Performance Analysis of the Randomized SIEVE/CLOCK Cache Replacement Algorithm

YIRONG WANG, Northeastern University, USA
PETER DESNOYERS, Northeastern University, USA
BENNY VAN HOUDT, University of Antwerp, Belgium

Recently SIEVE was introduced as a new cache-eviction algorithm that excels in efficiency and simplicity. Like the CLOCK algorithm it uses a single list and an *access bit* per cached item. The SIEVE and CLOCK algorithm operate in the same manner, except that SIEVE inserts items in a fixed position in the list, instead of in the position of the evicted item. Both the SIEVE and CLOCK algorithm can be naturally generalized to a setting where multiple access bits are used per cached item.

Motivated by the need to improve the performance for workloads with long scan sequences, we introduce a randomized version of the SIEVE/CLOCK algorithm using $\lceil \log_2(K+1) \rceil \geq 1$ access bits per cached item. We present a heterogeneous mean-field model to assess the performance of the randomized SIEVE/CLOCK algorithm and prove that it has a unique fixed point that can be easily computed using bisection. An explicit expression for the fixed point is presented when K tends to infinity. The accuracy of the model is demonstrated using simulations. Theoretical support for the observed accuracy is presented by arguing that the mean-field model of a slightly modified version of the algorithm is $O(1/n)$ accurate, where n is the population size.

We end the paper by comparing the performance of the randomized SIEVE/CLOCK algorithm with other cache replacement algorithms to demonstrate improved hit rates (by a factor 1.5 or more) on workloads with long scan sequences. Furthermore, though the cache hit rate improves with the number of access bits, we show that the majority of the gain is already achieved with as few as 4 access bits per cached item.

CCS Concepts: • **Theory of computation → Caching and paging algorithms**; • **Mathematics of computing → Stochastic processes**.

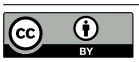
Additional Key Words and Phrases: Caching, CLOCK, SIEVE, Mean-Field Model

ACM Reference Format:
Yirong Wang, Peter Desnoyers, and Benny Van Houdt. 2026. Performance Analysis of the Randomized SIEVE/CLOCK Cache Replacement Algorithm. *Proc. ACM Meas. Anal. Comput. Syst.* 10, 2, Article 49 (June 2026), 25 pages. https://doi.org/10.1145/3805647

1 Introduction

Modern storage stacks and operating systems rely heavily on in-memory caching to bridge the gap between fast compute and slower persistent media. This has driven sustained interest in replacement policies that are simultaneously effective, simple to implement at scale, and predictable under production workloads. Among these, CLOCK-like algorithms remain particularly attractive: they approximate LRU using only a small amount of per-object state and a cheap in-place eviction mechanism dating back to early operating systems [8]. Recently, the systems community has continued to refine such "small-state" eviction policies for real workloads, emphasizing operational simplicity (e.g., SIEVE [23]).

Authors' Contact Information: Yirong Wang, Northeastern University, Boston, MA, USA, wang.yiron@northeastern.edu; Peter Desnoyers, Northeastern University, Boston, MA, USA, P.Desnoyers@northeastern.edu; Benny Van Houdt, University of Antwerp, Antwerp, Belgium, benny.vanhoudt@uantwerpen.be.



ACM 2476-1249/2026/6-ART49
https://doi.org/10.1145/3805647

A recurring practical challenge is that real workloads can become *scan-dominated*, leading to pathologies such as *performance cliffs*, wherein increasing cache capacity has little effect over a range of sizes until a workload-dependent threshold is crossed, after which hit rate improves abruptly and disproportionately. Figure 1 illustrates hit-rate cliffs under different policies on the production trace `volume28` from [15]—this effect is not well captured by classical Markovian workload assumptions, yet it is routinely observed in real traces. Policies with such cache behaviors are operationally problematic, since provisioning the cache just below a critical size can drive the hit ratio close to zero, causing catastrophic performance degradation due to excessive request misses.

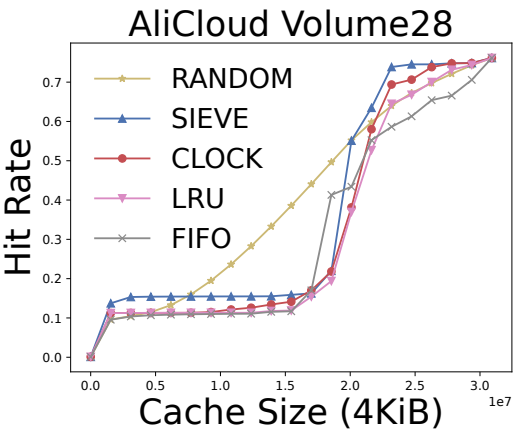


Fig. 1. Scan-induced performance cliffs for common cache replacement policies on Alibaba block I/O trace `volume28`.

Prior work has proposed several remedies for scan-induced performance cliffs, such as using mixed complementary policies based on online learning [18], using cache partitioning to ensure convex miss rate reduction [2]. While effective, these techniques lie outside the small-state replacement paradigm, as they rely on substantially more structure, coordination, and control overhead than lightweight eviction policies.

Performance cliffs arise precisely because scan-based eviction paths create a synchronized, deterministic aging frontier in which many objects are admitted and removed in lockstep. This observation motivates a design principle for cache replacement: *randomize the eviction path to mitigate worst-case scans while keeping the per-object state minimal*. Intuitively, randomization can mitigate this effects performance cliffs by dispersing eviction pressure across objects; from a queueing-theoretic perspective, this dispersion is akin to a per-object birth-death process. Indeed, using a naive RANDOM eviction policy on `volume28`, we can avoid the performance cliff and achieve about 10.22% higher average hit rate than SIEVE. Section 7 further substantiates this idea with a simple heuristic mean-field approximation.

Concretely, we introduce randomized counterparts of CLOCK and SIEVE that replace scan-based victim search with random probing. Objects carry only a small counter with maximum value K—equivalently, $\lceil \log_2(K+1) \rceil$ bits per object—which is incremented on hits and decremented during eviction attempts. Random probing preserves the spirit of CLOCK (cheap state, local updates) while making eviction behavior less sensitive to adversarial or trace-induced long scans.

Randomization comes at some costs: under purely Markovian arrivals, scan-based evictee search can leverage strong recency and may achieve slightly higher hit rates than their randomized counterparts. However, our experiments indicate that these losses are modest, while the robustness gains under renewal effects and real workloads can be substantial. This trade-off motivates a principled performance model that can quantify the performance of randomized CLOCK/SIEVE caching and how to tune the counter cap K.

To this end, we develop a heterogeneous mean-field model for the randomized CLOCK/SIEVE family. The mean-field limit yields a tractable dynamical system for the evolution of object-level state probabilities. At stationarity, each object's marginal distribution has the same form as the steady state of a finite-capacity queue driven by that object's request process: under the Independent Reference Model (IRM) it is the stationary distribution of an M/M/1/K+1 queue, while under

the phase-type renewal (PH) model it is the stationary distribution of a PH/M/1/K+1 queue. In both cases, the distribution depends on a single scalar (an effective service rate) that is uniquely determined by the cache-size constraint and can be found efficiently by bisection.

Theoretically, the randomized eviction path is not only a practical design choice but also a modeling advantage: with slight modification, it yields a natural CTMC description with bounded interaction degree, which fits the heterogeneous mean-field framework of Allmeier and Gast [1]. This provides an $O(1/n)$ justification for the accuracy of the mean-field approximation in large systems, and explains why the model remains predictive under the cache scalings considered in our experiments.

Our main contributions are:

- We introduce a randomized CLOCK/SIEVE family with maximum count K (i.e. $O(\log K)$ bits per cached object) designed to be robust to long scan sequences.
- We derive a heterogeneous mean-field model for the transient behavior and fixed points, the latter of which can be computed using a bisection algorithm. As K tends towards infinity, each fixed point admits a simple closed form solution.
- We validate the model against simulations and demonstrate that randomization provides scan resistance and robust performance on real workloads while incurring only modest losses in Markovian settings.
- We frame the model using heterogeneous mean-field theory [1], providing a theoretical justification for the observed accuracy.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces the policies and workload models and summarizes notation. Section 4 develops the mean-field model for the randomized CLOCK and SIEVE algorithms under IRM (Section 4.1) and PH (Section 4.2) workloads. Section 5 validates the model against simulations. Theoretical support for the mean-field approximation is demonstrated in Appendix B. Section 6 reports additional simulation results and distills practical guidelines for policy and parameter selection. Section 7 explains why randomized eviction mitigates performance cliffs, and Section 8 concludes the paper.

2 Related Work

Analytical models for caching have a long history, with a particularly rich literature for LRU under the IRM. Early approximations date back to classical work on buffer replacement [9], and the TTL/Che's approximation [7, 11] and its refinements have become standard tools for predicting hit ratios. More recent work develops accurate and broadly applicable approximations, including unified frameworks and TTL-based viewpoints for networks of caches [4, 12, 16].

Mean-field and fluid limits have also been used to analyze families of multi-list and randomized policies, where large-system limits yield deterministic dynamics and often tractable fixed points. Our approach is closely related in spirit to the mean-field analysis of list-based caches in [13, 14], but targets a CLOCK-like mechanism with access counters rather than a list discipline.

CLOCK and its variants are among the most widely deployed practical replacements, dating back to early operating systems [8]. Compared to LRU, rigorous performance characterizations are more limited, in part because scan-based eviction introduces strong coupling across many cached objects.

On the systems side, recent work continues to refine eviction policies for real workloads, emphasizing simplicity; SIEVE [23] is a notable example. SIEVE can be regarded as part of the broader CLOCK-like family because it uses the same per-object access-bit mechanism; it differs only in that newly admitted objects are always inserted at a fixed position.

Robustness in cache replacement under non-stationary access patterns, especially long scans and rapid churn, has motivated a class of scan- and churn-resistant designs. E.g., CACHEUS [18] models

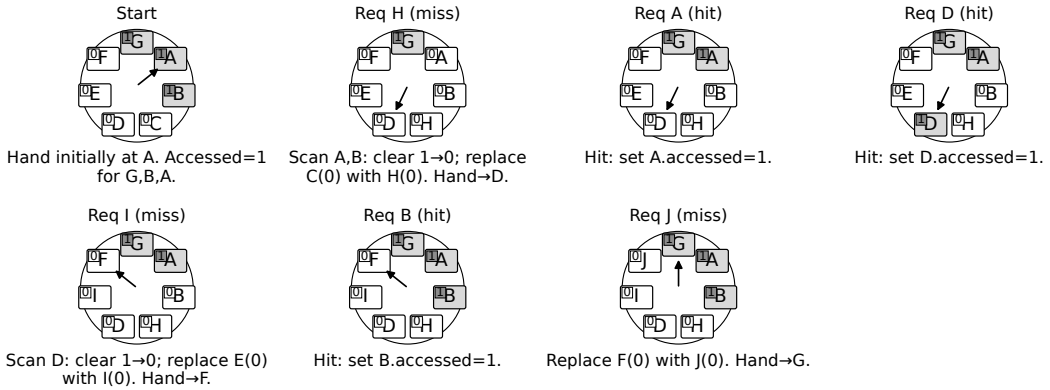


Fig. 2. CLOCK on the request sequence $H, A, D, I, B, J, \ldots$, starting from cache state $A(1), B(1), C(0), D(0), E(0), F(0), G(1)$; the eviction hand scans clockwise and initially points to A. Each panel shows the cache state immediately after processing the corresponding request.

production traces as mixtures of a small set of workload primitives and uses online learning to adaptively weight complementary expert policies as the dominant primitive shifts over time. Related analytical work studies a broad, tunable family of list-based policies [14] that enable exploring performance trade-offs via list tuning. Beyond replacement policy design, Talus [2] proposes a cache-management (partitioning) scheme to eliminate performance cliffs.

Randomization has appeared in cache replacement for purposes other than robustness. E.g., Hyperbolic caching [6] uses random sampling at eviction to approximate the lowest-priority item under a score-based policy, enabling flexible objectives such as cost- and expiration-aware eviction. In these settings, randomization serves mainly flexibility or implementation efficiency.

Reinforced counters [10] model each object as a birth-death process with a geometric stationary distribution over counter states, where insertion and eviction are governed by explicit counter thresholds. We note this per-object birth-death form admits a mean-field extension as in this work.

A closely related line of work develops tools and datasets for workload characterization and miss ratio curve analysis [21, 22], which we use to simulate and validate our modeling assumptions. In particular, miss ratio curve analysis makes the cache behavior of the workload under evaluation interpretable, and can be used to validate analytical models.

