(\boldsymbol{x}, y)$, $B(\boldsymbol{x}, y)$, and $D(\varepsilon)$ to be determined. Since (10) is an exact asymptotics, the assumed form (11) is guaranteed to converge to the exact solution in the limit of a sequence of models where n and m grow large in a constant ratio when γ_{kj} within each list are asymptotically identical to a constant γ_j. Thus, we expect the errors to be

larger for finite models with small caches and where the access factors γ_{kj} differ significantly. In spite of this, we show later in the numerical experiments that the approximation is empirically very accurate also in small caches and when the access factors are very different, due to Zipf-like item popularity.

2) Multivariate recurrence relations: The structure of (11) is particularly suitable for approximating linear multivariate recurrence relations such as (9), since the exponential term allows for simplifications. Plugging (11) into (9) we get

$$B(\boldsymbol{x},y)e^{\phi(\boldsymbol{x},y)/\varepsilon} = B(\boldsymbol{x},y-\varepsilon)e^{\phi(\boldsymbol{x},y-\varepsilon)/\varepsilon} + \textstyle\sum_{j=1}^{h} g_j(y)B(\boldsymbol{x}-\varepsilon 1_j, y-\varepsilon)e^{\phi(\boldsymbol{x}-\varepsilon 1_j,y-\varepsilon)/\varepsilon} \quad (12)$$

We now expand both sides in powers of ε, obtaining in particular the $O(1)$ and $O(\varepsilon)$ terms. A typical approach of the method is to treat the terms at each order separately, equating both sides of the equation individually at each order. This relies on the observation that as $\varepsilon \to 0$, the $O(\epsilon)$ term will vanish. With this approach, at the lowest order, one notices first that by Taylor expansion

$$\phi(\boldsymbol{x}-\varepsilon 1_j, y-\varepsilon) = \phi(\boldsymbol{x},y) - \phi_y(\boldsymbol{x},y)\varepsilon - \sum_{j=1}^{h}\phi_j(\boldsymbol{x},y)\varepsilon,$$

where ϕ_j (resp. ϕ_y) denotes partial differentiation with respect to x_j (resp. y). Plugging back into (12) in the limit $\varepsilon \to 0$ we get after simplifying the following PDE

$$e^{\phi_y} - \sum_j g_j(y)e^{-\phi_j} = 1$$

This is a PDE with linear degree with respect to the partial derivatives, which can be solved explicitly, as shown in Appendix C.

A similar derivation for the $O(\epsilon)$ terms produces a more complex equation in the second-order partial derivatives, which may also be explicitly solved. The derivation relies on boundary conditions which we choose to be the analytical solution for a cache with uniform item popularity shown in (10). Once such explicit solutions are obtained, (11) is scaled back to approximate the original normalizing constant $E(\boldsymbol{m})$.

B. Main result

The formal derivation to obtain $\phi(\boldsymbol{x},y)$, $B(\boldsymbol{x},y)$, and $D(\varepsilon)$ from (9) is given in Appendix C, and leads to

$$\phi(\boldsymbol{x},y) = \int_0^y \log\left(1+\sum_{j=1}^{h} g_j(v)\xi_j\right)dv - \sum_{j=1}^{h} x_j \log\xi_j \quad (13)$$

where the terms ξ_j, $j=1,\dots,h$ satisfy

$$x_j = \int_0^y \frac{g_j(v)\xi_j}{1+\sum_{l=1}^{h} g_l(v)\xi_l}dv \quad (14)$$

We show later in Section VI-A that the terms ξ_l may be seen as the limits of $\xi_l(\boldsymbol{m})$ defined in (6). The remaining coefficients are $D(\varepsilon) = \varepsilon^{h/2}$ and

$$B(\boldsymbol{x},y) = \frac{(2\pi)^{-h/2}}{\sqrt{\det \boldsymbol{J}}\sqrt{\xi_1\cdots\xi_h}} \quad (15)$$

where the matrix $\boldsymbol{J} = [J_{jl}]$, $j,l = 1,\dots,h$, is given by

$$J_{jl} = \int_0^y \frac{\delta_{jl}g_j(v)}{1+\sum_r g_r(v)\xi_r}dv - \int_0^y \frac{\xi_j g_j(v) g_l(v)}{(1+\sum_r g_r(v)\xi_r)^2}dv \quad (16)$$

with $\delta_{jl} = 1$ if $j = l$, and $\delta_{jl} = 0$ otherwise.

C. Approximating a finite system

Having determined the asymptotics of $h(\boldsymbol{x},y)$, we can now use the Euler-Maclaurin formula [34] to rewrite (11) in terms of the variables $\boldsymbol{m}$ and n. For a function $f(x)$ defined in $[0,n]$, the Euler-Maclaurin formula may be written as

$$\sum_{k=1}^{n} f(k) = \int_0^n f(x)dx + \frac{f(n)-f(0)}{2} + O(n^{-1})$$

Scaling variables in (14) so that the integral ranges in $[0,n]$, the error of the Euler-Maclaurin formula is $O(\varepsilon)$, and thus the resulting finite formula is asymptotically exact. Since we have chosen $g_j(0\varepsilon) \equiv g_j(n\varepsilon)$, this is given by

$$m_j = \sum_{k=1}^{n} \frac{\gamma_{kj}\xi_j}{1+\sum_{l=1}^{h}\gamma_{kl}\xi_l} \quad (17)$$

for $j = 1,\dots,h$. Under uniform access, [15] obtains an expression similar to (17) as the steady-state limit of mean field ordinary differential equations. Our derivation shows that a similar result holds under non-uniform access.

Using Euler-Maclaurin we can also write

$$\phi(\boldsymbol{m}) = \sum_{k=1}^{n}\log\left(1+\sum_{j=1}^{h}\gamma_{kj}\xi_j\right) - \sum_{j=1}^{h} m_j\log\xi_j$$

and replace $\boldsymbol{J}$ with matrix $\boldsymbol{C} = [C_{jl}]$ where

$$C_{jl} = \sum_{k=1}^{n}\frac{\delta_{jl}\gamma_{kj}}{1+\sum_{r=1}^{h}\gamma_{kr}\xi_r} - \sum_{k=1}^{n}\frac{\xi_j\gamma_{kj}\gamma_{kl}}{(1+\sum_{r=1}^{h}\gamma_{kr}\xi_r)^2}$$

The above expressions provide by (11) the asymptotic expansion of $H(\boldsymbol{m})$, which implies under the assumptions the following expansion of the original normalizing constant

$$E(\boldsymbol{m}) \sim \frac{1}{(2\pi)^{h/2}}\frac{\prod_{k=1}^{n}\left(1+\sum_{j=1}^{h}\gamma_{kj}\xi_j\right)}{\prod_{j=1}^{h}\xi_j^{m_j+1/2}}\frac{\prod_{j=1}^{h}m_j!}{\sqrt{\det\boldsymbol{C}}} \quad (18)$$

where the ξ_j terms are obtained from (17).

D. Performance measures

For large $\boldsymbol{m}$, we can expand $E_k(\boldsymbol{m}-1_l)$ in a neighborhood of $\boldsymbol{m}$ to obtain by (18) the leading terms

$$\pi_{kl}(\boldsymbol{m}) \sim \begin{cases} \dfrac{1}{1+\sum_{j=1}^{h}\gamma_{kj}\xi_j} & \text{if } l = 0 \\ \dfrac{\gamma_{kl}\xi_l}{1+\sum_{j=1}^{h}\gamma_{kj}\xi_j} & \text{if } l \geq 1 \end{cases} \quad (19)$$

for all $k = 1,\dots,n$ and $l = 0,\dots,j$. Additional terms in the expansions may also be used to generate higher-order approximations, which involve the derivatives of $\sqrt{\det\boldsymbol{C}}$. However, these are seldom needed since the computational cost

TABLE III: Example: normalizing constant asymptotics

n	m_1	m_2	$E(\boldsymbol{m})$	SPA - eq. (18)	relative error
4	2	0	$1.2969 \cdot 10^1$	$1.3691 \cdot 10^1$	0.056
8	4	0	$3.5950 \cdot 10^2$	$3.6940 \cdot 10^2$	0.028
16	8	0	$6.7136 \cdot 10^5$	$6.8063 \cdot 10^5$	0.014
20	10	0	$3.8500 \cdot 10^7$	$3.8926 \cdot 10^7$	0.011
4	1	1	$1.6173 \cdot 10^1$	$1.8919 \cdot 10^1$	0.170
8	2	2	$2.5697 \cdot 10^2$	$2.7810 \cdot 10^2$	0.082
16	4	4	$6.2439 \cdot 10^4$	$6.4990 \cdot 10^4$	0.041
20	5	5	$9.7236 \cdot 10^5$	$1.0042 \cdot 10^6$	0.033

is comparable to computing performance measures by direct application of (18) to the definitions given in Section III-B.

In what follows, we call *singular perturbation approximation* (SPA) the latter method, which consists of computing the normalizing constants appearing in performance measures such as (2) and derivatives by means of (18). The approach that numerically solves (17) for the ξ_l terms and then calculates performance measures by (19) is instead developed in the next section and referred to as the *fixed-point iteration* (FPI).

1) Example: Table III illustrates the singular perturbation method. We consider a small-scale model where $E(\boldsymbol{m})$ can be computed numerically using (3). The model has $u = 2$ users, $\lambda_{1k}(l) = k^{-0.6}$, $\lambda_{2k}(l) = k^{-1.4}$, $\forall l$. We consider a linear structure for the cache and set access probabilities to $c_{vk}(l, l+1) = 1$, $l = 0, \ldots, h-1$. We fix $n = 2S$ and $m = S$ and scaling $S = 2, 4, 8, 10$. As shown in the table, (18) converges to the exact value $E(\boldsymbol{m})$ as S grows.

VI. Further approximations

In this section, we introduce three new approximations. We first derive an efficient fixed-point iteration to solve (17). We then explain how to use the recent refined mean field approximation to *refine* this fixed point approximation. We also develop simple bounds to quickly assess performance measures in settings where speed can be more important than high-accuracy, as in computational optimization.

A. Fixed-point iteration (FPI)

We now introduce a fixed-point method to obtain the ξ_l variables. Let us first consider a regular perturbation expansion [35] of (7), which for large m and n, with $m/n \sim O(1)$, takes the form $\pi_{il}(\boldsymbol{m}) \sim \hat{\pi}_{il}^{(0)} + \varepsilon \hat{\pi}_{il}^{(1)} + \ldots$, for small $\varepsilon > 0$. Plugging the last expression in (7), we get at the lowest order

$$\hat{\pi}_{il}^{(0)} = \frac{m_l \gamma_{il} (1 - \sum_{j=1}^{h} \hat{\pi}_{ij}^{(0)})}{\sum_{k=1}^{n} \gamma_{kl} (1 - \sum_{j=1}^{h} \hat{\pi}_{kj}^{(0)})} \tag{20}$$

for $i = 1, \ldots, n$, $l = 1, \ldots, h$. This is a non-linear system of equations with nh equations and nh unknowns. It is possible to verify that the system is solved by setting

$$\hat{\pi}_{il}^{(0)} = \frac{\gamma_{il} \xi_l^{(0)}}{1 + \sum_{j=1}^{h} \gamma_{ij} \xi_j^{(0)}} \tag{21}$$

where $\xi_l^{(0)} = m_l (\sum_{k=1}^{n} \gamma_{kl} (1 - \sum_{l=1}^{h} \hat{\pi}_{il}^{(0)}))^{-1}$. Applying the regular perturbation expansion to the relation $\sum_{k=1}^{n} \pi_{kl}(\boldsymbol{m}) = m_l$, and plugging (21) in the resulting expression, we see that as $\varepsilon \to 0$ the $\xi_l^{(0)}$ terms become solutions of (17). Since $\xi_l^{(0)}$ is the value of ξ_l corresponding to the leading terms of the expansion of $\pi_{il}(\boldsymbol{m})$, this provides support to our earlier statement that the variables ξ_l appearing in the expansion of $E(\boldsymbol{m})$ are the limits of $\xi_l(\boldsymbol{m})$ for $\varepsilon \to 0$. A similar conclusion follows by calculating the leading term of (5) as in Section V-D.