3 Preliminaries

We consider a cache of capacity C storing objects from a fixed catalog $\{1, \ldots, n\}$ with $n > C$. All objects have equal size, so the cache can hold up to C objects, never more. A replacement algorithm is used to determine which object to evict on a cache miss when the cache is full. Time is continuous and requests arrive according to one of the workload models described in Section 3.2.

3.1 Replacement Algorithms

We briefly recall the policies used as baselines. The LRU, FIFO and RANDOM are simple enough to need no description, but several variants of CLOCK have been described; we use the classic single-handed CLOCK algorithm from the Multics system [8], which stores cached objects in a circular array with a pointer (the "clock hand") pointing to an eviction candidate. As a concrete example, Figure 2 illustrates CLOCK on a short request sequence, starting from a cache containing $\{A, B, \ldots, G\}$ with the hand initially pointing to A. The accessed bit is set for A, B, and G. The first

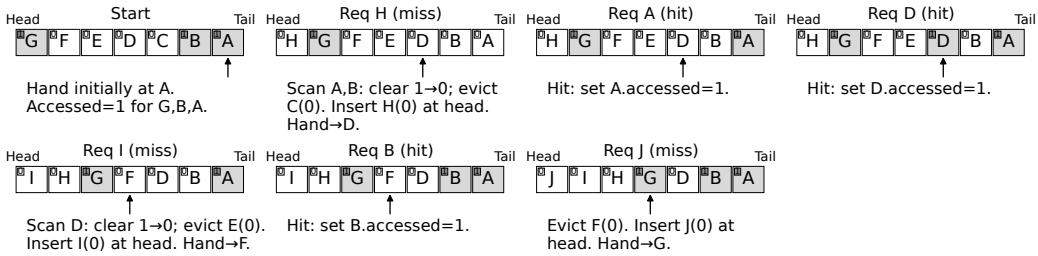


Fig. 3. SIEVE on the same request sequence $H, A, D, I, B, J, \ldots$, starting from the same cache state $A(1), B(1), C(0), D(0), E(0), F(0), G(1)$; the eviction hand scans from tail to head and initially points to A. Each panel shows the cache state immediately after processing the corresponding request.

request H misses, so the hand scans clockwise: it clears A and B and then replaces the first entry with bit 0 (here, C) by H; the hand then advances to D. Subsequent requests proceed analogously: hits set the accessed bit, while misses advance the hand, clearing any 1-bit object encountered until reaching a 0-bit victim for *in-place* replacement.

SIEVE stores cached objects in a FIFO-ordered list and similarly maintains a per-object access bit. On a miss when the cache is full, it probes the object at the *current* hand position: if the bit is 0 the object is evicted; otherwise the bit is reset to 0 and the hand advances one step toward the head (wrapping from head back to tail as needed) and probing continues until a 0-bit victim is found. The key difference with SIEVE is (a) that the evictee is *removed* from the list, not replaced in-place, and (b) the new item is *inserted* into the list at a fixed position (the head of the list, in a linear representation). To support these updates SIEVE typically requires a doubly-linked list or comparable data structure, rather than a simple circular array. Figure 3 illustrates SIEVE on the same request sequence and initial cache state as the CLOCK example.

We then introduce CLOCK(K)/SIEVE(K) and their randomized counterparts. All these algorithms use a small amount of per-object state: each cached object carries an *access counter* in $\{0, \ldots, K\}$ (equivalently, $\lceil \log_2(K{+}1) \rceil$ bits). On a hit, the counter is incremented by one up to K. On a miss, the policy repeatedly inspects candidate objects and (i) evicts the first object found with counter 0, or (ii) if the candidate has counter > 0, decrements it by one and continues searching. The policies differ only in how candidates are generated (sequential scan versus random probing) and where the missing item is inserted.

CLOCK(K) and SIEVE(K). CLOCK(K)[1] generalizes the access bit of CLOCK to an access counter: on a miss, the hand scans positions and decrements the counter of each examined object by one when it is positive, advancing until it finds a counter-0 victim to replace. New objects are inserted at the current hand position with counter 0. With K=1, CLOCK(K) is the classic CLOCK.

Similarly, we generalize SIEVE to SIEVE(K) where hits increment the counter up to K. With $K = 1$, SIEVE(K) is the original SIEVE.

With $K = 0$, both CLOCK(K) and SIEVE(K) degenerate to FIFO.

Ran-CLOCK(K)/Ran-SIEVE(K). These randomized variants replace sequential scans with uniform random probing. On a miss, Ran-CLOCK(K) repeatedly samples a cached object uniformly at random (with replacement): if its counter is 0, it is evicted and replaced; otherwise the counter is

[1]Corbató's original paper [8] describes a K-bit shift register per entry: references set the top bit, and each hand probe shifts the register, evicting when the register becomes zero. To our knowledge, this shift-register variant was never implemented in practice. A closely related generalization, often called GCLOCK [17, 20], replaces the register with a counter that is refreshed on each reference (typically set to K) and decremented on each probe, again evicting at zero. CLOCK(K) differs in that hits increment the counter by one up to K rather than resetting it.

decremented and probing continues. Ran-SIEVE(K) is defined analogously, differing only in its internal list representation and its *fixed* insertion position for new arrivals. Because probing is uniform and does not depend on the list order, these implementation details do not affect which objects are probed, decremented, or evicted. As a result, Ran-SIEVE(K) and Ran-CLOCK(K) induce the same evolution of object counters (up to permutation) and therefore have the same steady-state hit/miss ratios (equivalently, over sufficiently long streams). For the workload models in the next subsection this can be easily shown using a coupling argument.

For $K = 0$ both Ran-CLOCK(K) and Ran-SIEVE(K) reduce to RANDOM.

3.2 Workload Models

We model requests as the superposition of n independent object-level processes. Throughout, we normalize the total request rate to 1 for convenience.

Independent Reference Model. Under IRM, requests for object k arrive as a Poisson process of rate p_k, with

$$p_k \geq 0, \qquad \sum_{k=1}^{n} p_k = 1.$$

Equivalently, in discrete time one may view successive requests as i.i.d. samples from the distribution (p_k). We assume without loss of generality that $p_1 \geq p_2 \geq \cdots \geq p_n$.

Phase-type renewal model. To capture various inter-request times, we also consider a renewal model. For each object k, requests arrive according to an independent renewal process with i.i.d. inter-request times having a phase-type (PH) distribution described by a pair $(\boldsymbol{\alpha}_k, \mathbf{T}_k)$, where $\boldsymbol{\alpha}_k$ is an initial distribution over phases and $\mathbf{T}_k$ is a subgenerator matrix. Let $\mathbf{1}$ be the all-ones column vector and define the absorption-rate vector $\boldsymbol{\mu}_k := -\mathbf{T}_k\mathbf{1}$. The mean inter-request time of object k is

$$\mathbb{E}[A_k] = -\boldsymbol{\alpha}_k\mathbf{T}_k^{-1}\mathbf{1},$$

so the long-run request rate is $\lambda_k := 1/\mathbb{E}[A_k]$. We normalize $\sum_{k=1}^{n} \lambda_k = 1$ and define the corresponding popularity weights

$$p_k := \lambda_k,$$

so that p_k coincides with the request rates when inter-request times are exponential.

3.3 Notation

We summarize the main notation used throughout the paper in Table 1. Throughout, $x_{k,j}(t)$ denotes a probability, while aggregate quantities such as $x_0(t)$ and $x_{1+}(t)$ are expected counts. Vectors and matrices (e.g., $\mathbf{x}_{k,j}(t)$, $\boldsymbol{\alpha}_k$, and $\mathbf{T}_k$) are written in boldface, with dimensions clear from context.

4 Mean-Field Model for Ran-CLOCK(K)/Ran-SIEVE(K)

In this section, we develop a heterogeneous mean-field model for Ran-CLOCK/Ran-SIEVE(K), following the approach of [13] for RAND($\mathbf{m}$). We consider a cache of capacity C serving a catalog of $n > C$ equal-sized objects.

4.1 Independent Reference Model

In the IRM, requests for item k arrive at some rate p_k. We may assume that $\sum_{k=1}^{n} p_k = 1$ and that p_k is non-increasing in k, that is, item 1 is the most popular and item n the least popular. Define the variables

- $x_{k,-1}(t)$ as the probability that item k is not cached at time t,
- $x_{k,j}(t)$ as the probability that item k is cached and its access counter equals j at time t,

Table 1. Notation.

Symbol	Meaning
n	Catalog size; objects are indexed by $k \in \{1, \dots, n\}$.
C	Cache capacity (number of objects).
K	Counter cap in CLOCK(K), SIEVE(K), and Ran-CLOCK(K)/Ran-SIEVE(K); counters take values in $\{0, \dots, K\}$.
$\mathcal{S}$	Local state space $\mathcal{S} = \{-1, 0, 1, \dots, K\}$; -1 denotes "not cached".
$S_k(t)$	Local state of object k at time t; $S_k(t) \geq 0$ iff k is cached.
p_k	Request rate / popularity weight of object k (normalized so $\sum_{k=1}^{n} p_k = 1$).
θ	Zipf exponent, $p_k \propto k^{-\theta}$.
$x_{k,j}(t)$	Probability $x_{k,j}(t) = \mathbb{P}(S_k(t) = j)$ for $j \in \mathcal{S}$.
$x_0(t)$	Expected number of cached objects with counter 0, $x_0(t) = \sum_k x_{k,0}(t)$.
$x_{1+}(t)$	Expected number of cached objects with counter ≥ 1, $x_{1+}(t) = \sum_{j \geq 1} \sum_k x_{k,j}(t)$.
$m(t)$	Miss rate.
z	Service rate $z = m/x_0$; (at stationarity $z = z_K$).
z_K	Unique fixed-point service rate.
$p_{\text{miss}}(C, K)$	Steady-state miss rate.
$(\boldsymbol{\alpha}_k, \mathbf{T}_k)$	PH parameters of object k: initial phase distribution and subgenerator matrix.
d_k	Number of phases of object k in the PH model.
$\boldsymbol{\mu}_k$	Absorption-rate vector $\boldsymbol{\mu}_k = -\mathbf{T}_k \mathbf{1}$.
λ_k	Long-run request rate of object k ($\sum_k \lambda_k = 1$); we set $p_k = \lambda_k$.
$\mathbf{x}_{k,j}(t)$	Row vector of phase probabilities when object k has counter j in the PH model.
$h_{k,j}(t)$	Instantaneous request rate in state j under PH, $h_{k,j}(t) = \mathbf{x}_{k,j}(t)\boldsymbol{\mu}_k$.
$\mathbf{1}, \mathbf{I}$	All-ones column vector and identity matrix (dimension is clear from context).

for $j = 0, \dots, K$. Hence, $\sum_{j=-1}^{K} x_{k,j}(t) = 1$ for all k and t, and $\sum_k x_{k,-1}(t) = n - C$ for all t if we assume the cache is full at time 0 as the cache size equals C.

Let $m(t) = \sum_k p_k x_{k,-1}(t)$ be the miss rate at time t and $x_0(t) = \sum_k x_{k,0}(t)$ be the expected number of objects in the cache with access counter zero. Similarly, define $x_{1+}(t) = \sum_{j \geq 1} \sum_k x_{k,j}(t)$ as the expected number of objects with a use counter larger than zero. The drift equations characterizing the model are as follows:

$$\frac{d}{dt} x_{k,K}(t) = p_k x_{k,K-1}(t) - m(t) \frac{x_{k,K}(t)}{x_0(t)}, \tag{1}$$

$$\frac{d}{dt} x_{k,j}(t) = p_k x_{k,j-1}(t) - p_k x_{k,j}(t) + m(t) \left(\frac{x_{k,j+1}(t)}{x_0(t)} - \frac{x_{k,j}(t)}{x_0(t)} \right), \tag{2}$$

$$\frac{d}{dt} x_{k,-1}(t) = -p_k x_{k,-1}(t) + m(t) \frac{x_{k,0}(t)}{x_0(t)}, \tag{3}$$

for $j = 0, \dots, K-1$. The terms involving p_k are obvious as a request for item k occurs at rate p_k and increases its use counter by one (unless it equals K). To understand the terms involving $m(t)/x_0(t)$, note that when a miss occurs, $\frac{C}{x_0(t)} - 1 = x_{1+}(t)/x_0(t)$ represents the expected number of failed attempts that are observed on average to randomly select an item in the cache with an access counter equal to 0. For each failed attempt, item k with access counter $j \geq 1$ is selected with probability $x_{k,j}(t)/x_{1+}(t)$. Note that in the mean-field model, a single miss decreases the counter of an item by at most one, contrary to simulations where the same item can be selected multiple

times. The reasoning behind this is that the probability of selecting the same item twice tends to zero as the cache size tends to infinity.