The explicit expression of $\xi_l^{(0)}$ enables us to solve (17) by introducing the following fixed point iteration

$$\xi_l^{(t+1)} = \frac{m_l}{\sum_{k=1}^{n} \gamma_{kl} (1 - \sum_{l=1}^{h} \pi_{il}^{(t)})} \tag{22a}$$

$$\pi_{il}^{(t+1)} = \frac{\gamma_{il} \xi_l^{(t+1)}}{1 + \sum_{j=1}^{h} \gamma_{ij} \xi_j^{(t+1)}} \tag{22b}$$

for all $i = 1, \ldots, n$, $l = 1, \ldots, h$ and iterations $t \geq 0$.

As mentioned before, the FPI method approximates the solution of (17) by (22), and then plugs the obtained ξ_l values into (19) to obtain performance measures. The iteration can be initialized with a guess of the marginal probabilities, e.g., $\pi_{il}^{(0)} = 1/(h+1)$. We stop it when the maximum relative change of the miss ratios $\pi_{i0}^{(t)} = 1 - \sum_l \pi_{il}^{(t)}$ across iterations does not exceed a user-specified tolerance (e.g., $\tau = 10^{-6}$).

Over thousands of models considered in the experiments, none raised convergence issues. More general nonlinear equation solver may be used to find a set of values for ξ_l and π_l which satisfy the equations with convergence guarantees, where needed.

B. Derivation of the refined mean-field approximation (RMF)

Let us indicate with $\hat{\pi}^{FPI}$ the exact solution of (17). For uniform access model, it is possible to show that $\hat{\pi}^{FPI}$ is also a solution of the ordinary differential equations (ODEs) obtained by mean field approximation of the Markov process underlying RR($\boldsymbol{m}$) [15]. This prompts us to investigate the applicability of the mean-field methods also to RR-C($\boldsymbol{m}$).

Mean-field approximations are asymptotic solutions that enable approximate transient and steady-state analysis with accuracy guarantees. Since transient analysis is an important concern in caching systems, we propose in this section a highly-accurate refined mean field (RMF) approximation for RR-C($\boldsymbol{m}$) grounded in the general framework developed in [21]. Note that if RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$) have the same stationary distribution, their transient behavior is quite different. We do not believe that our analysis to study the transient behavior of RR-C($\boldsymbol{m}$) can be easily adapted to FIFO-C($\boldsymbol{m}$).

In RMF, the system scale is controlled by a positive parameter N. The original system is approximated by a scaled one where each item has N identical copies and the cache capacity is increased to mN. We refer to the copies of item i as the class-i items. As in the original model, class-i item are treated with the usual replacement rules of RR-$C(\boldsymbol{m})$.

Let $\pi_{il}^{(N)}$ be the marginal probability for a class-i item to be in list l in the scaled model. Then it is possible to show that there exists a vector $V = [V_{il}]$ such that

$$\pi_{il}^{(N)} = \hat{\pi}_{il}^{FPI} + (1/N) V_{il} + O(N^{-2}) \tag{23}$$

for all items $i = 1, \ldots, N$ and lists $l = 1, \ldots, h$. The RMF approximation consists in using $\widehat{\pi}_{il}^{RMF} = \pi_{il}^{(N)} + V_{il}$, which

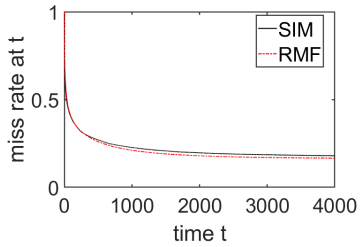


Fig. 3: RMF transient analysis for $n = 50$

corresponds to using the approve expression to the original system with $N = 1$.

A precise derivation of how to compute V is provided in Appendix J in the online supplement and here briefly summarized. Let us denote by $X_{il}^{(N)}(t)$ the fraction of class-i items that are in list l at time t (for the N-scaled model). With this notation, a class-i item from list i is exchanged with a class-j item from list l' at rate $N\sum_{v=1}^{u}\lambda_{iv}(l)c_{iv}(l,l')X_{il}X_{jl'}/m_{l'}$. In this expression, the factor $N\sum_{v=1}^{u}\lambda_{iv}(l)c_{iv}(ll')X_{il}$ is equal to the rate at which a class-i item will be promoted to list l' while $X_{jl'}/m_{l'}$ is the probability that it is an item of class j that will be demoted in exchange. Such a transition changes the vector $X^{(N)}(t) = (X_{il}^{(N)}(t))_{i\le n, l\le h}$ by adding $1/N$ on the its (il') and (jl) coordinates, and by removing $1/N$ on its (il) and (jl') coordinates. This implies that the family of scaled models defines what is called a density dependent population processes [36] and allows us to use the theory in [21] to define the expansion (23).

The method can also be used to study the transient regime [37], as it can be shown that the transient probability satisfies

$$\pi_{il}(t) = x_{il}^{FPI}(t) + \frac{1}{N}V_{il}(t) + O(N^{-2}) \tag{24}$$

where x is the solution of the mean-field ODE given in the supplementary material (see in particular (48)) and $V(t) = [V_{il}(t)]$ is a time-varying vector. The quantities x^{MF} and $V(t)$ satisfy ODEs that can be easily computed numerically.

1) Example: We illustrate the accuracy of the RMF approximation in transient regime by using (24) for the original model with $N = 1$. To compute numerically the values x^{MF} and V, we rely on a fast software implementation[1]. The original model consists of $n = 50$ items, $h = 5$ lists, each having capacity $m_l = 3$. There is a single stream that issues requests at Poisson rate $\lambda_{il} = r_i$, where r_i is the probability of accessing item i, which is Zipf-like distributed with parameter $\alpha = 1.4$.

Figure 3 illustrates the accuracy of the RMF approximation in comparison with estimates of $\pi_{il}(t)$ obtained by averaging 100 cache simulations. In the latter, we use the ratio of miss count to simulation time at t, requiring in the very first samples the value to be upper bounded at the theoretical maximum in this example of 1.0. The example is typical of RMF transient accuracy and shows high precision.

[1]We use rmf-tool (https://github.com/ngast/rmf_tool/) that numerically integrates the ODEs. It uses `scipy.integrate_solve_ivp` from the Python library `scipy`.

C. Bound analysis

Here we develop some simple analytical bounds on $\xi_l(\boldsymbol{m})$ and π_{i0}. Note that by (3) and by the definition of miss rate, we may write

$$M(\boldsymbol{m}) = \frac{E(\boldsymbol{m}+1_l)}{E(\boldsymbol{m})} = \frac{m_1+1}{\xi_1(\boldsymbol{m}+1_l)} \tag{25}$$

Bounds on $\xi_l(\boldsymbol{m})$ allow us to bound the miss rate $M(\boldsymbol{m})$, whereas bounds on the probabilities π_{i0} can be used to bound the miss rates using the definition of $M_{vk}(\boldsymbol{m})$.

1) Bounds on $\xi_l(\boldsymbol{m})$: It is possible to verify that

$$\frac{m_l}{\gamma_l^+(n-m+1)} \le \xi_l(\boldsymbol{m}) \le \frac{m_l}{\gamma_l^-(n-m+1)} \tag{26}$$

where $\gamma_l^+ = \max_k \gamma_{kl}$ and $\gamma_l^- = \min_k \gamma_{kl}$. These follow from (6) by bounding γ_{kl} with γ_l^+ or γ_l^- and noting that $\sum_k(1-\pi_k(\boldsymbol{m}-1_l)) = n-m+1$. For an important class of models, we then state a tighter lower bound. Lower bounds on $\xi_l(\boldsymbol{m})$ are the most important in practice, as they lead to pessimistic estimates of the miss rates. The bound assumptions are satisfied by a rather broad class of models including, but not limited to, models with uniform access.

Theorem 3. *Consider a cache where for each pair of items i and $k > i$, we have $\gamma_{il} \ge \gamma_{kl}$, $\forall l$. Then*

$$\frac{m_l}{\bar{\gamma}_l(n-m+1)} \le \xi_l(\boldsymbol{m}) \tag{27}$$

where $\bar{\gamma}_l = \sum_j \gamma_{jl}/n$ is the average access factor for list l.

The proof is given in Appendix I and shows that the argument generalizes for any list where $\gamma_{kl} \ge \gamma_{il}$ implies $\pi_k(\boldsymbol{m}-1_l) \ge \pi_i(\boldsymbol{m}-1_l)$. If the property holds for all lists, we say that the model is *monotone*.

2) Bounds on $\pi_{k0}(\boldsymbol{m})$: We now develop upper bounds on the miss rate $\pi_{k0}(\boldsymbol{m})$. Let $\xi_j^{-k}(\boldsymbol{m})$ be defined as $\xi_j(\boldsymbol{m})$, but for a model without item k. We are now ready to give an upper bound on the miss rates.

Theorem 4. *The item miss ratios satisfy the bound $\pi_{k0}(\boldsymbol{m}) \le \left(1+\sum_{j=1}^{h}\gamma_{kj}\xi_j(\boldsymbol{m})\right)^{-1}$ for all items $k = 1,\dots,n$ and lists $l = 1,\dots,h$.*

The proof is given in Appendix F. Note that bound is asymptotically exact for fixed h, as it converges to (19) when n and m increase in a fixed ratio. A closed-form expression for this upper bound can be readily obtained by replacing $\xi_l(\boldsymbol{m})$ with the lower bound in either (26) or (27). Although (27) is not guaranteed to be a bound in non-monotone models, combining it with Theorem 4 still provides a useful non-iterative heuristic approximation of π_{k0}, as illustrated in the next example.

3) Example: Figure 4 illustrates Theorem 4 for a monotone and a non-monotone cache. Estimates are computed by replacing $\xi_j(\boldsymbol{m})$ in the bound of Theorem 4 with (27). In monotone caches, which include caches with uniform access as a special case, rank is inverse proportional to item miss rates [15]. However, this clearly does not hold in non-monotone caches, as shown in the figure. As visible from comparison against the exact value, the heuristic provides rather accurate estimates of the item miss ratio.

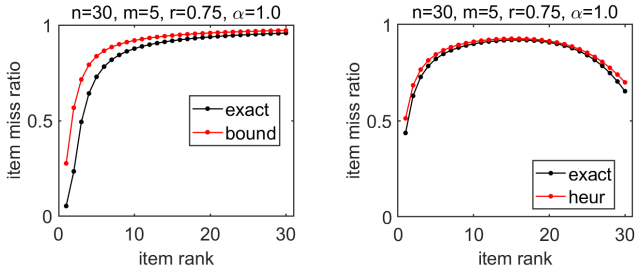


Fig. 4: Single-stream cache with capacity $\boldsymbol{m} = (3, 2)$ and request rates $\lambda_{vk}(l) = r_k\lambda$, with $\lambda = 1$ and Zipf-like popularity $r_k = k^{-\alpha}$. Items are ranked according to r_k. The left figure shows a monotone cache with $c_{vk}(l, l+1) = (1-r)r^k$, the right cache is non-monotone with costs c_{vkl}^{-1}.

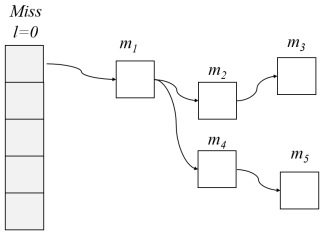


Fig. 5: A tree cache with $h = 5$ lists. List capacities vary between $m_j = 1$ and $m_j = 50$.

VII. Numerical results

A. Simulation-based validation

1) Non-uniform access in linear caches: Since exact numerical solution is applicable only to small models, we have performed a simulation study of RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$) to assess the accuracy of the proposed approximations.

The simulation parameters are the number of items $n \in \{10, 100, 1000\}$, the number of lists $h \in \{1, 2, 5\}$, all list having size $m_i = \lceil n/(\beta h) \rceil$ with an item-capacity ratio $n/m \approx \beta \in \{2, 4, 10\}$, the number of streams $u \in \{1, 4\}$, and the access probability value $c \in \{0.1, 0.5, 1.0\}$. When $n = m$ we set $n = m + 2$. For comparison with prior work, we use linear caches with $c_{vi}(0, 1) = 1$, $c_{vi}(1, l+1) = \ldots = c_{vi}(l, l+1) = c$, with $c = 0.1, 0.5, 1.0$ and for all items i.

Request rate is $\lambda = 1$ and item popularity is Zipf-like with parameter α, with the object ranks assigned uniformly at random within each stream. Each simulation collects 10^9 samples. From the results, we observe as expected that RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$) have statistically indistinguishable performance, thus we discuss only RR-C($\boldsymbol{m}$) Let $\widehat{\pi}_{0k}(\boldsymbol{m})$ be the estimated miss ratio for item k. The error metric is

$$\epsilon^{\pi}_{mape} = \frac{1}{n}\sum_{k=1}^{n}\left|1 - \frac{\widehat{\pi}_{0k}(\boldsymbol{m})}{\pi_{0k}(\boldsymbol{m})}\right| \tag{28}$$

In small models with $n = 10$ items, SPA incurs the lowest mean absolute percentage error, around 0.4% while RMF is around 1.2% and FPI around 10.2%. For such instances, we have observed the maximum error of RMF and FPI to reach respectively 5% and 35.1%, whereas for SPA it is just 0.6%. We have observed the largest errors of FPI to be in the estimation of miss probabilities for high popularity items. Conversely, with $n = 1000$ items we observe a mean absolute percentage error of around 0.6% for the three methods. Note that we believe that, even for $n = 1000$, SPA and RMF should be significantly more accurate than FPI. We do not observe it here as we believe that for $n = 1000$, the three approximations provide estimations of $\pi_{0k}(\boldsymbol{m})$ that are more accurate than a simulation with 10^9 samples.

Sensitivity analysis results on these experiments may be found in [22] and illustrates that the error grows slightly with the ratio n/m and α, but remains very small in magnitude, with a mean around 1% at worst for SPA. Accuracy is slightly better for smaller values of h. The results are practically insensitive to c. Overall, the results indicate that FPI often suffices, but SPA and RMF provide more robust estimates for models far from the asymptotic regime.

2) Non-uniform access in tree caches: The previous section highlights the resilience of the approximations in capturing non-uniform access. We now focus on approximating caches with a tree structures as in Figure 5. To compare accuracy with simpler cases, we also explore the case where the cache in the figure is restricted to a single list ($h = 1$) or the first two lists ($h = 2$). The list capacities are set to $m_j = \lceil m/h \rceil$, $j = 1, \ldots, h$. We vary sequentially these parameters: $n \in \{10, 50, 500\}$, $n/m \in \{2, 4, 10\}$, $\alpha \in \{0.6, 1.0, 1.4\}$, $u \in \{1, 4\}$, $h \in \{1, 2, 5\}$. Conversely, the access probabilities between lists are now randomized uniformly and normalized to form a valid discrete-time Markov chain. For each combination of the other parameters, we consider 30 random instances of the access probabilities. The arrival rate for stream v is set to $\lambda_{vkj} = vr_i$, where r_i follows a Zipf-like distribution with parameter α. The simulations use 10^8 samples.

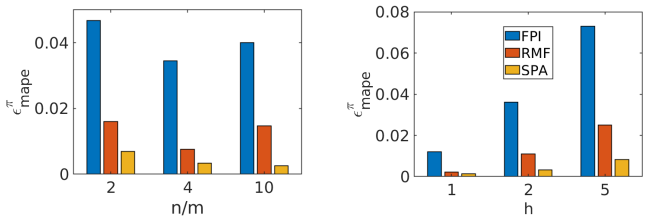
Approximation results are given in Figure 6. We compare FPI, which in fact gives for this system the same solution as the classical mean field approximation, the refined mean field (RMF) and SPA. The results indicate that RMF and SPA improve over FPI thanks to the additional information they carry in the equations. Overall, the average error is 4.04% for FPI, 1.28% for RMF, and 0.43% for SPA.

We have also tried to analyze the mean absolute percentage error on miss rates, i.e.

$$\epsilon_{mape} = \left|1 - \frac{\widehat{M}(\boldsymbol{m})}{M(\boldsymbol{m})}\right| \tag{29}$$

where $\widehat{M}(\boldsymbol{m})$ is the estimated miss rate. For this metric, we observe over all instances 1.75% error for FPI, 0.25% for RMF, and 0.10% for SPA. This is consistent with the intuition that this metric captures average performance and therefore can be better characterized than ϵ^{π}_{mape} by first-order methods such as FPI. The additional terms in RMF and SPA instead capture higher-order interactions between the items more visible upon account the entire miss ratio distribution as in ϵ^{π}_{mape}.

Since caches have typically a large number of objects, we conclude that both RMF and SPA provide an accurate way to analyze list-based caches. The main differences are that SPA has a compact expression and does not require to use ordinary differential equations (ODEs) and thus does not incur issues such as ODE stiffness or implementation-specific overheads



(a) Sensitivity to n/m ratio (b) Sensitivity to h

Fig. 6: Comparison between FPI, SPA and refined mean-field approximation (RMF) on models with random access probabilities.

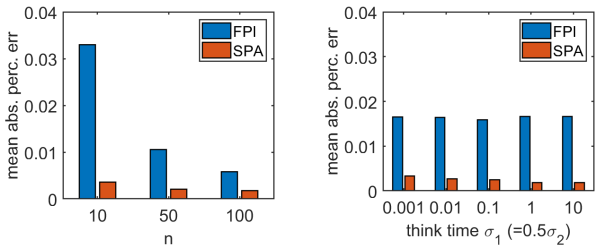


Fig. 7: Item miss ratios for models with synchronous requests.

associated to ODEs. Conversely, RMF may be used also in the transient regime, which is not accessible to SPA, since the method is derived under equilibrium assumptions.

B. Youtube trace

Lastly, we use a trace-driven simulation to illustrate the applicability of the proposed methods to real workloads. We have simulated the 2008 Youtube trace collected in [38] and also used to validate the list-based cache models in [15], [28]. The trace consists of $611,968$ requests, spanning $n = 303,332$ items and originating from $u = 16,337$ client IPs. Each client IP is mapped to a single stream v.

We run trace-driven simulations, using a cache with linear structure and capacities $m = 5000$, $m_1 = 2900$, and $m_2 = \cdots = m_h = (m - m_1)/(h - 1)$ and varying the number of lists $h = 2, 3, 5$, the replacement policy (FIFO-C($\boldsymbol{m}$), RR-C($\boldsymbol{m}$)), and the normalized access probability to the next list ($c = 0.1, 0.5, 1.0$).

For each simulation, we compute the item-miss rates M_k, $k = 1 \ldots, n$, and compare them with the estimates returned by FPI. In the stochastic model, each stream v is modeled as a Poisson process with rate $\lambda_v(k) = n_{vk}/T$, where n_{vk} is the number of times item k is requested by stream v and T is the time span of the trace.

Despite the large number of items, FPI produces an estimate of all item miss rates in just 8.2 seconds on a laptop (Intel Core i5). Even though the Youtube trace is temporally-correlated, thus departing from IRM and Poisson assumptions, and about half of the items are accessed just a single time each, the mean absolute percentage error of FPI on the M_k values is only 1.74%, with a maximum error of just 4.68%. Both values are compatible with the uncertainties of the simulation, in which the trace is simulated only once. Overall, the experiments indicate that our model can reliably reproduce simulation results in the presence of access probabilities and multiple request streams.

VIII. Synchronous streams

We further generalize the reference model to include synchronous streams, i.e., arrival processes where the stream awaits reply from the previous request before issuing a new one, as in pooled cache access. We assume the following reference model. Stream v can have up to s_v outstanding requests. The inter-request time is the sum of an exponential time with mean σ_v, representing an arbitrary delay in-between successive requests (think time), and a miss (or fetch) latency, also exponential with mean θ_{v0} (or θ_{v1} for fetch). The cache state is updated at arrival time, latency (either due to miss or due to fetch, but not both) is incurred right after.

We wish to determine the cache arrival rates, which at steady-state match the sum of hit and miss rates. We leverage FPI and SPA to develop an iterative approximation. Let $\Lambda_v = \sum_k \Lambda_{vk}$ be the unknown total arrival rate from synchronous stream v. By Little's law, for each stream $v = 1, \ldots, u$ we have

$$s_v = \sum_{k=1}^{n} \Big(\Lambda_{vk}\sigma_v + M_{vk}\theta_{v0} + H_{vk}\theta_{v1} \Big)$$

where M_{vk} and $H_{vk} = \Lambda_{vk} - M_{vk}$ are the miss rate and hit rate for item k requests from stream v. Each term in the right-hand side of the above expression corresponds to the average number of requests either accumulating think think or incurring fetch latency. Denoting with $H_v = \sum_{k=1}^{n} H_{vk}$ the total hit rate, we may rewrite the last equation as

$$\Lambda_v = \frac{s_v}{\sigma_v + (M_v/\Lambda_v)\theta_{v0} + (H_v/\Lambda_v)\theta_{v1}} \tag{30}$$

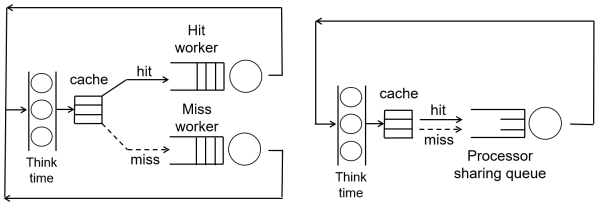
which we may solve by seeking a fixed-point for the iteration

$$\Lambda_v^{(t+1)} = \frac{s_v \Lambda_v^{(t)}}{\Lambda_v^{(t)}\sigma_v + M_v^{(t)}\theta_{v0} + H_v^{(t)}\theta_{v1}} \tag{31}$$

for $t \geq 0$. Using a decomposition approximation, and assuming an initial guess $\Lambda_v^{(0)}$, $v = 1, \ldots, h$, we can readily estimate $M_v^{(0)}$ and $H_v^{(0)}$ by studying the cache in isolation, fed by (asynchronous) Poisson streams with total rates $\Lambda_v^{(0)}$. Solving this isolated model using FPI or SPA, we can then apply (31) to obtain $\Lambda_v^{(1)}$. By iterating this scheme, we can obtain successive approximations for the miss and hit rates $M_v^{(t)}$ and $H_v^{(t)}$ and for the arrival rates $\Lambda_v^{(t)}$, until convergence to a fixed-point.