4.1.1 Computing the Unique Fixed Point. The fixed point equations are given by

$$0 = p_k x_{k,K-1} - \frac{m}{x_0} x_{k,K}, \tag{4}$$

$$0 = p_k x_{k,j-1} - p_k x_{k,j} + \frac{m}{x_0}(x_{k,j+1} - x_{k,j}), \tag{5}$$

$$0 = -p_k x_{k,-1} + \frac{m}{x_0} x_{k,0}, \tag{6}$$

with $m = \sum_k p_k x_{k,-1}$ and $x_0 = \sum_k x_{k,0}$, for $j = 0, \ldots, K-1$.

Theorem 1. *The set of fixed point equations given by (4)-(6) has a unique solution given by*

$$x_{k,j} = \frac{(p_k/z_K)^{j+1}}{\sum_{i=0}^{K+1}(p_k/z_K)^i},$$

for $j = -1, \ldots, K$, where $z_K \in (0,1)$ is the unique solution of

$$n - C = \sum_{k=1}^{n} \frac{1}{\sum_{i=0}^{K+1}(p_k/z)^i}, \tag{7}$$

which can be computed using a simple bisection algorithm on $(0,1)$. The miss probability equals

$$p_{miss}(C,K) = \sum_{k=1}^{n} \frac{p_k}{\sum_{i=0}^{K+1}(p_k/z_K)^i}. \tag{8}$$

Proof. Assume we have a fixed point $(x_{k,j})_{k=1,\ldots,n;j=-1,\ldots,K}$ and denote $z = \sum_k x_{k,-1}p_k / \sum_k x_{k,0} = m/x_0$. The set of fixed point equations is a linear set of equations if we assume z is a fixed number (which we determine later). These equations correspond to a birth-death process, such that

$$x_{k,j} = \frac{(p_k/z)^{j+1}}{\sum_{i=0}^{K+1}(p_k/z)^i},$$

must hold. Define

$$f(z) = \sum_k x_{k,-1} = \sum_k \frac{1}{\sum_{i=0}^{K+1}(p_k/z)^i}.$$

Then $\lim_{z\to 0^+} f(z) = 0, \lim_{z\to\infty} f(z) = n$ and $f(z)$ is increasing in z on $(0,\infty)$. As the number of items in the cache equals C, we must have that

$$f(z) = n - C.$$

Hence, there exists a unique $z_K \in (0,\infty)$ that solves this equation and therefore there is a unique fixed point. Further,

$$f(1) = \sum_k \frac{1}{\sum_{i=0}^{K+1}(p_k)^i} > \sum_k \frac{1}{\sum_{i=0}^{\infty}(p_k)^i} = \sum_k (1-p_k) = n-1.$$

This means that the value of z for which $f(z) = n - C$ is strictly upper bounded by one. □

It is worth noting that $(x_{k,-1}, \ldots, x_{k,K})$ corresponds to the stationary distribution of an $M/M/1/K+1$ queue with arrival rate p_k and service rate z_K for all k. The service rate z_K is such that the sum of

the probabilities that these n queues are empty, given by $\sum_k x_{k,-1}$, equals $n - C$. We further note that the expected number of probes needed per miss is given by

$$\frac{C}{x_0} = \frac{Cz_K}{p_{miss}(C,K)}. \tag{9}$$

The next result shows that a simple explicit expression exists for the limit of z_K when K tends to infinity, which also yields a simple explicit expression for the limit of $p_{miss}(C,K)$ as K tends to infinity.

Corollary 1. *Let $\ell \in \{0,\ldots,C-1\}$ be the smallest number such that*

$$p_{\ell+1} < \frac{\sum_{k>\ell} p_k}{C-\ell}, \tag{10}$$

then

$$z_\infty = \lim_{K\to\infty} z_K = \frac{\sum_{k>\ell} p_k}{C-\ell}, \tag{11}$$

and

$$\lim_{K\to\infty} p_{miss}(C,K) = \sum_{k>\ell} p_k - (C-\ell)\frac{\sum_{k>\ell} p_k^2}{\sum_{k>\ell} p_k}.$$

Proof. When $\ell = 0$, we have $p_1 < 1/C$ and one readily verifies that $z = 1/C$ is the unique solution of (7) as $p_k/z < 1$ for all k. For $\ell > 0$, denote $E_s = \sum_{k\geq s} p_k$. We prove that the unique z solving (7) for K tending to infinity is given by $z_\infty = E_{\ell+1}/(C-\ell)$. By definition of ℓ we have

$$p_s \geq \frac{E_s}{C-s+1}, \tag{12}$$

for $s = 1,\ldots,\ell$. We first show that $p_s \geq z_\infty$ for $s = 1,\ldots,\ell$. Hence, by (12) it suffices to show that

$$a_s := \frac{E_s}{C-s+1} \geq \frac{E_{\ell+1}}{(C-\ell)} = a_{\ell+1} = z_\infty,$$

for $s = 1,\ldots,\ell$. This clearly holds when $a_s \geq a_{s+1}$. It is easy to see that

$$a_s - a_{s+1} = \frac{p_s(C-s) - E_{s+1}}{(C-s+1)(C-s)}.$$

So $a_s - a_{s+1}$ is positive if and only if $p_s(C-s) - E_{s+1}$ is positive. By (12) we find

$$p_s \geq \frac{p_s + E_{s+1}}{C-s+1} \quad \Leftrightarrow \quad p_s(C-s) \geq E_{s+1},$$

allowing us to conclude that $a_s \geq a_{s+1}$ for $s = 1,\ldots,\ell$ and therefore $p_k \geq z_\infty$ for $k \leq \ell$ and $p_k < z_\infty$ for $k > \ell$. Using the same definition for f as in the previous proof, we have

$$\lim_{K\to\infty} f(z_K) = \sum_{k>\ell}(1 - \frac{p_k}{z_\infty}) = n - \ell - \frac{E_{\ell+1}}{z_\infty} = n - C,$$

as the first ℓ terms converge to zero. The expression for $\lim_{K\to\infty} p_{miss}(C,K)$ then follows from (8). □

When $p_k < 1/C$ for all k, we have $\lim_{K\to\infty} p_{miss}(C,K) = 1 - C\sum_k p_k^2$. The miss probability therefore only depends on C and the collision probability $\sum_k p_k^2$, which represents the probability that two successive requests are for the same item. When $p_1 \geq 1/C$, meaning $\ell > 0$, the ℓ most popular items never generate a miss (in the limit for K to infinity), while the miss probability is fully determined by C, the probability $\sum_{k>\ell} p_k$ that a request does not belong to the ℓ most popular items and the collision probability of the least $n-\ell$ popular items.

4.2 Phase-Type Renewal Models

We now consider the PH workload model described in Section 3.2. For object k, inter-request times have PH representation $(\boldsymbol{\alpha}_k, \mathbf{T}_k)$ (order d_k), and we write the absorption-rate vector as $\boldsymbol{\mu}_k := -\mathbf{T}_k\mathbf{1}$.

We define the state of item k at time t using row vectors $\mathbf{x}_{k,j}(t) \in \mathbb{R}^{1\times d_k}$ for $j \in \{-1, 0, \ldots, K\}$. The ℓ-th component of $\mathbf{x}_{k,j}(t)$ represents the probability that item k has access counter j (where $j = -1$ indicates the item is not cached) and is in renewal phase ℓ.

The scalar request rate for item k while at counter j is given by the inner product:

$$h_{k,j}(t) = \mathbf{x}_{k,j}(t)\boldsymbol{\mu}_k.$$

The total miss rate is $m(t) = \sum_k h_{k,-1}(t)$, and the normalization factor for the cache probe is $x_0(t) = \sum_k \mathbf{x}_{k,0}(t)\mathbf{1}$.

Let $x_{k,j,\ell}(t)$ denote the ℓ-th component of $\mathbf{x}_{k,j}(t)$, and let $(\mathbf{T}_k)_{r,\ell}$ and $(\boldsymbol{\alpha}_k)_\ell$ denote the (r,ℓ)-th entry of $\mathbf{T}_k$ and the ℓ-th entry of $\boldsymbol{\alpha}_k$, resp. Then, for each phase ℓ, the drift of the state probabilities is given by

$$\begin{aligned}
\frac{d}{dt}x_{k,j,\ell}(t) &= \sum_{r\neq\ell} x_{k,j,r}(t)(\mathbf{T}_k)_{r,\ell} - x_{k,j,\ell}(t)\sum_{r\neq\ell}(\mathbf{T}_k)_{\ell,r} - x_{k,j,\ell}(t)\mu_{k,\ell} \\
&\qquad + \sum_r x_{k,j-1,r}(t)\mu_{k,r}\alpha_{k,\ell} + m(t)\frac{x_{k,j+1,\ell}(t) - x_{k,j,\ell}(t)}{x_0(t)}, \\
&= \sum_{r\neq\ell} x_{k,j,r}(t)(\mathbf{T}_k)_{r,\ell} - x_{k,j,\ell}(t)(-\mathbf{T}_k)_{\ell,\ell} + h_{k,j-1}(t)\alpha_{k,\ell} + m(t)\frac{x_{k,j+1,\ell}(t) - x_{k,j,\ell}(t)}{x_0(t)}, \\
&= (x_{k,j}(t)\mathbf{T}_k)_\ell + h_{k,j-1}(t)\alpha_{k,\ell} + m(t)\frac{x_{k,j+1,\ell}(t) - x_{k,j,\ell}(t)}{x_0(t)},
\end{aligned}$$

for $0 \le j < K$. Collecting the phase components into the row vectors $\mathbf{x}_{k,j}(t)$ yields the following compact form, where $\mathbf{x}_{k,j}(t)\mathbf{T}_k$ captures both internal phase evolution and loss of mass due to absorption.

For $0 \le j < K$:

$$\frac{d}{dt}\mathbf{x}_{k,j}(t) = \mathbf{x}_{k,j}(t)\mathbf{T}_k + h_{k,j-1}(t)\boldsymbol{\alpha}_k + m(t)\left(\frac{\mathbf{x}_{k,j+1}(t)}{x_0(t)} - \frac{\mathbf{x}_{k,j}(t)}{x_0(t)}\right). \tag{13}$$

Here, hits on counter $j-1$ increment the counter to j and reset the phase to $\boldsymbol{\alpha}_k$, while the Ran-CLOCK/Ran-SIEVE probe decrements counters from $j+1$ or out of j without altering their phases. In the same vein we find the following compact expressions for $j = K$ and $j = -1$.

For $j = K$:

$$\frac{d}{dt}\mathbf{x}_{k,K}(t) = \mathbf{x}_{k,K}(t)\mathbf{T}_k + \big(h_{k,K-1}(t) + h_{k,K}(t)\big)\boldsymbol{\alpha}_k - m(t)\frac{\mathbf{x}_{k,K}(t)}{x_0(t)}. \tag{14}$$

The term $(h_{k,K-1} + h_{k,K})\boldsymbol{\alpha}_k$ represents items reaching or staying at counter K (either from a hit at $K-1$ or a hit at K) with phase distribution reset to $\boldsymbol{\alpha}_k$.

For $j = -1$:

$$\frac{d}{dt}\mathbf{x}_{k,-1}(t) = \mathbf{x}_{k,-1}(t)\mathbf{T}_k + m(t)\frac{\mathbf{x}_{k,0}(t)}{x_0(t)}. \tag{15}$$

The term $\mathbf{x}_{k,-1}(t)\mathbf{T}_k$ accounts for the phase evolution of uncached items and the departure of items upon a miss (absorption). The term $m(t)\frac{\mathbf{x}_{k,0}(t)}{x_0(t)}$ represents evicted items (from counter 0) returning to the uncached state, retaining their current phase.

4.2.1 Computing the Unique Fixed Point. We start with the following Lemma, which to the best of our knowledge, is a new result for the PH/M/1/K queue that could be of separate interest.

LEMMA 1. *The stationary distribution* $(\mathbf{x}_0, \ldots, \mathbf{x}_K)$ *of a PH/M/1/K queue, where the arrival renewal process is characterized by* $(\boldsymbol{\alpha}, \mathbf{T})$ *and service rate* z*, is proportional to* $(\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_K)$ *with*

$$\boldsymbol{\theta}_i = -\boldsymbol{\alpha}(\mathbf{T} - z\mathbf{I})^{-1}\mathbf{R}^{K-i}, \tag{16}$$

$$\boldsymbol{\theta}_0 = -z\boldsymbol{\alpha}(\mathbf{T} - z\mathbf{I})^{-1}\mathbf{R}^{K-1}(-\mathbf{T})^{-1}, \tag{17}$$

for $i = 1, \ldots, K$*, with*

$$\mathbf{R} = -z(\mathbf{T} - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha})^{-1}.$$

Hence,

$$\mathbf{x}_0\mathbf{1} = \frac{\boldsymbol{\theta}_0\mathbf{1}}{\sum_{i=0}^{K}\boldsymbol{\theta}_i\mathbf{1}} = \frac{\boldsymbol{\theta}_K(z\mathbf{R}^{K-1}(-\mathbf{T})^{-1}\mathbf{1})}{\boldsymbol{\theta}_K(z\mathbf{R}^{K-1}(-\mathbf{T})^{-1}\mathbf{1} + (\mathbf{I} + \sum_{i=1}^{K-1}\mathbf{R}^i)\mathbf{1})}. \tag{18}$$

PROOF. The rate matrix of a PH/M/1/K queue is typically written as

$$\mathbf{Q} = \begin{bmatrix} \mathbf{T} & \boldsymbol{\mu}\boldsymbol{\alpha} & & & & \\ z\mathbf{I} & \mathbf{T} - z\mathbf{I} & \boldsymbol{\mu}\boldsymbol{\alpha} & & & \\ & \ddots & \ddots & \ddots & & \\ & & z\mathbf{I} & \mathbf{T} - z\mathbf{I} & \boldsymbol{\mu}\boldsymbol{\alpha} & \\ & & & z\mathbf{I} & \mathbf{T} + \boldsymbol{\mu}\boldsymbol{\alpha} - z\mathbf{I} \end{bmatrix}, \tag{19}$$

with $K + 1$ block rows, such that the i-th block row corresponds to the states with $i - 1$ jobs. To prove the theorem, the trick is to reverse the order of the block rows to obtain

$$\mathbf{Q}^{(rev)} = \begin{bmatrix} \mathbf{T} + \boldsymbol{\mu}\boldsymbol{\alpha} - z\mathbf{I} & z\mathbf{I} & & & \\ \boldsymbol{\mu}\boldsymbol{\alpha} & \mathbf{T} - z\mathbf{I} & z\mathbf{I} & & \\ & \ddots & \ddots & \ddots & \\ & & \boldsymbol{\mu}\boldsymbol{\alpha} & \mathbf{T} - z\mathbf{I} & z\mathbf{I} \\ & & & \boldsymbol{\mu}\boldsymbol{\alpha} & \mathbf{T} \end{bmatrix}.$$

Let $(\mathbf{x}_0^{(rev)}, \ldots, \mathbf{x}_K^{(rev)}) = (\mathbf{x}_K, \ldots, \mathbf{x}_0)$ be its stationary distribution. The Markov chains characterized by $\mathbf{Q}$ and $\mathbf{Q}^{(rev)}$ are special cases of a finite level-dependent Quasi-Birth-Death Markov chain [5, Section 7-8]. Such a Markov chain has a rate matrix of the form

$$\hat{\mathbf{Q}} = \begin{bmatrix} \mathbf{A}_{0,0} & \mathbf{A}_{1,0} & & & \\ \mathbf{A}_{-1,1} & \mathbf{A}_{0,1} & \mathbf{A}_{1,1} & & \\ & \mathbf{A}_{-1,2} & \mathbf{A}_{0,2} & \ddots & \\ & & \ddots & \ddots & \mathbf{A}_{1,K-1} \\ & & & \mathbf{A}_{-1,K} & \mathbf{A}_{0,K} \end{bmatrix},$$

and the following backward recursion can be used for its stationary distribution $(\mathbf{y}_0, \ldots, \mathbf{y}_K)$:

$$\mathbf{y}_{i+1} = \mathbf{y}_i\hat{\mathbf{R}}_{i+1}, \tag{20}$$

for $i = 0, \ldots, K - 1$. The $\hat{\mathbf{R}}_i$ matrices can be computed as

$$\hat{\mathbf{R}}_K = -\mathbf{A}_{1,K-1}\mathbf{A}_{0,K}^{-1}, \tag{21}$$

$$\hat{\mathbf{R}}_i = -\mathbf{A}_{1,i-1}\left(\mathbf{A}_{0,i} + \hat{\mathbf{R}}_{i+1}\mathbf{A}_{-1,i+1}\right)^{-1}, \tag{22}$$

for $i = 1, \ldots, K-1$. The vector $\mathbf{y}_0$ solves the system

$$\mathbf{y}_0\left(\mathbf{A}_{0,0} + \hat{\mathbf{R}}_1\mathbf{A}_{-1,1}\right) = 0. \tag{23}$$

For the Markov chain characterized by $\mathbf{Q}^{(rev)}$, we therefore have

$$\hat{\mathbf{R}}_K = z(-\mathbf{T})^{-1}, \tag{24}$$

$$\hat{\mathbf{R}}_i = -z\left(\mathbf{T} - z\mathbf{I} + \hat{\mathbf{R}}_{i+1}\boldsymbol{\mu}\boldsymbol{\alpha}\right)^{-1}, \tag{25}$$

for $i = 1, \ldots, K-1$. We now argue that

$$\hat{\mathbf{R}}_1 = \hat{\mathbf{R}}_2 = \ldots = \hat{\mathbf{R}}_{K-1} = \mathbf{R} = -z(\mathbf{T} - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha})^{-1}. \tag{26}$$

This is the reason for working with $\mathbf{Q}^{(rev)}$ as such a simplification does not exist for $\mathbf{Q}$. For $i = K-1$ this is immediate as $z(-\mathbf{T})^{-1}\boldsymbol{\mu} = z\mathbf{1}$. For $i < K-1$ it suffices to show that $\mathbf{R}\boldsymbol{\mu} = z\mathbf{1}$, which corresponds to showing that

$$-(\mathbf{T} - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha})^{-1}\boldsymbol{\mu} = \mathbf{1}.$$

Multiplying left and right with $(\mathbf{T} - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha})$, we get

$$\boldsymbol{\mu} = -(\mathbf{T} - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha})\mathbf{1},$$

which holds as $\boldsymbol{\alpha}\mathbf{1} = \mathbf{1}$ and $\boldsymbol{\mu} = -\mathbf{T}\mathbf{1}$. By (20), (24) and (26), we therefore have

$$\mathbf{x}_{i+1}^{(rev)} = \mathbf{x}_i^{(rev)}\mathbf{R}, \quad \text{and} \quad \mathbf{x}_K^{(rev)} = \mathbf{x}_{K-1}^{(rev)} z(-\mathbf{T})^{-1} \tag{27}$$

for $i = 0, \ldots, K-2$. By (23) the vector $\mathbf{x}_0^{(rev)}$ solves

$$\mathbf{x}_0^{(rev)}(\mathbf{T} + \boldsymbol{\mu}\boldsymbol{\alpha} - z\mathbf{I} + \mathbf{R}\boldsymbol{\mu}\boldsymbol{\alpha}) = \mathbf{0}.$$

We now argue that $-\boldsymbol{\alpha}(\mathbf{T} - z\mathbf{I})^{-1}$ solves this equation. Using the relation $\mathbf{R}\boldsymbol{\mu} = z\mathbf{1}$, we need to verify that

$$-\boldsymbol{\alpha}(\mathbf{T} - z\mathbf{I})^{-1}(\mathbf{T} + \boldsymbol{\mu}\boldsymbol{\alpha} - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha}) = \mathbf{0},$$

that is,

$$-\boldsymbol{\alpha}(\mathbf{T} - z\mathbf{I})^{-1}(\boldsymbol{\mu} + z\mathbf{1})\boldsymbol{\alpha} = \boldsymbol{\alpha},$$

which holds if $-(\mathbf{T} - z\mathbf{I})^{-1}(\boldsymbol{\mu} + z\mathbf{1}) = \mathbf{1}$. The latter equality is readily verified by multiplying left and right by $(\mathbf{T} - z\mathbf{I})$. Combining this with (27) and the fact that $\mathbf{x}_{K-i} = \mathbf{x}_i^{(rev)}$ completes the proof. □

Theorem 2. *The set of fixed point equations given by (14)-(13) has a unique solution where $(\mathbf{x}_{k,-1}, \ldots, \mathbf{x}_{k,K})$ is given by the stationary distribution of a PH/M/1/K+1 queue with inter-arrival time distribution $(\boldsymbol{\alpha}_k, \mathbf{T}_k)$ and service rate $z_K^{(PH)}$. The service rate $z_K^{(PH)} \in (0,1)$ is the unique solution of*

$$n - C = \sum_{k=1}^{n} \mathbf{x}_{k,-1}\mathbf{1} = \sum_{k=1}^{n} \frac{-\boldsymbol{\alpha}_k(\mathbf{T}_k - z\mathbf{I})^{-1}(z\mathbf{R}_k^K(-\mathbf{T}_k)^{-1}\mathbf{1})}{-\boldsymbol{\alpha}_k(\mathbf{T}_k - z\mathbf{I})^{-1}(z\mathbf{R}_k^K(-\mathbf{T}_k)^{-1}\mathbf{1} + (\mathbf{I} + \sum_{i=1}^{K}\mathbf{R}_k^i)\mathbf{1})}, \tag{28}$$

with

$$\mathbf{R}_k = -z(\mathbf{T}_k - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha}_k)^{-1}. \tag{29}$$

The value of $z_K^{(PH)}$ can be computed using a simple bisection algorithm on $(0,1)$. The miss probability equals

$$p_{miss}(C,K) = \sum_{k=1}^{n} \mathbf{x}_{k,-1}\boldsymbol{\mu}_k = \sum_{k=1}^{n} z_K^{(PH)}\mathbf{x}_{k,0}\mathbf{1},$$

as the total request rate $\sum_k \lambda_k = 1$.

Proof. Assume we have a fixed point $(\mathbf{x}_{k,j})_{k=1,\ldots,n;j=-1,\ldots,K}$ and denote $z = \sum_k \mathbf{x}_{k,-1}\boldsymbol{\mu}_k / \sum_k \mathbf{x}_{k,0}\mathbf{1} = m/x_0$. The set of fixed point equations is given by

$$\begin{aligned}
0 &= \mathbf{x}_{k,K}\left(\mathbf{T}_k + \boldsymbol{\mu}_k\boldsymbol{\alpha}_k - z\mathbf{I}\right) + \mathbf{x}_{k,K-1}\boldsymbol{\mu}_k\boldsymbol{\alpha},\\
0 &= \mathbf{x}_{k,j}\left(\mathbf{T}_k - z\mathbf{I}\right) + \mathbf{x}_{k,j-1}\boldsymbol{\mu}_k\boldsymbol{\alpha}_k + z\mathbf{x}_{k,j+1},\\
0 &= \mathbf{x}_{k,-1}\mathbf{T}_k + z\mathbf{x}_{k,0}.
\end{aligned}$$

If we compare these with (19), we find that $(\mathbf{x}_{k,-1},\ldots,\mathbf{x}_{k,K})$ is the invariant vector of a PH/M/1/K+1 queue with service rate z. In a PH/M/1/K+1 queue, the probability that the queue is empty $\mathbf{x}_{k,-1}\mathbf{1}$ increases in z for all k. This can be proven using the following coupling argument. Consider two PH/M/1/K+1 queues with service rates z and z' respectively, with $z > z'$. We refer to these queues as Q1 and Q2. For any sample path of Q1, we construct a sample path for Q2 as follows:

- All events related to the arrival process are the same in both queues.
- When a service completion occurs in Q1, a service completion is triggered in Q2 with probability z'/z.
- When Q1 is empty and Q2 is not, a service completion is triggered in Q2 at rate z'.

From this coupling it is clear that at any point in time Q2 contains at least as many jobs as Q1. This shows that $\sum_k \mathbf{x}_{k,-1}\mathbf{1}$ is increasing in z for all k and therefore there is a unique z such that $\sum_k \mathbf{x}_{k,-1}\mathbf{1} = n - C$. The expression for $\mathbf{x}_{k,-1}\mathbf{1}$ follows from Lemma 1.