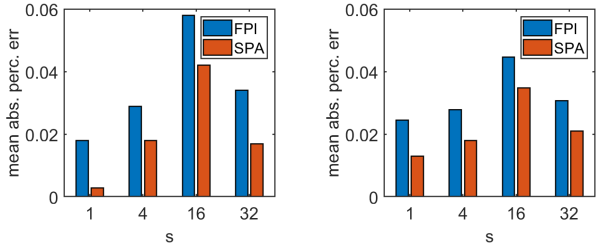
1) Extentions: It is not difficult to generalize the proposed decomposition in two ways. Models processing both synchronous and asynchronous streams require only to include in the isolated cache model additional (asynchronous) Poisson streams. We also support models where requests, after visiting the cache, travel through queueing stations to better characterize the miss (or fetch) latency. This may be due to network transfer or queueing at the worker threads that handle the requests.

Such hybrid caching-queueing models may be seen as multi-chain closed queueing networks with state-dependent routing probabilities, determined by cache hits and misses. A cache can be treated as a topological element with no sojourn time, where given a guess of the arrival rates it is possible to use FPI and



(a) Parallel topology (b) Serial topology

Fig. 8: Examples of models including a cache and queueing stations to model miss and fetch latencies. Routing and class-switching are state-dependent, since they depend on cache hits and cache misses.



(a) Parallel topology (b) Serial topology

Fig. 9: FPI and SPA item miss ratios on the models in Figure 8.

SPA to obtain miss and hit ratios. Miss and hit ratios can then be treated as routing probabilities. This leads naturally to an iterative scheme, where the routing matrix is considered fixed at each iteration, the queueing model solved with approximate MVA [39] obtaining arrival rates at the cache nodes. This yields updated miss and hit ratios, which produce the routing matrix for the next iteration. This iteration rapidly estimates the performance metrics[2].

A. Validation results

1) Cache-only synchronous models: To validate the decomposition approximations, we have first considered synchronous models consisting only of a single cache and think times. We use a set of 810 experiments with the parameters given in Appendix K1 in the supplemental material. Item access patterns follow Zipf distributions. Each simulation collects 10^7 samples. The initial guess for the decomposition method is $\Lambda_v^{(0)} = 1.0$, $\forall v$. Converge criterion is the maximum absolute relative difference between $\Lambda_v^{(t+1)}$ and $\Lambda_v^{(t)}$ falls below 10^{-3}.

Prediction errors for FPI and SPA obtained for FIFO-C($\boldsymbol{m}$) and RR-C($\boldsymbol{m}$) against simulation are shown in Figure 7. The overall mean percentage error Λ_v is 1.64% for FPI and 0.25% for SPA. The approximation error is generally consistent with the trends observed before in the baseline (asynchronous) models. We have verified that the trends as a function of n/m and h are also consistent with previous experiments. We also note that think times do not significantly affect precision.

2) Hybrid caching-queueing models: We now consider the examples in Figure 8(a)-(b), with the parameters reported in Appendix L1 of the supplemental material. Scheduling at queues is first-come first-served in Figure 8(a), processor sharing in Figure 8(b). Each parameterization is evaluated with 216 independent models, each taking 10^7 samples, for a total of 648 simulations. Other experiments are given in Appendix K.

a) Parallel topology: We consider the model in Figure 8(a). The two streams can issue $s_1 = s_2 = s$ synchronous requests each. We use the iterative method with tolerance 10^{-3} on the maximum absolute percentage error on the arrival rates Λ_{iv} at all nodes. The simulation uses 10^7 samples. Results are shown in Figure 9(a). Both FPI and SPA are again accurate, with errors below 4% for FPI and 2% for SPA.

b) Serial topology: We now switch our attention to the model in Figure 8(b). Here, job hits or misses determine the mean service time at the queue. Results are obtained as in the last example and shown in Figure 9(b). Trends are similar, with errors marginally lower than under the serial topology.

IX. Access and Storage Costs

In this section, we show that our results can be used also for cost analysis, which is a topic of broad interest in caching theory [1], [16], [29], [40], [41]. This problem is particularly relevant in the context of non-uniform access, in which a preference may be expressed to store some items in particular lists, based on their cost.

In the proposed models, one may naturally consider two types of costs: access costs and storage costs. Access costs relate to the costs for item movements between lists under non-uniform access patterns, while storage costs are related to holding of particular items in cache. Access costs are easily computed in our framework. Recall that π_{ij} is the steady-state probability that item i is in list j and that $\lambda_{vi}(p(j))c_{vi}(p(j), j)$ is the rate at which stream v triggers a move of item i from list $p(i)$ to $p(i+1)$. This implies that $A_{ij} = \sum_{v=1}^{V} \pi_{ij}\lambda_{vi}(j)c_{vi}(p(j), j)$ is the average rate at which access cost is accumulated for item i access into list j from its parent list $p(j)$. The quantity can be readily computed with FPI, RMF, or SPA by first estimating the probabilities π_{ij} as shown before. Other ways to account probabilistically for access costs that can be used in our models exist in the literature [16].

To determine storage costs, we assume first that each item i incurs a storage cost σ_i upon residing in the cache. For example, σ_i may simply be the size of item i, and the given cost constraint may capture limited storage space. We wish to evaluate the performance of RR-C($\boldsymbol{m}$) when list j cannot exceed a cost cap k_j for its stored items. Formally, we require that the cost cap is respected for each list, i.e., $\sum_{i=1}^{m_j} \sigma_{s(i,j)} \leq k_j$, for $j = 1, \ldots, h$. We indicate with $\boldsymbol{k} = (k_1, \ldots, k_h)$ the vector of cost caps.

We extend RR-C($\boldsymbol{m}$) to account for cost constraints as follows. Initially, we assume the cache to be in a feasible state that respects all the cost caps. Upon receiving a request for item i in list j that would promote it to list j', while simultaneously demoting item i', the policy checks whether the replacement would violate the cost cap for either list j or j'. If this is the case, item i is served but without changing the cache state. This rule applies also to list 0, i.e., a cache miss would serve the object, but not cache it afterwards.

[2]A software tool implementing this approximation may be found at http://line-solver.sf.net.

A. Exact analysis

Let $\mathcal{O} \subset \mathcal{S}$ be the reachable state for the CTMC underlying RR-C($\boldsymbol{m}$). Due to reversibility [42], a restriction of the state space does not break the product-form solution, which remains as in (1) but with normalizing constant

$$E(\boldsymbol{m}, \boldsymbol{k}) = \sum_{\boldsymbol{s} \in \mathcal{O}} \prod_{j=0}^{h} \prod_{i=1}^{m_j} \gamma_{s(i,j)j}$$

It is routine to show that the new normalizing constant satisfies similar properties as before, e.g., (3) is now replaced by

$$E(\boldsymbol{m}, \boldsymbol{k}) = E_k(\boldsymbol{m}, \boldsymbol{k}) + \sum_{j=1}^{h} m_j \gamma_{ij} E_k(\boldsymbol{m} - 1_j, \boldsymbol{k} - \sigma_i 1_j) \quad (32)$$

for arbitrary $i = 1, \ldots, n$, with the usual termination conditions and the requirement that $E(\boldsymbol{m}, \boldsymbol{k}) = 0$ if there exists a $k_j < 0$. The marginal probabilities may be obtained from (32) yielding an expression similar to (2). Similar expression also hold for miss rates and other performance measures. It is also possible to obtain an expression for the average cost K_j for the items in list j at steady-state as

$$K_j = \sum_{i=1}^{n} \sigma_i \pi_{ij} = m_j \sum_{i=1}^{n} \sigma_i \gamma_{ij} \frac{E_i(\boldsymbol{m} - 1_j, \boldsymbol{k} - \sigma_i 1_j)}{E(\boldsymbol{m}, \boldsymbol{k})}$$

We have verified that the above analysis applies similarly under a global cost constraint for the entire cache, in which case $\boldsymbol{k}$ is replaced by a scalar specifying the global cost cap.

The above results hold exactly only for reversible CTMCs, hence they should be treated as approximations in the case of the FIFO-C($\boldsymbol{m}$) policy.

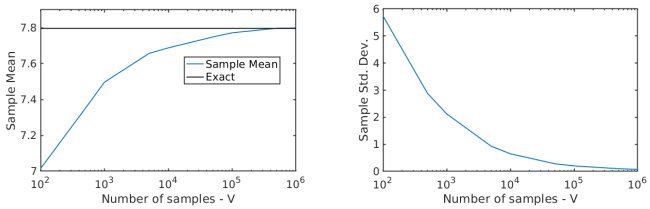
B. Approximate analysis

As before, exact relations are efficient to use only in small caches. Approximations for larger models are quite difficult to develop under cost constraints, since the recursion on $\boldsymbol{k} - \sigma_k 1_j$ introduces dependencies between the object sizes and the structure of the recursion graph. We may alternatively leverage the product-form solution to use Monte Carlo summation to estimate $E(\boldsymbol{m}, \boldsymbol{k})$. This process ensures rapid convergence to accurate estimates since samples are independent [43].

For a list with capacity m, we sample a set of k-permutations of n items, with $k = m$. The v-th sample maps to a random vector $\boldsymbol{S}_v = (s(v,1,1), \ldots, s(v,m_h,h))$, where $s(v,i,j)$ indicates the item in position i of list j. Define $q(\boldsymbol{S}_v) = \prod_{j=0}^{h} \prod_{i=1}^{m_j} \gamma_{s(v,i,j)j}$, and let the sampling distribution be uniform, i.e., $p(\boldsymbol{S}_v) = (n-m)!/n!$. After V samples, we compute the estimator

$$\bar{S}_V = \frac{1}{V} \sum_{v=1}^{V} I\{\boldsymbol{S}_v \in \mathcal{O}\} \frac{q(\boldsymbol{S}_v)}{p(\boldsymbol{S}_v)}$$

where $I\{\cdot\}$ is the indicator function. An unbiased estimator of the normalizing constant is readily given by $E(\boldsymbol{m}, \boldsymbol{k}) = E[\bar{S}_V]$. Confidence intervals may be easily obtained using the Central limit theorem [43].



(a) Mean of estimator (b) Std. dev. of estimator

Fig. 10: Convergence of Monte Carlo estimator for $E(\boldsymbol{m}, \boldsymbol{k})$

1) Example: We illustrate the Monte Carlo summation approach using the tree cache with non-uniform access given in Section II-B (Model 7).The items accessed by Stream 1 have assumed to have size $\sigma_1 = \cdots = \sigma_{n/2} = 1$, with $n = 10$. The remaining items are accessed only by Stream 2 and have sizes $\sigma_{n/2+1} = \cdots = \sigma_n = 2$. The cache has capacity vector $\boldsymbol{m} = (2, 1, 1, 2)$. The cost cap vector is $\boldsymbol{k} = (2, 1, 2, 4)$, which requires in particular that Stream 1 cannot place items in list 2, due to their sizes. The exact value of of the normalizing constant is computed by (32) to be $E(\boldsymbol{m}, \boldsymbol{k}) = 7.7963$, significantly different from the normalizing constant in the corresponding model without cost caps, which is $E(\boldsymbol{m}) = 238.7982$.