When $z = 1$, we have $\mathbf{x}_{k,-1}\mathbf{1} > 1 - p_k$ (as the load equals p_k and there are some losses), which implies that

$$\sum_{k=1}^{n} \mathbf{x}_{k,-1}\mathbf{1} > n - \sum_k p_k = n - 1 \geq n - C.$$

Therefore the unique $z_K^{(PH)} \in (0,1)$. □

Corollary 2. *Define ℓ as in Corollary 1, then*

$$z_\infty = \lim_{K\to\infty} z_K^{(PH)} = \frac{\sum_{k>\ell} p_k}{C - \ell}, \tag{30}$$

is independent of the shape of the inter-request distribution $(\boldsymbol{\alpha}_k, \mathbf{T}_k)$ and

$$\begin{aligned}
\lim_{K\to\infty} p_{miss}(C,K) &= \sum_{k=1}^{n} \lim_{K\to\infty} \frac{-\boldsymbol{\alpha}_k(\mathbf{T}_k - z_\infty\mathbf{I})^{-1} z_\infty \mathbf{R}_k^K \mathbf{1}}{-\boldsymbol{\alpha}_k(\mathbf{T}_k - z_\infty\mathbf{I})^{-1}(z_\infty\mathbf{R}_k^K(-\mathbf{T})^{-1}\mathbf{1} + (\mathbf{I} + \sum_{i=1}^{K}\mathbf{R}_k^i)\mathbf{1})},\\
&= \sum_{k=\ell+1}^{n} \frac{z_\infty}{z_\infty \mathbf{y}_k^T(-\mathbf{T}_k)^{-1}\mathbf{1} + \rho_k/(\rho_k - 1)},
\end{aligned} \tag{31}$$

with $\rho_k = sp(\mathbf{R}_k)$, $\mathbf{y}_k^T\mathbf{R}_k = \rho_k\mathbf{y}_k^T$, $\mathbf{y}_k^T\mathbf{1} = 1$ and

$$\mathbf{R}_k = -z_\infty(\mathbf{T}_k - z_\infty\mathbf{I} + z_\infty\mathbf{1}\boldsymbol{\alpha}_k)^{-1}.$$

which does depend on the shape of the distribution $(\boldsymbol{\alpha}_k, \mathbf{T}_k)$.

Proof. The $\lim_{K\to\infty} z_K^{(PH)}$ does not depend on the shape of the phase-type distribution as the probability $\mathbf{x}_{k,-1}\mathbf{1}$ that the queue is empty converges to $1 - p_k/z$ when $p_k < z$ and converges to zero when $z \leq p_k$ as K tends to infinity. The $\lim_{K\to\infty} z_K^{(PH)}$ is therefore the same as for the M/M/1/K+1 queue and (30) follows from Corollary 1. The expression for the limit in (31) can be found as follows. First, $\mathbf{T}_k - z_\infty\mathbf{I} + z_\infty\mathbf{1}\boldsymbol{\alpha}_k$ is a sub-generator, therefore

$$\mathbf{R}_k/z_\infty = -(\mathbf{T}_k - z_\infty\mathbf{I} + z_\infty\mathbf{1}\boldsymbol{\alpha}_k)^{-1} = \int_0^\infty e^{(\mathbf{T}_k - z_\infty\mathbf{I} + z_\infty\mathbf{1}\boldsymbol{\alpha}_k)t}dt > 0,$$

meaning $\mathbf{R}_k$ is primitive. By the Perron-Frobenius theorem for primitive non-negative matrices [19], we have

$$\mathbf{R}_k^K = \rho_k^K(\mathbf{x}_k\mathbf{y}_k^T + o(1)),$$

with $\mathbf{R}_k\mathbf{x}_k = \rho_k\mathbf{x}_k$, $\mathbf{y}_k^T\mathbf{R}_k = \rho_k\mathbf{y}_k^T$, $\mathbf{y}_k^T\mathbf{1} = 1$ and $\mathbf{y}_k^T\mathbf{x}_k = 1$, in the sense that $||\mathbf{R}_k^K/\rho_k^K - \mathbf{x}_k\mathbf{y}_k^T||$ converges to zero. Note that $\rho_k > 1$ for $k > \ell$ (as the load of the associated PH/M/1/K+1 queue is less than 1 for $k > \ell$ and $\mathbf{R}_k$ was such that $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i\mathbf{R}_k^{-1}$). This implies that for any $\mathbf{u}, \mathbf{v} > 0$, we have for $k > \ell$

$$\lim_{K\to\infty} \frac{\mathbf{u}^T\mathbf{R}_k^K\mathbf{v}}{\mathbf{u}^T\mathbf{R}_k^K\mathbf{w}} = \lim_{K\to\infty} \frac{\rho_k^K(\mathbf{u}^T\mathbf{x}_k)\mathbf{y}_k^T\mathbf{v} + o(\rho_k^K)}{\rho_k^K(\mathbf{u}^T\mathbf{x}_k)\mathbf{y}_k^T\mathbf{w} + o(\rho_k^K)} = \frac{\mathbf{y}_k^T\mathbf{v}}{\mathbf{y}_k^T\mathbf{w}}, \tag{32}$$

$$\lim_{K\to\infty} \frac{\mathbf{u}^T(\mathbf{I} + \sum_{i=1}^K \mathbf{R}_k^i\mathbf{v})}{\mathbf{u}^T\mathbf{R}_k^K\mathbf{v}} = \lim_{K\to\infty} \frac{\frac{\rho_k^{K+1}-1}{\rho_k-1}(\mathbf{u}^T\mathbf{x}_k)\mathbf{y}_k^T\mathbf{v} + o(\rho_k^K)}{\rho_k^K(\mathbf{u}^T\mathbf{x}_k)\mathbf{y}_k^T\mathbf{v} + o(\rho_k^K)} = \frac{\rho_k}{\rho_k - 1.} \tag{33}$$

□

In solving (28) for z, each bisection step requires, for every object k, computing $(\mathbf{T}_k - z\mathbf{I})^{-1}$ and powers of $\mathbf{R}_k$ (29). A straightforward implementation costs $O\left(\sum_{k=1}^n Kd_k^3\right)$ per bisection iteration. Since $\mathbf{T}_k - z\mathbf{I}$ and $\mathbf{T}_k - z\mathbf{I} + z\mathbf{1}\boldsymbol{\alpha}_k$ are subgenerators, they are non-singular, so their inverses are well-defined.

5 Validations via Trace-Driven Simulations

We validate the accuracy of the mean-field model by comparing its predictions against cache simulations driven by synthetic traces. Two workload settings are considered:
IRM. Zipf(θ) popularity distribution, i.e., $p_k = (1/k^\theta)/\sum_{i=1}^n (1/i)^\theta$.
PH. Keep the same Zipf(θ) popularities and replace the inter-request times with phase-type renewal processes. Here we use a two-phase hyperexponential distribution with $\boldsymbol{\alpha}_k = (1/2, 1/2)$ and $\mathbf{T}_k = \mathbf{T}p_k$, where

$$\mathbf{T} = \begin{bmatrix} -\mu_1 & 0 \\ 0 & -\mu_2 \end{bmatrix},$$

such that $\mu_1/\mu_2 = 10$ and $1/(2\mu_1) + 1/(2\mu_2) = 1$.

For each of the above workload models, we report both pointwise comparisons based on multiple simulation runs and mean absolute errors over miss ratio curves across a range of system parameters.

5.1 Pointwise Comparison and Error

We report pointwise comparisons for a representative set of system sizes, namely $(n, C) \in \{(30, 10), (60, 20), (120, 40), (240, 60), (480, 100), (960, 200)\}$. To capture varying degrees of popularity skew, we consider Zipf exponents $\theta \in \{0.5, 0.8, 1.1\}$, and we evaluate two counter caps, $K = 1$ and $K = 15$.

For each workload setting, MF represents the fixed-point miss rate from the mean-field model, SIM is the average miss rate over 10 simulation runs of trace length 10^7.

For IRM, Table 2 reports the validation results for $K = 15$; similar results for $K = 1$ can be found in Appendix A, Table 5. For PH $K = 1$ and $K = 15$, validation results are shown in Tables 6 and 7, resp. of the same appendix.

Across all experiments, the mean-field predictions match simulation closely. Relative errors are typically well below 1%, with larger deviations appearing mainly at large θ and small n. Accuracy tends to improve as n increases.

	n	30	60	120	240	480	960
	C	10	20	60	40	100	200
$\theta = 0.5$	MF	0.5707	0.5529	0.3720	0.7332	0.6688	0.6627
$\theta = 0.5$	SIM–MF	2.3021e-04	1.2750e-04	7.7574e-06	4.7149e-05	3.6656e-05	3.5027e-05
$\theta = 0.8$	MF	0.4345	0.3990	0.2411	0.5312	0.4526	0.4336
$\theta = 0.8$	SIM–MF	2.9091e-04	1.2780e-04	5.7068e-06	-1.4520e-06	5.5569e-05	1.5947e-05
$\theta = 1.1$	MF	0.2943	0.2460	0.1272	0.3014	0.2262	0.1976
$\theta = 1.1$	SIM–MF	2.1135e-04	1.0898e-04	4.4526e-06	5.3183e-05	-7.3156e-06	-6.3186e-06

Table 2. Validation by simulation of miss probability for IRM model with $K = 15$. Mean-field is nearly always optimistic, i.e., predicts a fractionally smaller miss probability.

5.2 Validations on Miss Ratio Curves

Here we evaluate our mean-field model through a parameter sweep over a multi-dimensional parameter grid, for each workload model and each combination of (n, θ, K). Experiments span $n \in \{30, 60, \ldots, 960\}$ on fixed trace length 10^7, Zipf exponents $\theta \in \{0.5, 0.8, 1.1\}$, and Ran-CLOCK/Ran-SIEVE parameter $K \in \{1, 15, 63\}$. Cache sizes C vary from 2 up to n, and we summarize the resulting accuracy using the mean absolute error (MAE) between model predictions and simulations.

The distribution of MAE values is shown as box plots in Figure 4. For readability, the figure uses labels A–R to denote specific workload models and (θ, K) configurations, as mapped in Table 3.

Table 3. Short-label legend for each experimental configuration. Labels are reused for the MF+SIM miss ratio curves of the same workload.

Label	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Workload					IRM									PH				
θ	0.5	0.5	0.5	0.8	0.8	0.8	1.1	1.1	1.1	0.5	0.5	0.5	0.8	0.8	0.8	1.1	1.1	1.1
K	1	15	63	1	15	63	1	15	63	1	15	63	1	15	63	1	15	63

Again, the mean-field model becomes more accurate as n increases. In Figure 4, the MAEs are below 0.015 for $n = 30$ and typically well below 0.0005 for $n = 960$. We provide conceptual support in Appendix B for the observed accuracy by showing that a slightly modified version of Ran-CLOCK/Ran-SIEVE(K) is $O(1/n)$ accurate.

6 Simulation Results and Guidelines for Policy/Parameter Selection

This section reports trace-driven simulation results and practical guidance for selecting policy variants and K. We examine two workload characteristics, namely *frequency* and *recency*, and the effect thereof on hit ratios and the relative performance ranking of policies.

We first show that changing recency (inter-request time distribution) while keeping frequency (popularity skew θ) fixed can alter policy rankings and narrow (or widen) the performance gap between policies, so conclusions drawn under IRM do not always transfer to renewal processes. Second, we sweep K in controlled experiments to extract simple tuning guidelines. Finally, we evaluate on production traces to assess the robustness of selected policies under real access patterns.

All simulations are run on a modern C++/Python microbenchmark [22].

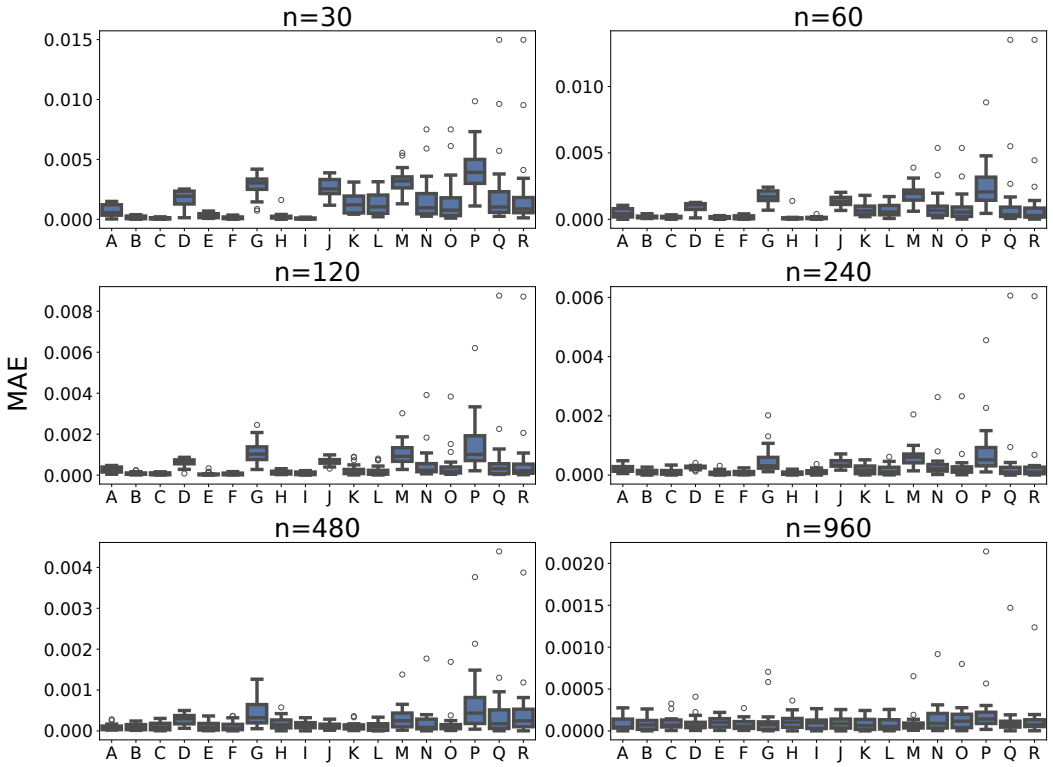


Fig. 4. Mean absolute error (MAE) between model-predicted (MF) and simulated (SIM) miss ratio curves for A-R parameter combinations across multiple n scales. Accuracy tends to improve as n increases.

6.1 Hit Ratio Performance under Markovian Settings

We consider workload variations that decouple *frequency* from *recency*, demonstrating that the same popularity distribution can induce different policy rankings when the inter-request time distribution changes. We use the IRM and PH models defined in Section 5, each with Zipf(θ) popularities (with $\theta \in \{0.8, 1.1\}$) over $n = 120$ objects and trace length 10^7.

We simulate the hit ratio curves of policies described in Section 3.1 and list-based policies RAND(**m**)/FIFO(**m**) from [14]. (For all of our simulations we set the list $\mathbf{m} = [0.5C, 0.5C]$.)

Under IRM with popularity vector (p_k), an upper bound on the hit ratio is given by:

$$H^*_{\text{IRM}}(C) \;=\; \sum_{k \le C} p_k. \tag{34}$$

To better visualize differences across hit ratio curves, we normalize each hit ratio by the IRM upper bound $H^*_{\text{IRM}}(C)$ for the same Zipf exponent θ, as the raw curves are otherwise difficult to distinguish visually. Figure 5 shows the resulting normalized hit ratio curves. Note that several policy pairs are effectively indistinguishable in these experiments: (i) FIFO(**m**) and RAND(**m**) under IRM, (ii) Ran-CLOCK(K) and Ran-SIEVE(K) for all K under IRM and PH, and (iii) Ran-CLOCK(K)/Ran-SIEVE(K) and CLOCK(K) under IRM for sufficiently large K (e.g., K=15).

Under IRM, where i.i.d. requests induce memoryless inter-request times, SIEVE($K = 15$) achieves the best average hit ratio in our experiments, SIEVE($K = 1$) second. Ran-CLOCK/Ran-SIEVE with

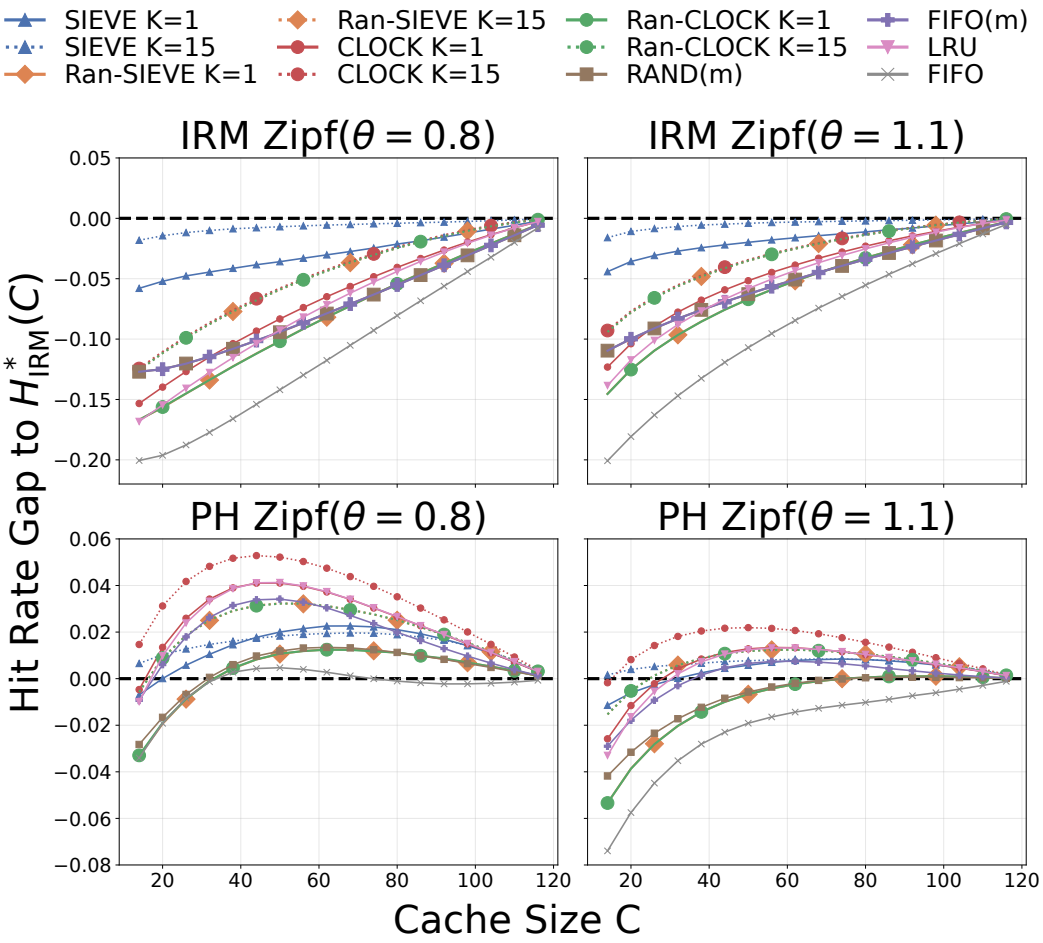


Fig. 5. Simulated hit ratio curves using synthetic workloads; all curves are normalized by the IRM performance upper bound (34) of the same θ for better visualization. Large markers indicate overlapping curves. Under IRM, RAND(**m**) and FIFO(**m**) overlap. For $K = 15$, CLOCK performs the same with Ran-CLOCK/Ran-SIEVE under IRM (red, green, and orange dashed curves overlap). Ran-CLOCK (green) and Ran-SIEVE (orange) overlap in all cases. Note that y scales vary for IRM and PH due to the differing result ranges.

$K = 15$ ranks the third. Increasing θ from 0.8 to 1.1 increases popularity skew and improves hit ratios for most policies (with the caveat that FIFO can behave poorly at very small cache sizes), while the rank among all policies is consistent.

Under PH, popularities stay the same while inter-request times become heavier-tailed via a two-phase construction with $\mu_1/\mu_2 = 10$. Compared to IRM, performance differences across policies are smaller (though this is less visually apparent because the IRM and PH plots use different y-axis scales). As θ increases from 0.8 to 1.1, all policies see lower performance relative to their IRM upper bound, and the differences between them narrow further. Despite these reduced performance gaps, CLOCK(K=15) remains the top performer.

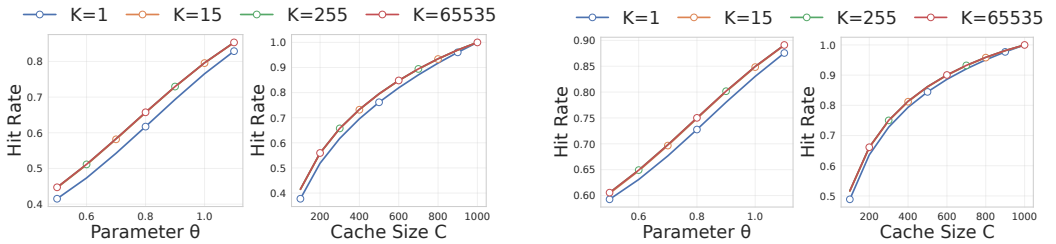
SIEVE is comparatively less θ-sensitive under PH: while its relative hit ratio still slightly decreases with larger θ, its *ranking* improves because competing policies degrade faster.

Finally, K is critical for Ran-CLOCK/Ran-SIEVE: with K=1 they rank near the bottom, ahead of FIFO only, whereas increasing to K=15 lifts them into the top four in our experiments, outperform SIEVE.

Comparing IRM and PH further shows that the sensitivity to hyperparameters can be workload-dependent: larger K can be beneficial under strongly skewed IRM workloads, while under PH the effect of increasing K varies across policies. E.g, it helps CLOCK and Ran-CLOCK/Ran-SIEVE but can be harmful to SIEVE.

6.2 The Effects of K under Markovian Settings

In the previous section, we observed that increasing K can improve the hit ratios of Ran-CLOCK/Ran-SIEVE under the Markovian workload models we study. To better understand this effect, we examine how hit ratios evolve as K increases. We also quantify the overhead of larger K in terms of the mean number of probes per eviction.



(a) IRM (b) PH

Fig. 6. Ran-CLOCK/Ran-SIEVE hit ratios versus K (trace length 10^7, $n = 1000$, $K \in \{1, 15, 255, 65535\}$). In each subfigure, the left panel fixes $C = 300$ and varies $\theta \in \{0.5, 0.6, \ldots, 1.1\}$; the right panel fixes $\theta = 0.8$ and varies $C \in \{100, 200, \ldots, 1000\}$. Curves for $K \geq 15$ are indistinguishable from each other.

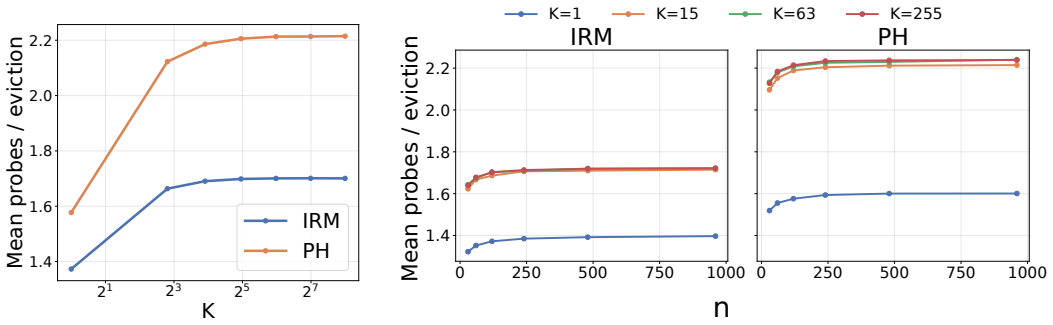
6.2.1 Hit Ratio Evolution over K. Here our experiments span $K \in \{1, 15, 255, 65535\}$, under IRM (Figure 6a) and under PH (Figure 6b), resp. For all cases, trace length and n are fixed at 10^7 and 1000, resp.

Two sets of results are reported under each workload family as visualized in Figure 6. In both subfigures the left panel shows hit ratios at a fixed cache size C=300 (30% of n) over $\theta \in \{0.5, \ldots, 1.1\}$; the right panel fixes θ=0.8 and varies cache size $C \in \{100, 200, \ldots, 1000\}$ (10% to 100% of n).

Across both workload families, hit ratios increase with θ and with cache size C, as expected. They also improve with K, but with clear diminishing returns: most of the gain occurs from $K = 1$ to $K = 15$, while further increases yield only marginal improvements.

6.2.2 Mean Number of Probes per Eviction over K. We further examine the overhead of larger counters by measuring the mean number of probes per eviction under Ran-CLOCK, conditional on a miss. Figure 7a varies K from 1 to 255 (with $n = 120$, $\theta = 0.8$, $C = 24$) under IRM and PH. As K grows from 1, probing becomes more expensive, but the increase quickly saturates, with little change beyond $K \approx 15$. Figure 7b shows that when fixing cache size at $n/5$ and varying n from 30 to 960, the mean number of probes per eviction remains essentially flat once n is moderately large (here $n \gtrsim 250$), indicating that overhead is driven mainly by the cache fraction.

As a quick check against our mean-field model, for $K = 15$, $C = 24$, $\theta = 0.8$, and $n = 120$, the fixed point yields $x_0 \approx 14.19$ under IRM and $x_0 \approx 10.96$ under PH. By (9), the expected number of



(a) Mean probes per eviction versus K ($n = 120$, $\theta = 0.8$, trace length 10^7, $C = 24$, and $K \in \{1, 7, 15, 31, 63, 127, 255\}$).

(b) Mean probes per eviction versus n. Experiments use trace length 10^7, $\theta = 0.8$, $K \in \{1, 15, 63, 255\}$, $n \in \{30, 60, 120, 240, 480, 960\}$ and $C = n/5$.

Fig. 7. Probe cost under Ran-CLOCK across K and n.

probes per miss is approximately $24/14.19 \approx 1.69$ (IRM) and $24/10.96 \approx 2.19$ (PH), which closely match the simulation results in Figure 7.