Figure 10 illustrates the rapid convergence of the Monte Carlo estimator $E[\bar{S}_V]$ to the true value of the normalizing constant. The curve in Figure 10(a) is obtained by averaging the value of $E[\bar{S}_V]$ over 100 independent repetitions, each with the value of V shown in the horizontal axis. The associated standard deviation is shown in Figure 10(b), indicating that with as little as $V = 10^4$ samples the approximation error of a single estimate of $E[\bar{S}_V]$ is typically below 15%. The result also cross-validates the correctness of (32), which we have also verified using the direct computation from the definition.

X. Conclusion

We have generalized models for list-based caches to include non-uniform access. After solving the Markov process for the RR-C($\boldsymbol{m}$) and FIFO-C($\boldsymbol{m}$) policies, we have developed exact and approximate formulas to efficiently analyze performance measures such as miss rates. We have then applied singular perturbation methods and a refined mean-field approximation to determine the asymptotic behavior of the cache in transient and steady-state regimes. Simulation results indicate that our approximations typically incur less than 1.5% error on large models. We have further proposed to use this result to approximate models with synchronous requests, fetch latencies, and item costs or sizes, obtaining similar levels of accuracy.

References

[1] S. Podlipnig and L. Boszormenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys*, vol. 35, 2003.

[2] M. M. Amble, P. Parag, S. Shakkottai, and L. Ying, "Content-aware caching and traffic management in content distribution networks," in *Proc. INFOCOM*, pp. 2858–2866, 2011.

[3] N. Carlsson and D. L. Eager, "Ephemeral content popularity at the edge and implications for on-demand caching," *IEEE TPDS*, vol. 28, no. 6, pp. 1621–1634, 2017.

[4] O. Saleh and M. Hefeeda, "Modeling and caching of peer-to-peer traffic," in *Proc. ICNP*, pp. 249–258, IEEE, 2006.

[5] R. Fagin, "Asymptotic miss ratios over independent references," *J. Comp. and Sys. Sci.*, vol. 14, pp. 222–250, Apr. 1977.

[6] R. Fagin and T. G. Price, "Efficient calculation of expected miss ratios in the independent reference model," *SIAM Journal on Computing*, vol. 7, 1978.

[7] E. Gelenbe, "A unified approach to the evaluation of a class of replacement algorithms," *IEEE Trans. Computers*, vol. 22, no. 6, pp. 611–618, 1973.

[8] O. I. Aven, E. G. Coffman, and Y. A. Kogan, *Stochastic analysis of computer storage*. Reidel, 1987.

[9] J. V. D. Berg and A. Gandolfi, "LRU is better than FIFO under the independent reference model," *J. App. Prob.*, vol. 29, no. 1, pp. 239–243, 1992.

[10] K. Psounis and B. Prabhakar, "Efficient randomized web-cache replacement schemes using samples from past eviction times," *IEEE/ACM Trans. Netw*, vol. 10, no. 4, pp. 441–455, 2002.

[11] N. Tsukada, R. Hirade, and N. Miyoshi, "Fluid limit analysis of FIFO and RR caching for independent reference models," *PEVA*, vol. 69, no. 9, pp. 403–412, 2012.

[12] J. Li, S. Shakkottai, J. C. S. Lui, and V. Subramanian, "Accurate learning or fast mixing? Dynamic adaptability of caching algorithms," *CoRR*, vol. abs/1701.02214, 2017.

[13] N. Carlsson and D. L. Eager, "Worst-case bounds and optimized cache on mth request cache insertion policies under elastic conditions," *PEVA*, 2018.

[14] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *Proc. of INFOCOM*, pp. 2040–2048, 2014.

[15] N. Gast and B. V. Houdt, "Transient and steady-state regime of a family of list-based cache replacement algorithms," in *Proc. SIGMETRICS*, pp. 123–136, ACM, 2015.

[16] D. Starobinski and D. Tse, "Probabilistic methods in web caching," *PEVA*, vol. 46, pp. 125–137, Oct. 2001.

[17] A. Aho, P. Denning, and J. Ullman, "Principles of optimal page replacement," *JACM*, vol. 18, 1971.

[18] W. F. King, "Analysis of demand paging algorithms," in *IFIP Congress (1)*, pp. 485–490, 1971.

[19] N. Gast and B. V. Houdt, "Asymptotically exact TTL-approximations of the cache replacement algorithms LRU(m) and h-LRU," in *Proc. ITC*, pp. 157–165, IEEE, 2016.

[20] C. Knessl and C. Tier, "Asymptotic expansions for large closed queueing networks with multiple job classes," *IEEE Trans. Computers*, vol. 41, no. 4, pp. 480–488, 1992.

[21] N. Gast and B. V. Houdt, "A refined mean field approximation," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, p. 33, 2017.

[22] G. Casale, "Analyzing replacement policies in list-based caches with non-uniform access costs," in *Proc. INFOCOM*, pp. 432–440, IEEE, 2018.

[23] D. S. Berger, P. Gland, S. Singla, and F. Ciucu, "Exact analysis of TTL cache networks," *Perform. Eval*, vol. 79, pp. 2–23, 2014.

[24] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: modeling, design and experimental results," *IEEE JSAC*, vol. 20, no. 7, pp. 1305–1314, 2002.

[25] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proc. ITC*, pp. 1–8, 2012.

[26] A. Dan and D. F. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proc. SIGMETRICS*, pp. 143–152, ACM, 1990.

[27] M. Gallo, B. Kauffmann, L. Muscariello, A. Simonian, and C. Tanguy, "Performance evaluation of the random replacement policy for networks of caches," *PEVA*, vol. 72, pp. 16–36, 2014.

[28] G. Ju, Y. Li, Y. Xu, J. Chen, and J. C. S. Lui, "Stochastic modeling of hybrid cache systems," in *Proc. MASCOTS*, pp. 69–78, IEEE, 2016.

[29] O. Bahat and A. M. Makowski, "Optimal replacement policies for non-uniform cache objects with optional eviction," in *Proc. INFOCOM*, pp. 427–437, IEEE, 2003.

[30] S. Lam, "Dynamic scaling and growth behavior of queueing network normalization constants," *JACM*, vol. 29, no. 2, pp. 492–513, 1982.

[31] M. Reiser and S. S. Lavenberg, "Mean-value analysis of closed multichain queueing networks," *JACM*, vol. 27, no. 2, pp. 312–322, 1980.

[32] J. Zahorjan, "The distribution of network states during residence times in product form queueing networks," *Perform. Eval*, vol. 4, no. 2, pp. 99–104, 1984.

[33] C. Knessl and C. Tier, "Applications of singular perturbation methods in queueing," in *Advances in queueing theory, methods, and open problems, probability and stochastics series*, pp. 311–36, 1995.

[34] E. W. Weisstein, *CRC Concise Encyclopedia of Mathematics*. Chapman and Hall, 2003.

[35] C. Knessl and C. Tier, "Applications of singular perturbation methods in queueing," in *Frontiers in Queueing: Models and Applications in Science and Engineering*, pp. 331–336, CRC Press, 1996.

[36] T. G. Kurtz, "Limit theorems for sequences of jump markov processes approximating ordinary differential processes," *Journal of Applied Probability*, vol. 8, no. 2, pp. 344–356, 1971.

[37] N. Gast, "Size expansions of mean field approximation: Transient and steady-state analysis," *Performance Evaluation*, 2018.

[38] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of YouTube network traffic at a campus network - Measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501–514, 2009.

[39] *et al.*. G. Bolch, *Queueing Networks and Markov Chains*. Wiley, 2006.

[40] G. Neglia, D. Carra, M. Feng, V. Janardhan, P. Michiardi, and D. Tsigkari, "Access-time aware cache algorithms," in *Proc. of ITC*, pp. 148–156, IEEE, 2016.

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

[42] F. P. Kelly, *Reversibility and Stochastic Networks*. New York: John Wiley & Sons, 1979.

[43] K.W. Ross, D.H.K. Tsang and J. Wang, "Monte carlo summation and integration applied to multiclass queueing networks.," *JACM*, vol. 41, no. 6, pp. 1110–1135, 1994.

[44] R. Courant and D. Hilbert, *Methods of Mathematical Physics, Volume II*. Interscience, 1962.

[45] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*. CUP, 1952.

[46] N. Gast, "Expected values estimated via mean-field approximation are 1/n-accurate," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 1, p. 17, 2017.

[47] N. Gast, "Construction of lyapunov functions via relative entropy with application to caching," *ACM SIGMETRICS PER*, vol. 44, no. 2, pp. 6–8, 2016.

APPENDIX

A. Proof of product-form solution

Compared to the proof techniques used in prior work [15], [28], which are based on the global balance equations of the underlying Markov process, our proof gives for RR-C($\boldsymbol{m}$) an argument based on Kolmogorov's criterion [42].

1) RR-C($\boldsymbol{m}$) policy: Under RR-C($\boldsymbol{m}$), consider a state $\boldsymbol{s}_t \in \mathcal{S}$, and let $\boldsymbol{s}_{t+1} \in \mathcal{S}$ be obtained from $\boldsymbol{s}_t$ after completing a request that promotes item k from list j to list j', simultaneously demoting i from list j' to list j. The transition rates for RR-C($\boldsymbol{m}$) are then $q(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) = \sum_{v=1}^{u} \lambda_{vk}(j) c_{vk}(j, j')/m_{j'}$ and $q(\boldsymbol{s}_{t+1}, \boldsymbol{s}_t) = \sum_{v=1}^{u} \lambda_{vi}(j) c_{vi}(j, j')/m_{j'}$. The same holds also for cache misses, with items that reside outside the cache being in list $j = 0$. With the definitions in Theorem 1, we have

$$\pi(\boldsymbol{s}_t) q(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) = \alpha \gamma_{kj} \gamma_{ij'} \sum_{v=1}^{u} \lambda_{vk}(j) c_{vk}(j, j') \qquad (33)$$

where $\alpha = (1/m'_j) \prod_{l \neq j \neq j'} \prod_{k=1}^{m_l} \gamma_{s(k,l)l}$. Similarly,

$$\pi(\boldsymbol{s}_{t+1}) q(\boldsymbol{s}_{t+1}, \boldsymbol{s}_t) = \alpha \gamma_{ij} \gamma_{kj'} \sum_{v=1}^{u} \lambda_{vi}(j) c_{vi}(j, j') \qquad (34)$$

If both left-hand sides of (33) and (34) are strictly positive, then they must be equal since $\gamma_{ij'}/\gamma_{ij}$ and $\gamma_{kj'}/\gamma_{kj}$ match the values of the two summations by the definitions in Theorem 1. For the same reasons, if any of the factors in zero, then both left-hand sides of (33) and (34) are zero, e.g., if the summation

in (33) is zero, then $\gamma_{kj'}$ in (34) is zero as it also includes this sum in its definition.

Note that the last passages are a consequence of the fact that the parent relationship are stream-independent, as otherwise there could be two streams v and v' such that item k after a request from stream v' reaches list j' from list $l \neq j \neq j'$ and without visiting list j. In this case $c_{v'k}(l,j')\lambda_{v'k}(l)$ would not be present within the summation in (33), whereas it would need to be accounted for within $\gamma_{kj'}$, as this describes a position for k that is reachable also via the promotion from l.