Overall, the results in this section suggest a simple guideline: modest K (≤ 4 bits) is sufficient to provide most of the hit ratio gains, while larger K adds little overhead under realistic cache provisioning and scaling.

6.3 Real-World Production Traces

Table 4. Trace description.

Trace	volume28	volume766	w11	w44
Source	Alibaba		CloudPhysics	
Length (mil.)	129.61	3.34	296.89	25.26
No. of blocks (mil.)	30.91	0.12	2.99	3.68
Block Size	4096 Bytes			
Year collected	2020		2015	

We next evaluate policies on four production block I/O traces from two open-source trace corpuses: `volume28` and `volume766` (Alibaba) [15], and `w11` and `w44` (CloudPhysics) [21]. These traces were chosen to span diverse access patterns, including two where the relative rankings of CLOCK and SIEVE change (`w11` and `volume766`) and two that exhibit long (`volume28`) or multiple (`w44`) scans and are notably cache-unfriendly.

The traces were preprocessed for simulation. For the CloudPhysics ones, we retain only read requests, and all traces were converted into a one-dimensional sequence of 4096-byte block addresses. Summary statistics after preprocessing are given in Table 4.

Figure 8 shows the hit ratio curves under various policies. We also include Belady's MIN oracle [3] as a roofline. In all panels, cache size varies from 0 up to the catalog size n (in blocks) and is normalized to $(0, 1]$.

SIEVE(K) in most cases is competitive; it achieves the highest mean hit ratio on `w44` and `volume766`, but degrades on `volume28` due to a visible cliff near $C/n \approx 0.6$, and dramatically performs the worst on `w11`. Varying its counter cap provides little benefit in these cases: increasing K is essentially neutral on `w44` and `volume28` and notably harmful on `volume766` and `w11`.

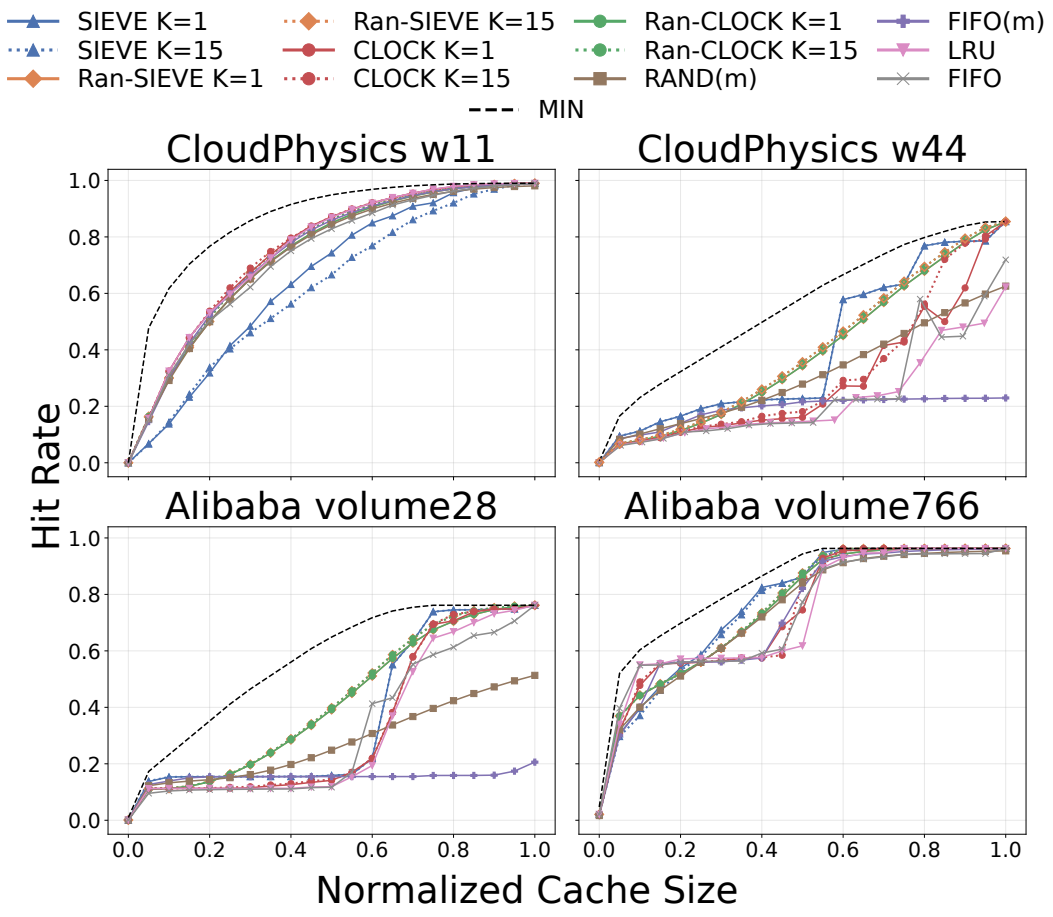


Fig. 8. Simulated hit ratio curves on four production block I/O traces; cache size is normalized by the catalog size n. MIN denotes Belady's oracle. Ran-CLOCK (green) and Ran-SIEVE (orange) overlap in all panels and notably smooth the cliffs. Increasing K has little effect on Ran-CLOCK/Ran-SIEVE, negligible effect on SIEVE for w44 and volume28, and a negative effect on SIEVE for volume766 and w11.

CLOCK(K) performs best on w11: CLOCK(K=15) slightly outperforms CLOCK(K=1) and the remaining policies. However, like SIEVE, LRU and FIFO, CLOCK(K) is not scan-resistant.

RAND(**m**) and FIFO(**m**) underperform throughout due to a naive list setting. While RAND(**m**) typically improves upon FIFO(**m**) (except on w11) because randomization makes it immune to scans.

LRU performs poorly on most of our cases, except for w11, where it slightly exceeds the naive FIFO and most alternatives.

Ran-CLOCK(K)/Ran-SIEVE(K) shows strong and stable performance across traces. It ranks first on volume28 (mean hit ratio 0.4170 versus 0.3614 for the second-best SIEVE(K=15)) and also dominates SIEVE on w11, where it ranks third, slightly behind CLOCK and LRU. On w44 it ranks second and remains close to SIEVE(K=1) (0.3986 vs. 0.4019). The main exception is volume766, where SIEVE leads over a substantial portion of the hit ratio curves, yet the gap between Ran-CLOCK(K)/Ran-SIEVE(K) and SIEVE(K) is modest (at most 10%) and the mean hit ratios remain close (0.7471 vs. 0.7558).

Notably, on `w44` and `volume28` with poor cacheability, Ran-CLOCK(K)/Ran-SIEVE(K) apparently avoids the cliffs seen under other policies. Their hit ratios increase steadily under incremental cache provisioning, indicating robustness under scan-dominated and cache-unfriendly workloads.

7 Why Randomization Mitigates Cliffs

We provide a simple heuristic mean-field approximation explaining why randomization mitigates scan-induced cliffs. Taking RANDOM replacement as a representative of the randomized family, its scan resistance can be examined by studying how its hit probability varies with the ratio of cache size to scan length.

Consider a cache of size C under a repeated scan sequence of length $s > C$, and let P_{miss} denote the long-run miss probability. Under RANDOM replacement, we have approximately

$$\left(1 - \frac{1}{C}\right)^{sP_{\text{miss}}} = 1 - P_{\text{miss}},$$

as with probability $(1-1/C)$ an item survives a random selection and about sP_{miss} random selections take place per cycle (as a hit does not trigger a random replacement).

Now, using $\left(1 - \frac{1}{C}\right)^{sP_{\text{miss}}} \approx (\exp(-s/C))^{P_{\text{miss}}}$, we are looking for a root P_{miss} in $(0, 1)$ of

$$a^{P_{\text{miss}}} = 1 - P_{\text{miss}}, \qquad a = \exp(-s/C),$$

This root can be expressed using the principal branch of the Lambert W function:

$$P_{\text{miss}} = 1 + \frac{W(-be^{-b})}{b}, \qquad b = -\ln a = s/C > 0.$$

Figure 9 shows the heuristic hit probability as a function of C/s in the blue solid curve. Note that this heuristic can be viewed as a mean-field approximation in the sense that randomness is replaced by expectations. We validate this approximation against simulations of a repeated scan over 5000 distinct objects. The warm-up period consists of 200 scans. Measurements are then collected over the next 1000 scans. The simulation results are shown as an orange line with markers in the same figure, with an MAE of 8.5×10^{-5} relative to the model predictions.

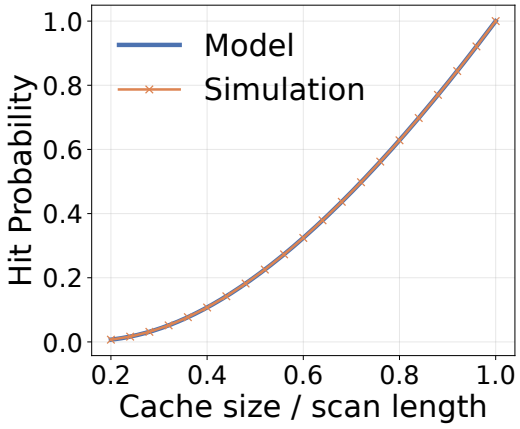


Fig. 9. Modeled vs simulated hit probability (MAE 8.5×10^{-5}) under repeated scans with RANDOM replacement, as a function of the cache-size-to-scan-length ratio C/s.

Note that the curves exhibit a smooth ramp-up as C/s increases, meaning that under RANDOM replacement on workload with repeated scans, incrementally provisioning the cache produces incremental gains in hit ratio rather than an abrupt jump after a critical threshold is crossed. This captures the same hit ratio behavior seen in Figure 8 under randomized replacement policies.

8 Conclusion

We studied Ran-CLOCK/Ran-SIEVE(K), which replace scan-based evictee search in CLOCK/SIEVE(K) with random probing while retaining small per-object state. We developed a heterogeneous mean-field model for both IRM and phase-type renewal processes. We showed that the fixed point used to approximate stationary behavior is characterized by a single scalar parameter, which can be computed efficiently by bisection. We then validated the model against extensive simulations.

Simulations on production workloads show that randomization mitigates scan-induced pathologies, with only modest losses under purely Markovian arrivals. This behavior is further supported by a mean-field approximation of the hit ratio under RANDOM replacement as a function of the cache-to-scan ratio under workloads with repeated scan sequences.

Empirically, we found that policy rankings depend strongly on the interplay between frequency and recency; changes in inter-request time distributions can reorder policies even when popularities are fixed. Across traces, Ran-CLOCK/Ran-SIEVE provides stable and robust performance with steady hit ratio growth as cache size increases, whereas naive CLOCK, SIEVE, LRU, and FIFO can exhibit sharp performance cliffs. Our parameter sweeps further indicate diminishing returns in the counter cap, suggesting that small to moderate values (e.g., $K \leq 15$, i.e., 4 bits) are practical.

As future work, it may be possible to develop a heterogeneous mean-field model for the CLOCK(K) replacement algorithm that uses a scan-based evictee search. This may allow proving structural results between the unique fixed points of the mean-field model for RAN-CLOCK(K) and for the mean-field model of CLOCK(K), resp.

Acknowledgments

We thank the anonymous reviewers for their substantive and thoughtful feedback, and our shepherd for helpful suggestions that improved the paper.

References

[1] S. Allmeier and N. Gast. Mean field and refined mean field approximations for heterogeneous systems: It works! *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6(1):13:1–13:43, 2022.

[2] N. Beckmann and D. Sanchez. Talus: A simple way to remove cliffs in cache performance. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 64–75. IEEE, 2015.

[3] L. A. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Systems journal*, 5(2):78–101, 1966.

[4] D. S. Berger, P. Gland, S. Singla, and F. Ciucu. Exact analysis of ttl cache networks. *Performance Evaluation*, 79:2–23, 2014.

[5] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains*. Oxford University Press, 2005.

[6] A. Blankstein, S. Sen, and M. J. Freedman. Hyperbolic caching: Flexible caching for web applications. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 499–511, 2017.

[7] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *IEEE journal on Selected Areas in Communications*, 20(7):1305–1314, 2002.

[8] F. J. Corbato. A Paging Experiment with the Multics System. Technical Report MAC-M-384, MIT Project MAC, July 1968.

[9] A. Dan and D. Towsley. An approximate analysis of the lru and fifo buffer replacement schemes. In *Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 143–152, 1990.

[10] G. Domingues, G. Mendonça, E. D. S. E. Silva, R. M. Leão, D. S. Menasché, O. Rottenstreich, M. Dehghan, and D. Towsley. The role of hysteresis in caching systems. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 6(1):1–38, 2021.