Applying recursively the last result, it follows that the product-form (1) for any feasible sequence of states $\boldsymbol{s}_0 \to \boldsymbol{s}_1 \to \cdots \to \boldsymbol{s}_S$ satisfies Kolmogorov's criterion [42] $\pi(\boldsymbol{s}_0)\prod_{t=1}^{S} q(\boldsymbol{s}_{t-1},\boldsymbol{s}_t) = \pi(\boldsymbol{s}_S)\prod_{t=1}^{S} q(\boldsymbol{s}_t,\boldsymbol{s}_{t-1})$. This is a necessary and sufficient condition for the Markov process to be reversible and for (1) to be its equilibrium distribution [42].

2) FIFO-C($\boldsymbol{m}$) policy: In the case of FIFO-C($\boldsymbol{m}$), the process is not reversible. However, it is sufficient to plug (1) in the global balance equations to verify the statement. The full derivation is given in Appendix D in the online supplement.

B. Proof of Theorem 2

It is possible to verify with a little algebra that

$$\frac{\partial E(\boldsymbol{m})}{\partial \gamma_{kl}} = m_l E_k(\boldsymbol{m}-1_l) = \frac{\pi_{kl}(\boldsymbol{m})}{\gamma_{kl}} E(\boldsymbol{m}). \tag{35}$$

Applying this relationship to (3) and using (5) we find

$$\pi_{kl}(\boldsymbol{m}) = m_l \gamma_{kl} \frac{E_k(\boldsymbol{m}-1_l)}{E(\boldsymbol{m})} = \gamma_{kl}\xi_l(\boldsymbol{m}) \frac{E_k(\boldsymbol{m}-1_l)}{E(\boldsymbol{m}-1_l)}$$

and the result follows since by definition

$$\pi_{k0}(\boldsymbol{m}) = \frac{E_k(\boldsymbol{m})}{E(\boldsymbol{m})} = 1 - \pi_k(\boldsymbol{m}) \tag{36}$$

C. Proof of normalizing constant asymptotic expansion

1) Derivation of $\phi(\boldsymbol{x}, y)$: Expanding (12) in powers of ε, we obtain at the lowest order $1 = e^{-\phi_y} + \sum_j g_j(y) e^{-\phi_y - \phi_j}$, where ϕ_j (resp. ϕ_y) denotes partial differentiation with respect to x_j (resp. y). Multiplying both sides by e^{ϕ_y} and simplifying yields $1 - e^{\phi_y} + \sum_{j=1}^{h} g_j(y) e^{-\phi_j} = 0$. Except for the sign in front of ϕ_y, this is the eikonal equation studied in [20]. The method of characteristics for non-linear PDEs [44] then yields

$$dy = -e^{\phi_y} dt \tag{37a}$$
$$dx_j = -g_j(y) e^{-\phi_j} dt \tag{37b}$$
$$d\phi = \phi_y dy + \textstyle\sum_j \phi_j dx_j \tag{37c}$$
$$d\phi_y = -\textstyle\sum_j g_j'(y) e^{-\phi_j} dt \tag{37d}$$
$$d\phi_j = 0 \tag{37e}$$

where all variables are intended as functions of $(t, \boldsymbol{\xi})$, with t being the integration variable over the characteristic curves and let $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_h)$ parameterize the family of curves. To solve this system, we follow a strategy similar to the one used in [20] to analyze normalizing constants in queueing networks; our strategy differs mainly at the boundary condition.

We first solve (37e) as $\phi_j = \log \xi_j$, where we have set the integration constant to $\log \xi_j$, $j = 1, \ldots, h$. Multiplying (37d) by (37a) after dividing both by dt, and integrating we find under the initial conditions $\phi = 0$, $\boldsymbol{x} = 0$, and $y = 0$ if $t = 0$, that $\phi_y = \log(1 + \sum_j g_j(y)\xi_j)$, where intermediate integration constants need to be set to unity for consistency with the eikonal equation. Plugging the last result in (37a) and integrating we obtain t, after which (37b) yields (14). Using the above results, (37c) integrates to (13).

2) Derivation of $B(\boldsymbol{x}, y)$: Expanding (12) and matching first-order terms, we write after simplifications $\frac{dB}{dt} = \frac{B}{2}(\phi_{yy} e^{\phi_y} + \sum_j g_j(y)(\phi_{jj} + 2\phi_{jy}) e^{-\phi_j})$. Plugging (13), this becomes similar to the transport equation studied in [20]. Similar passages as in [20] yield $B(\boldsymbol{x}, y) = b(\xi_1, \ldots, \xi_h)(\det \boldsymbol{J})^{-1/2}(1 + \sum_j g_j(y)\xi_j)^{-1/2}$, in which $b(\xi_1, \ldots, \xi_h)$ is an arbitrary function, the matrix $\boldsymbol{J}$ is given in (16), and the ξ_j's depend on $\boldsymbol{x}$ and y via (14).

3) Boundary condition: At the boundary, we choose to match the arbitrary function $b(\xi_1, \ldots, \xi_h)$ to the value of $H(\boldsymbol{m})$ when $\gamma_{kj} = g_j(k) = \gamma_j$, $\forall k, j$, for arbitrary γ_j. This is simply given by (10). For small $\boldsymbol{x}$ and y, we note that (14) gives $x_j = \xi_j \gamma_j y (1 + \sum_{l=1}^{h} \gamma_l \xi_l)^{-1}$, implying $\xi_j = x_j (y - x)^{-1} \gamma_j^{-1}$ upon summing over j. Substituting the expression of ξ_j in (13) it is possible to verify that ϕ_0 is matched. We then require

$$\frac{b(\xi_1, \ldots, \xi_h)}{\sqrt{\det \boldsymbol{J}}\sqrt{1 + \sum_j \gamma_j \xi_j}} = \frac{(2\pi)^{-h/2}}{\sqrt{\prod_j x_j}} \sqrt{\frac{y}{y - x}}$$

By definition of J_{ik}, we find $J_{ik} = (y - x)(\delta_{ik}\gamma_i - \frac{x_i}{y}\gamma_k)$. Using this result we compute $\det \boldsymbol{J}$ and determine with a little algebra the required value of $b(\xi_1, \ldots, \xi_h)$, which yields (15). Lastly, to match (10), we need to set $D(\varepsilon) = \varepsilon^{h/2}$.

Giuliano Casale Giuliano Casale joined the Department of Computing at Imperial College London in 2010, where he is currently a Reader in Modelling and Simulation. Previously, he worked at SAP Research UK and in the capacity planning industry. He teaches and does research in performance engineering, cloud computing, and Big data, topics on which he has published more than 100 refereed papers. He has served on the technical program committee of over 80 conferences and workshops and as co-chair for several conferences such as ACM SIGMETRICS/Performance and IEEE/IFIP DSN. His research is recipient of multiple awards, recently the best paper award at ACM SIGMETRICS 2017. He serves on the editorial boards of IEEE TNSM and ACM TOMPECS and as chair of ACM SIGMETRICS.

Nicolas Gast Nicolas Gast is a tenured research scientist at Inria (Grenoble, France) since 2014. He graduated from Ecole Normale Superieure (Paris, France) in 2007 and received a Ph.D. from the University of Grenoble in 2010. He was a research fellow at EPFL from 2010 to 2014. His research focuses on the development and the use of stochastic models and optimization methods for the design of control algorithms in large-scale systems. He works on the control of energy networks.

SUPPLEMENTAL MATERIAL

D. Proof of product-form solution for FIFO-C($\boldsymbol{m}$)

The global balance equations are

$$\left(\sum_{j=0}^{h-1}\sum_{\substack{j'=1..h\\j'\neq j}}\sum_{k=1}^{m_j}\left(\sum_{v=1}^{u}\lambda_{vs(k,j)}(j)c_{vs(k,j)}(j,j')\right)\right)\pi(\boldsymbol{s}) = \sum_{j=0}^{h-1}\sum_{\substack{j'=1..h\\j'\neq j}}\sum_{k=1}^{m_j}\left(\sum_{v=1}^{u}\lambda_{vs(1,j')}(j)c_{vs(1,j')}(j,j')\right)\pi(\boldsymbol{s}_{kjj'}) \tag{38}$$

where $\boldsymbol{s}$ is reachable from the initial state and where $s \equiv s(i,j)$ and $\boldsymbol{s}$ is obtained from $\boldsymbol{s}_{kjj'} \in \mathcal{S}$ by promoting item $s(1,j')$, assumed in $\boldsymbol{s}_{kjj'}$ at position k of list j, to the front of list $j' \neq j$. We now note that the innermost brackets in the global balance equations if positive may be seen as ratios of access factors, e.g., if all the terms are positive

$$\sum_{j=0}^{h-1}\sum_{\substack{j'=1..h\\j'\neq j}}\sum_{k=1}^{m_j}\frac{\gamma_{s(k,j)j'}}{\gamma_{s(k,j)j}}\pi(\boldsymbol{s}) = \sum_{j=0}^{h-1}\sum_{\substack{j'=1..h\\j'\neq j}}\sum_{k=1}^{m_j}\frac{\gamma_{s(1,j')j'}}{\gamma_{s(1,j')j}}\pi(\boldsymbol{s}_{kjj'}) \tag{39}$$

We also observe that the product-form in (1) satisfies

$$\pi(\boldsymbol{s}_{kjj'}) = \frac{\gamma_{s(1,j')j}\gamma_{s(k,j)j'}}{\gamma_{s(1,j')j'}\gamma_{s(k,j)j}}\pi(\boldsymbol{s}) \tag{40}$$

Plugging (40) into (39) the expression reduces to an identity. This holds also if some of the innermost brackets in (38) are zero, since the terms are neglected from the balance equation.

E. Additional recurrence relations

We develop here additional recurrence relations for the normalizing constant $E(\boldsymbol{m})$ and for the quantities $\xi_l(\boldsymbol{m})$.

1) $O(n^m)$ recurrence relation for $E(\boldsymbol{m})$: Plugging (2) into the condition $\sum_i \pi_{il}(\boldsymbol{m}) = m_l$ yields the recurrence relation

$$E(\boldsymbol{m}) = \sum_{i=1}^{n}\gamma_{il}E_i(\boldsymbol{m}-1_l) \tag{41}$$

for all $l = 1,\ldots,h$. The right-hand side may be seen as summing over all the (un-normalized) conditional probabilities that item i is in position m_l in list l. Further, summing (3) over $k = 1,\ldots,n$ we get

$$\sum_{k=1}^{n}E(\boldsymbol{m}) = \sum_{k=1}^{n}E_k(\boldsymbol{m}) + \sum_{j=1}^{h}m_j\sum_{k=1}^{n}\gamma_{kj}E_k(\boldsymbol{m}-1_j)$$

Finally, using (41) on the last summation yields

$$(n-m)E(\boldsymbol{m}) = \sum_{i=1}^{n}E_i(\boldsymbol{m}) \tag{42}$$

2) Recurrence relation for $\xi_l(\boldsymbol{m})$: Let us observe that $\xi_l(\boldsymbol{m})$ admits the exact recurrence relation

$$\xi_l(\boldsymbol{m}) = \frac{1+\sum_{\substack{j=1\\j\neq l}}^{h}\gamma_{kj}\xi_j^{-k}(\boldsymbol{m}-1_l)+\gamma_{kl}\xi_l^{-k}(\boldsymbol{m}-1_l)}{\frac{I\{n>m\}}{\xi_l^{-k}(\boldsymbol{m})}+\sum_{\substack{j=1\\j\neq l}}^{h}\gamma_{kj}\frac{\xi_j^{-k}(\boldsymbol{m}-1_l)}{\xi_l^{-k}(\boldsymbol{m}-1_j)}+\gamma_{kl}} \tag{43}$$

for arbitrary $k = 1,\ldots,n$, where I is the indicator function. Relation (43) is a non-uniform access extension of known results for the function $F_l(\boldsymbol{m}) == m_l/\xi_l(\boldsymbol{m})$ [6, Eq. 2.7] and [15, Thm. 2] in uniform access models. The proof of (43) follows similarly to earlier work, that is plugging (3) into (5) we get

$$\xi_l(\boldsymbol{m}) = \frac{E_k(\boldsymbol{m}-1_l)+\sum_{j=1}^{h}(m_j-\delta_{jl})\gamma_{kj}E_k(\boldsymbol{m}-1_j-1_l)}{E_k(\boldsymbol{m})+\sum_{j=1}^{h}m_j\gamma_{kj}E_k(\boldsymbol{m}-1_j)} \tag{44}$$

The result is then obtained with simple passages by dividing both numerator and denominator by $E_k(\boldsymbol{m})$ and using that

$$\frac{T_l^{-k}(\boldsymbol{m})}{T_j^{-k}(\boldsymbol{m}-1_l)} = \frac{E_k(\boldsymbol{m}-1_j-1_l)}{E_k(\boldsymbol{m}-1_l)}\frac{E_k(\boldsymbol{m}-1_l)}{E_k(\boldsymbol{m})}$$

Using (43), we can compute the miss rate in $\mathcal{O}(hn\prod_{j=1}^{h}(m_h+1))$. The recursion has boundary conditions: i) $\xi_l(\boldsymbol{m}) = +\infty$ if $m > n$ or $\min_l m_l < 0$; ii) $\xi_l(\boldsymbol{0}) = 0$; iii) $\xi_l(1_l) = \gamma_{k,l}$ if $m_k = 1$ and $n = 1$; iv) $\xi_l(\boldsymbol{m}) = 0$ if $m_l = 0$.

F. Proof of Theorem 4

The proof relies on the following two lemmas, proven later in Appendix G and Appendix H.

Lemma 1. *$\xi_j(\boldsymbol{m}-1_l) \leq \xi_j(\boldsymbol{m}) \leq \xi_j^{-k}(\boldsymbol{m})$, $\forall j,l,k$.*

Lemma 2. *$\pi_{k0}(\boldsymbol{m}) \leq \pi_{k0}(\boldsymbol{m}-1_l)$, $\forall k,l$.*

We obtain the bound by summing (7) over all l, and then by Lemma 2 replacing $\pi_i(\boldsymbol{m}-1_l)$ with its upper bound $\pi_i(\boldsymbol{m})$. This yields $\pi_i(\boldsymbol{m}) \geq \sum_{l=1}^{h}\gamma_{il}\xi_l(\boldsymbol{m})(1-\pi_i(\boldsymbol{m}))$. The proof follows after solving for $\pi_i(\boldsymbol{m})$ and using the result to compute $\pi_{i0}(\boldsymbol{m})$.

G. Proof of Lemma 1

The proof is by induction on the number of items n and depends on results in Appendix E2, in particular on the recurrence relation

$$\xi_l(\boldsymbol{m}) = \frac{1+\sum_{\substack{j=1\\j\neq l}}^{h}\gamma_{kj}\xi_j^{-k}(\boldsymbol{m}-1_l)+\gamma_{kl}\xi_l^{-k}(\boldsymbol{m}-1_l)}{\frac{I\{n>m\}}{\xi_l^{-k}(\boldsymbol{m})}+\sum_{\substack{j=1\\j\neq l}}^{h}\gamma_{kj}\frac{\xi_j^{-k}(\boldsymbol{m}-1_l)}{\xi_l^{-k}(\boldsymbol{m}-1_j)}+\gamma_{kl}} \tag{45}$$

where $I\{n>m\} = 1$ if and only if $n > m$.

We now prove the result.

Case $n = m+1$. The case holds since by (5) it is $\xi_j^{-i}(\boldsymbol{m}-1_l) = 0$, being $E(\boldsymbol{m}-1_l-1_j) = 0$.

Induction step. Let the model with $n-1$ items exclude item i, we focus on the hypothesis $\xi_j^{-i}(\boldsymbol{m}-1_l) \leq \xi_j^{-i}(\boldsymbol{m})$, $\forall j,l$. We choose $k = i$ in (45) and divide both sides by $\xi_l^{-i}(\boldsymbol{m})$ so that

$$\frac{\xi_l(\boldsymbol{m})}{\xi_l^{-i}(\boldsymbol{m})} = \frac{1+\sum_{\substack{j=1\\j\neq l}}^{h}\gamma_{ij}\xi_j^{-i}(\boldsymbol{m}-1_l)+\gamma_{il}\xi_l^{-i}(\boldsymbol{m}-1_l)}{1+\sum_{\substack{j=1\\j\neq l}}^{h}\gamma_{ij}\xi_j^{-i}(\boldsymbol{m})+\gamma_{il}\xi_l^{-i}(\boldsymbol{m})}$$

since $I\{n>m\} = 1$ and by (5) we note that $\xi_j^{-i}(\boldsymbol{m})/\xi_l^{-i}(\boldsymbol{m}) = E_j(\boldsymbol{m}-1_l)/E_i(\boldsymbol{m}-1_l) = \xi_j^{-i}(\boldsymbol{m}-1_l)/\xi_l^{-i}(\boldsymbol{m}-1_j)$. If we compare term-by-term numerator and denominator in the above expression,

by the induction hypothesis we obtain the upper bound $\xi_l(\boldsymbol{m}) \leq \xi_l^{-i}(\boldsymbol{m})$. Plugging (5) in the last inequality, we get $E(\boldsymbol{m}-1_l)E_i(\boldsymbol{m}) \leq E_i(\boldsymbol{m}-1_l)E(\boldsymbol{m})$. Diving both sides by $E(\boldsymbol{m}-1_l)E(\boldsymbol{m})$, by (36) we get $\pi_i(\boldsymbol{m}) \geq \pi_i(\boldsymbol{m}-1_l)$. The last relationship implies in particular $\pi_i(\boldsymbol{m}-1_j) \geq \pi_i(\boldsymbol{m}-1_l-1_j)$, thus by (6) we find the other bound $\xi_j(\boldsymbol{m}-1_l) \leq m_j\big(\sum_{i=1}^n \gamma_{il}(1-\pi_i(\boldsymbol{m}-1_j))\big)^{-1} = \xi_j(\boldsymbol{m})$.

H. Proof of Lemma 2

We give the proof for $\pi_i(\boldsymbol{m}) = 1-\pi_{i0}(\boldsymbol{m})$ by showing that $\pi_i(\boldsymbol{m}) \geq \pi_i(\boldsymbol{m}-1_l)$, for any l. By (36), we need show that $E(\boldsymbol{m}-1_l)E_i(\boldsymbol{m}) \leq E_i(\boldsymbol{m}-1_l)E(\boldsymbol{m})$. Plugging (3) in $E(\boldsymbol{m}-1_l)$ and $E(\boldsymbol{m})$, with the choice $k=i$, after simplifications it is sufficient to show that $E_i(\boldsymbol{m}-1_l-1_j)E_i(\boldsymbol{m}) \leq E_i(\boldsymbol{m}-1_l)E_i(\boldsymbol{m}-1_j)$, for all $j \neq l$. Dividing by $E_i(\boldsymbol{m})E_i(\boldsymbol{m}-1_j)$ this may be seen as stating $\xi_j^{-i}(\boldsymbol{m}-1_l) \leq \xi_j^{-i}(\boldsymbol{m})$, which holds after applying Lemma 1 to a model without item i.

I. Proof of Theorem 3

To prove the lower bound, we show that the assumption implies $\pi_i(\boldsymbol{m}) \geq \pi_k(\boldsymbol{m})$, $\forall \boldsymbol{m}$, so that the statement follows by applying Chebyshev's sum inequality [45] to the denominator of (6). Using (36), we need to show that $E_k(\boldsymbol{m}) \geq E_i(\boldsymbol{m})$ for all i, $k > i$. Note that $E_k(\boldsymbol{m})$ is obtained from $E_i(\boldsymbol{m})$ by replacing γ_{kl} in $E_i(\boldsymbol{m})$ with $\gamma_{il} \geq \gamma_{kl}$. The statement then holds since the normalizing constant is by (35) non-decreasing with the access factors.

J. Derivation of the refined mean-field approximation (RMF)

In this section, we discuss in detail how to compute the refined mean-field approximation.

The refined mean field is obtained by considering a scaled system with N identical copies of each item and by multiplying all list sizes by N. Recall that $X_{il}^{(N)}(t)$ the fraction of class-i items that are in list l at time t, which implies that $NX_{il}^{(N)}(t)$ is the number of class-i items that are in list l at time t.

To simplify the notations, let $\rho_{ill'} = \sum_{v=1}^u \lambda_{v,k}(l)c_{vk}(l,l')$ be the rate at which a given item k is promoted from list l to list l'. At rate $\rho_{ill'}NX_{il}$, a class-i item will be promoted to list l'. It will be exchanged with an item from list l'. In such a list, there are $NX_{jl'}$ items of type j among the $Nm_{l'}$ items. Combining the two, this means that a class i items of state l is exchanged with a class j item of list l' at rate $N\rho_{ill'}NX_{il}X_{jl}/m_{l'}$. Such a transition will modify the state $X^{(N)}$ into

$$X^{(N)} + \frac{1}{N}\left(-\mathbf{e}_{il} + \mathbf{e}_{il'} + \mathbf{e}_{jl} - \mathbf{e}_{jl'}\right), \tag{46}$$

where $\mathbf{e}_{il}$ is a vector whose only non-zero coordinate is the (il) coordinate.

This shows that the process is a density dependent population process as defined in [36]. The case $N=1$ corresponds to our original caching model. According to [46], there exists a deterministic dynamical system $x(t) = (x_{il}(t))$ such that for all item i and list l: $X_{il}(t) = x_{il}(t) + O(1/N)$, where x satisfies the differential equation $\dot{x} = f(x)$ with

$$f(x) = \sum_{il,jl'} \left(-\mathbf{e}_{il} + \mathbf{e}_{il'} + \mathbf{e}_{jl} - \mathbf{e}_{jl'}\right) x_{il}x_{jl}/m_{l'}. \tag{47}$$

In particular, the (il) coordinate of f is

$$f_{il}(x) = \sum_{l'} \Bigg[\rho_{il'l}x_{il'} - \rho_{ill'}x_{il} + \frac{x_{il'}}{m_{l'}}\sum_j \rho_{jll'}x_{jl} - \frac{x_{il}}{m_l}\sum_j \rho_{jl'l}x_{jl'}\Bigg]. \tag{48}$$

In the above equation, the first line correspond to items that are promoted to list l from list l' (the term $\rho_{il'l}x_{il'}$) or promoted from list l to list l' (the term $\rho_{ill'}x_{il}$). The second line correspond to class i items that are demoted from list l' to list l (here $\sum_j \rho_{jl,l'}x_{jl}$ is the rate at which any item is promoted from list l to list l' and $\frac{x_{il'}}{m_{l'}}$ is the probability that a class i item is exchanged with it) or demoted from list l to list l' ($\frac{x_{il}}{m_l}\sum_j \rho_{jl'l}x_{jl'}$).

It is straightforward to verify that $\hat{\pi}^{FPI}$ satisfies $f(\hat{\pi}^{FPI}) = 0$, *i.e.*, it is a fixed point of the differential equation $\dot{x} = f(x)$. Following [47], it can be shown that this fixed point is unique and is a global attractor of all the trajectories of the differential equation. It is shown in [21], that this implies that there exists a vector V such that, in steady-state,

$$\mathbb{E}[X_{il}] = \hat{\pi}_{il}^{FPI}(t) + \frac{1}{N}V_{il} + O(1/N^2).$$

Note that in the above equation $\mathbb{E}[X_{il}(t)]$ is equal to the steady-state probability that an item i is in list l.

The above vector V_{il} can be computed by solving the following linear system that depends on the first (Jacobian) and second derivative of the drift evaluated in $\hat{\pi}^{FPI}$:

$$\forall il, jl' : \sum_{kl''}\Big(A_{il,kl''}W_{kl'',jl'} + W_{il,kl''}A_{kl'',jl}\Big) + Q_{il,jl'} = 0.$$

$$\forall il : \sum_{jl'} A_{il,,jl'}V_{jl'} + \frac{1}{2}\sum_{jl',kl''} B_{il,jl',kl''}W_{jl',kl''} = 0,$$

In the above equation, $A_{il,jl'}$ is the derivative of $f_{il}(x)$ with respect to $x_{jl'}$, $B_{il,jl',kl''}$ the second derivative of $f_{il}(x)$ with respect to $x_{jl'}$ and $x_{kl''}$ and Q is a variance term whose expression is given later. The terms could be computed using symbolic analysis tools such a rmf-tool[3].

The above equations form a system of linear equations with $nh + (nh)^2$ unknowns (nh unknowns for V and $(nh)^2$ for W). This system can be solved by first computing W and then computing V. Note that the equation for W is a Lyapunov equation that can be solved by using appropriate algorithms such as the ones implemented in the function `scipy.linalg.solve_lyapunov` of the library `scipy`.

To compute the derivatives of f, the simplest numerical procedure is to use the definition of the drift of Equation (47).

[3]https://github.com/ngast/rmf_tool/

Denoting $e_{il,kl''}$ a matrix of size $(nh \times nh)$ whose only non-zero element is the (il, kl'') element, the derivative of f with respect to $x_{kl''}$ is the vector $A_{.,kl''}$:

$$\begin{aligned}
A_{.,kl''} &= \sum_{il,jl'} \left(-\mathbf{e}_{il} + \mathbf{e}_{il'} + \mathbf{e}_{jl} - \mathbf{e}_{jl'}\right) \frac{\partial(x_{il}x_{jl})}{\partial x_{kl''}} \frac{\rho_{ill'}}{m_{l'}} \\
&= \sum_{il,jl'} \left(-\mathbf{e}_{il} + \mathbf{e}_{il'} + \mathbf{e}_{jl} - \mathbf{e}_{jl'}\right) \left(x_{jl'}\mathbf{1}_{il=kl''} + x_{il}\mathbf{1}_{jl=kl''}\right) \frac{\rho_{ill'}}{m_{l'}} \\
&= \sum_{jl'} \left(-\mathbf{e}_{kl''} + \mathbf{e}_{kl'} + \mathbf{e}_{jl''} - \mathbf{e}_{jl'}\right) \frac{\rho_{kl''l'}x_{jl'}}{m_{l'}} \\
&\quad + \sum_{il} \left(-\mathbf{e}_{il} + \mathbf{e}_{il''} + \mathbf{e}_{kl} - \mathbf{e}_{kl''}\right) \frac{\rho_{ill'}x_{il}}{m_{l''}}.
\end{aligned}$$

Note that $\frac{\partial(x_{il}x_{jl})}{\partial x_{kl''}} = (x_{jl'}\mathbf{1}_{il=kl''} + x_{il}\mathbf{1}_{jl=kl''})$ only when $il \neq jl'$. In the above sum, we can write this identity because when $l = l'$, the first factor of the sum is equal to 0.

Similarly, the second derivative of f with respect to $x_{kl''}$ and $x_{k'l'''}$ is the vector $B_{.,kl'',k'l'''}$ that is such that:

$$\begin{aligned}
B_{.,kl'',kl'''} &= \frac{\partial A_{.,kl'}}{\partial x_{k'l'''}} \\
&= \sum_{il,jl'} \left(-\mathbf{e}_{il} + \mathbf{e}_{il'} + \mathbf{e}_{jl} - \mathbf{e}_{jl'}\right) 2\mathbf{1}_{il=kl'',jl=k'l'''} \frac{\rho_{ill'}}{m_{l'}} \\
&= 2\left(-\mathbf{e}_{kl''} + \mathbf{e}_{kl'''} + \mathbf{e}_{k'l''} - \mathbf{e}_{k'l'''}\right) \frac{\rho_{kl''l'''}}{m_{l'''}}.
\end{aligned}$$

Finally following [21], the matrix Q is

$$\begin{aligned}
Q = \sum_{il,jl'} \rho_{ill'} \frac{x_{il}x_{jl'}}{m_{l'}} \Big(&\mathbf{e}_{il,il} + \mathbf{e}_{il',il'} + \mathbf{e}_{jl,jl} + \mathbf{e}_{jl',jl'} \\
&+2(-\mathbf{e}_{il,il'} - \mathbf{e}_{il,jl} + \mathbf{e}_{il,jl'} + \mathbf{e}_{il',jl} - \mathbf{e}_{il',jl'} - \mathbf{e}_{jl,jl'}) \Big).
\end{aligned}$$

K. Further validation: models with synchronous streams

1) Parameters for models with synchronous streams: The following table lists the model parameters for the experiments discussed in Section VIII-A.

n	Number of items	10, 50, 100
n/m	Items-to-capacity ratio	2, 4, 10
h	Number of lists	1, 2, 3
u	Number of streams	2
α_1	Zipf parameter, stream 1	0.6, 1.0, 1.4
α_2	Zipf parameter, stream 2	1.0
σ_1	Mean think time, stream 1	0.001s, 0.01s, 0.1s, 1.0s, 10s
σ_2	Mean think time, stream 2	$2\sigma_1$
θ_{v0}	Mean fetch (hit) latency	0.01s
θ_{v1}	Mean miss latency	0.10s
s_v	Maximum outstanding requests per stream	1
c	Normalized access prob.	1.0

2) Sensitivity to number of outstanding requests: We have also considered variants where for every stream v the maximum number of outstanding requests is varied as $s_v = 1, 4, 16$, and different mixes of think time values are used in the model. We keep the number of samples used in the simulations to 10^7. In these experiments, we set $n = 10, 50, 100$, $h = 2$, $c = 0.5$, $\alpha_1 = 1.4$, and use both FIFO-C($\boldsymbol{m}$) and RR-C($\boldsymbol{m}$), for a total of 162 experiments. The results indicate that average errors remain small for all mixes (σ_1, σ_2), with FPI ranging between 2.01% and 2.33% mean absolute percentage errors across the mixes and with SPA between 0.18% and 0.49%. For both methods, the largest error for FPI is observed for mix $(16, 1)$, while for SPA is for mix $(1, 16)$.

3) Sensitivity to access probabilities and tree structure: We now consider another variant where the cache has $h = 5$ lists arranged in a tree structure. As before, we consider $n = 10, 50, 100$, $n/m = 10, 50, 100$, $\alpha_1 = 0.6, 1.0, 1.4$, $\alpha_2 = 1.0$, and $s_1 = s_2 = 1$. Access probabilities for list $l = 0$, corresponding to cache misses, are set to 1.0, for all $v = 1, \ldots, u$. Access probabilities for stream v requests to item k with a cache hit are given by the following matrix

$$C_{vk} = \begin{bmatrix} 0 & 1 & & & & \\ & \frac{k\cdot v}{1+k\cdot v} & \frac{1}{3(1+k\cdot v)} & & \frac{2}{3(1+k\cdot v)} & \\ & & \frac{v}{1+v} & \frac{1}{1+v} & & \\ & & & 1 & & \\ & & & & \frac{3}{10} & \frac{7}{10} \\ & & & & & 1 \end{bmatrix}$$

where element in row l and column j corresponds to $c_{vk}(l, j)$, for $l, j = 0, \ldots, h$. Costs on the diagonal give the probability that the item upon a hit does not get promoted to another list.

With the above parameters we find that for FIFO-C($\boldsymbol{m}$) the average errors are 1.23% for FPI and 0.38% for SPA, while maximum errors are 4.34% for FPI and 1.41% for SPA. We also repeat the experiment with RR-C($\boldsymbol{m}$) finding average errors of 1.21% for FPI and 0.32% for SPA, and maximum errors of 4.17% for FPI and 1.41% for SPA, which are close to the FIFO-C($\boldsymbol{m}$) results and compatible with simulation error.

L. Validation: hybrid caching-queueing models

1) Parameters for models with queueing: The following table lists the model parameters for the experiments on synchronous models with queueing stations. We index nodes as t (think time), c (cache), h (hit queue), and m (miss queue).

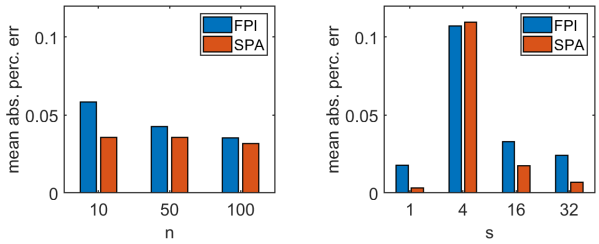


Fig. 11: FPI and SPA item miss ratios on the parallel topology in Figure 8 with correlated item reads.

n	Number of items	10, 50, 100
n/m	Items-to-capacity ratio	2, 4, 10
h	Number of lists	2
u	Number of streams	2
α_{c1}	Zipf parameter at cache c, stream 1	1.4
α_{c2}	Zipf parameter at cache c, stream 2	1.0
θ_{t1}	Mean think time, stream 1	1.0s
θ_{t2}	Mean think time, stream 2	2.0s
θ_{h1}	Mean fetch (hit) latency, stream 1	0.01s
θ_{h2}	Mean fetch (hit) latency, stream 2	0.01s
θ_{m1}	Mean miss latency, stream 1	0.10s
θ_{m2}	Mean miss latency, stream 2	0.50s
s	Maximum outstanding requests in each stream	1, 4, 16, 32
c	Normalized access prob.	1.0

2) Correlated item reads: We consider a variant of the parallel topology in Figure 8 in which a request that completes in stream 1 deterministically switches class at the reference station (think time node) so to re-enter in the following cycle as a synchronous request of stream 2, and vice-versa for requests completing in stream 2. This is sufficient to introduce a degree of serial correlation in the item references at the cache, since each requests will deterministically alternate requests drawn for Zipf-like distributions with different parameters ($\alpha_{c1} \neq \alpha_{c2}$).

While our approximations cannot explicitly represent correlations, the results shown in Figure 11 show that both FPI and SPI remain fairly accurate on the considered model, especially with small and large number of outstanding requests. This suggests that the proposed methods can still provide reasonably accurate results in models that are reasonable close to the reference ones, but that strictly speaking do not meet the assumptions.