[11] R. Fagin. Asymptotic miss ratios over independent references. *Journal of Computer and System Sciences*, 14(2):222–250, 1977.
[12] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for lru cache performance. In *2012 24th international teletraffic congress (ITC 24)*, pages 1–8. IEEE, 2012.
[13] N. Gast and B. Van Houdt. Transient and steady-state regime of a family of list-based cache replacement algorithms. In *ACM SIGMETRICS 2015*, 2015.
[14] N. Gast and B. Van Houdt. Transient and steady-state regime of a family of list-based cache replacement algorithms. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 123–136. ACM, 2015.
[15] J. Li, Q. Wang, P. P. Lee, and C. Shi. An in-depth analysis of cloud block storage workloads in large-scale production. In *2020 IEEE International Symposium on Workload Characterization (IISWC)*, pages 37–47. IEEE, 2020.
[16] V. Martina, M. Garetto, and E. Leonardi. A unified approach to the performance analysis of caching systems. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2040–2048. IEEE, 2014.
[17] V. F. Nicola, A. Dan, and D. M. Dias. Analysis of the generalized clock buffer replacement scheme for database transaction processing. In *Proceedings of the 1992 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 35–46, 1992.
[18] L. V. Rodriguez, F. Yusuf, S. Lyons, E. Paz, R. Rangaswami, J. Liu, M. Zhao, and G. Narasimhan. Learning cache replacement with {CACHEUS}. In *19th USENIX Conference on File and Storage Technologies (FAST 21)*, pages 341–354, 2021.
[19] E. Seneta. *Non-negative Matrices and Markov Chains*. Springer-Verlang, New York, 1981.
[20] A. J. Smith. Sequentiality and prefetching in database systems. *ACM Transactions on Database Systems (TODS)*, 3(3):223–247, 1978.
[21] C. A. Waldspurger, N. Park, A. Garthwaite, and I. Ahmad. Efficient {MRC} construction with {SHARDS}. In *13th USENIX Conference on File and Storage Technologies (FAST 15)*, pages 95–110, 2015.
[22] Y. Wang, I. Khor, and P. Desnoyers. 2dio: A cache-accurate storage microbenchmark. *arXiv preprint arXiv:2603.19971*, 2026.
[23] Y. Zhang, J. Yang, Y. Yue, Y. Vigfusson, and K. Rashmi. {SIEVE} is simpler than {LRU}: an efficient {Turn-Key} eviction algorithm for web caches. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1229–1246, 2024.

A Validation Results for Section 5.1

	n	30	60	120	240	480	960
	C	10	20	60	40	100	200
$\theta = 0.5$	MF	0.5989	0.5840	0.4071	0.7536	0.6939	0.6878
$\theta = 0.5$	SIM−MF	-1.9207e-04	8.8524e-05	1.9143e-04	-9.8217e-05	-6.1211e-05	3.8439e-05
$\theta = 0.8$	MF	0.4844	0.4463	0.2796	0.5737	0.4936	0.4729
$\theta = 0.8$	SIM−MF	1.0899e-03	1.1013e-03	7.3245e-04	4.4538e-04	3.4148e-04	2.5574e-04
$\theta = 1.1$	MF	0.3439	0.2876	0.1539	0.3406	0.2578	0.2252
$\theta = 1.1$	SIM−MF	3.4237e-03	2.2831e-03	9.6567e-04	1.0318e-03	4.8686e-04	2.0866e-04

Table 5. Validation by simulation of miss probability for IRM model with $K = 1$. mean-field is nearly always optimistic, i.e., predicts a fractionally smaller miss probability.

B Theoretical Support for the Mean-Field Approximation

We would like to provide conceptual support for the observed accuracy of the mean-field model in Section 5 using the heterogeneous mean-field framework of [1]. Their Theorem 4.1 gives an $O(1/n)$ error bound for n-object CTMCs with (i) bounded interaction degree $d \le d_{\max}$ independent of n, and (ii) interaction rates that admit the normalization

$$q^{(n)}_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x) = \frac{1}{d\, n^{d-1}}\, r^{(n)}_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x), \quad d \le d_{\max}, \tag{35}$$

	n	30	60	120	240	480	960
	C	10	20	60	40	100	200
$\theta = 0.5$	MF	0.4061	0.3972	0.2589	0.5780	0.5138	0.5096
$\theta = 0.5$	SIM–MF	3.1897e-03	1.7490e-03	8.3556e-04	5.1291e-04	1.2413e-04	1.4753e-04
$\theta = 0.8$	MF	0.3360	0.3110	0.1820	0.4442	0.3714	0.3559
$\theta = 0.8$	SIM–MF	3.5011e-03	2.1895e-03	9.6463e-04	9.3499e-04	4.3560e-04	2.3220e-04
$\theta = 1.1$	MF	0.2443	0.2050	0.1024	0.2660	0.1967	0.1719
$\theta = 1.1$	SIM–MF	4.5827e-03	2.7373e-03	9.1606e-04	1.1193e-03	4.7090e-04	1.7684e-04

Table 6. Validation by simulation of miss probability for Zipf-like popularity and hyperexponential inter-request times with $K = 1$ and $\mu_1/\mu_2 = 10$. The mean-field model is always optimistic.

	n	30	60	120	240	480	960
	C	10	20	60	40	100	200
$\theta = 0.5$	MF	0.3996	0.3878	0.2470	0.5690	0.5034	0.4989
$\theta = 0.5$	SIM–MF	1.8039e-03	8.7746e-04	3.2949e-04	3.7862e-04	1.4419e-04	6.8345e-05
$\theta = 0.8$	MF	0.3114	0.2865	0.1635	0.4172	0.3466	0.3321
$\theta = 0.8$	SIM–MF	1.8201e-03	8.4784e-04	1.8701e-04	3.6049e-04	1.9250e-04	1.6499e-05
$\theta = 1.1$	MF	0.2150	0.1801	0.0879	0.2386	0.1754	0.1533
$\theta = 1.1$	SIM–MF	1.4504e-03	6.0305e-04	1.7861e-04	3.0522e-04	5.3843e-05	6.3506e-05

Table 7. Validation by simulation of miss probability for Zipf-like popularity and hyperexponential inter-request times with $K = 15$ and $\mu_1/\mu_2 = 10$. The mean-field model is always optimistic.

where the rate functions $r^{(n)}_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x)$ are uniformly bounded (cf. Equations (2a)–(2c), (3), (4) in [1]). More specifically, Theorem 4.1 in [1] states that the probability that a item k is in state j at time t and the solution $x_{k,j}(t)$ of the mean-field model differ by a term $O(1/n)$ for all t.

To enforce bounded interactions, we introduce a truncated policy Ran-CLOCK/Ran-SIEVE(K, d^*) that caps the scan length at d^* probes; if no zero counter is found within $d^* - 1$ probes, a victim is chosen uniformly at random among all cached items. Under Ran-CLOCK/Ran-SIEVE(K, d^*), each transition updates at most $d_{\max} = d^* + 1$ items. We limit ourselves to the IRM model. The only additional requirement for the phase-type model is that the rates appearing in the matrices $\mathbf{T}_k$ must be uniformly bounded across all items k. This condition holds for instance if the subgenerator matrix $\mathbf{T}_k = p_k\mathbf{T}$ as in our numerical experiments.

The modified mean-field equations due to the truncation are given by

$$\frac{d}{dt}x_{k,K}(t) = p_k x_{k,K-1}(t) - m(t)x_{k,K}(t)\left(\frac{1-p_f(t)}{x_0(t)} + \frac{p_f(t)}{C}\right), \tag{36}$$

$$\frac{d}{dt}x_{k,j}(t) = p_k x_{k,j-1}(t) - p_k x_{k,j}(t) + m(t)\left(x_{k,j+1}(t)\frac{1-p_f(t)}{x_0(t)} - x_{k,j}(t)\left(\frac{1-p_f(t)}{x_0(t)} + \frac{p_f(t)}{C}\right)\right), \tag{37}$$

$$\frac{d}{dt}x_{k,-1}(t) = -p_k x_{k,-1}(t) + m(t)\left(x_{k,0}(t)\frac{1-p_f(t)}{x_0(t)} + \frac{p_f(t)}{C}\sum_{j=0}^{K} x_{k,j}(t)\right), \tag{38}$$

for $j = 0, \ldots, K-1$, with $p_f(t) = (1 - \frac{x_0(t)}{C})^{d^*-1}$ the probability that the first $d^* - 1$ probes fail. To understand the terms involving the factor $1 - p_f(t)$, note that on average the number of failed probes among the first $d^* - 1$ equals

$$\sum_{k=1}^{d^*-1} (1-\frac{x_0(t)}{C})^k = \frac{1 - (1 - \frac{x_0(t)}{C})^{d^*}}{\frac{x_0(t)}{C}} - 1 = \frac{C - x_0(t)}{x_0(t)} - p_f(t)(1-\frac{x_0(t)}{C})\frac{C}{x_0(t)} = \frac{C - x_0(t)}{x_0(t)}(1-p_f(t)).$$

Hence, the $\frac{1}{x_0(t)}$ factors in (4)-(6) needs to be replaced by $\frac{1-p_f(z)}{x_0(t)}$. The terms of the form $p_f(z)/C$ are due to the random d^*-th probe that selects the victim.

We write the microstate as $x = (S_1, \ldots, S_n) \in \mathcal{S}^n$ with local state space $\mathcal{S} = \{-1, 0, 1, \ldots, K\}$, where $S_k = -1$ means "not cached" and $S_k = j \geq 0$ means "cached with counter j." Consider the IRM scaling used in Section 4: item k is requested at rate p_k, with $\sum_k p_k = 1$. Fix an ordered d-tuple $\mathbf{k} = (k_1, \ldots, k_d)$ of distinct items and local state vectors $\mathbf{s}, \mathbf{s}' \in \mathcal{S}^d$ with $d \leq d_{\max}$. Let $q^{(n)}_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x)$ be the CTMC transition rate from microstate x for the jump that updates exactly the items in $\mathbf{k}$ from $\mathbf{s}$ to $\mathbf{s}'$. Changes to the microstate x that only change the state of a single object are called unilateral transitions. These occur under the IRM model when a hit occurs and the hit rate of any item is clearly bounded by $p_1 < 1$, meaning (35) holds for $d = 1$.

For Ran-CLOCK/Ran-SIEVE(K, d^*), any such d-item update, with $d > 1$, can only be triggered by a miss. Without loss of generality we may assume the miss is triggered by item k_1, hence

$$q^{(n)}_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x) := p_{k_1}\, \phi_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x), \tag{39}$$

where $\phi_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x)$ is the conditional probability that, given a request for k_1 in microstate x, the truncated scan probes exactly the items $k_2, \ldots, k_d$ (in some order and some items possibly more than once) and the resulting update maps the pre-scan local states $\mathbf{s}$ to the post-scan local states $\mathbf{s}'$. Note that $v_i = s_i - s'_i > 0$ for $i = 2, \ldots, d$. Let $V = \sum_{i=2}^d v_i \geq d - 1$. Without loss of generality assume item k_2 is the one replaced in the cache, meaning $s'_2 = -1$. If we scale the cache size with the population size by setting $C = \xi n$, with $0 < \xi < 1$, then if the last probe succeeds (that is, the V-th as each failed probe decreases a counter by one)

$$\phi_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x) = \frac{(V-1)!}{(v_2-1)!\prod_{i=3}^d v_i!}\frac{1}{C^V} = \frac{1}{dn^{d-1}}\frac{d(V-1)!}{\xi^{d-1}C^{V-d+1}(v_2-1)!\prod_{i=3}^d v_i!} \leq \frac{1}{dn^{d-1}}\frac{(d^*+1)!}{\xi^{d^*}},$$

as $V \leq d^*$. If the first $d^* - 1$ probes all fail and the victim k_2 was selected at random, we have

$$\phi_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x) = \frac{(d^*-1)!}{(d^*-1-\sum_{i=3}^d v_i)!\prod_{i=3}^d v_i!}\frac{1}{C^{d^*}} \leq \frac{1}{dn^{d-1}}\frac{(d^*+1)!}{\xi^{d^*}},$$

as item k_2 is probed exactly $d^* - 1 - \sum_{i=3}^d v_i$ times in the first $d^* - 1$ probes before it is selected at random during the d^*-th probe.

This shows that if we scale the cache size C with the population size n for the Ran-CLOCK/Ran-SIEVE(K, d^*) algorithm, all interactions involving $d > 1$ items can be written as in (35) with $r^{(n)}_{\mathbf{k},\mathbf{s}\to\mathbf{s}'}(x)$ uniformly bounded by $(d^*+1)!/\xi^{d^*}$.

Received January 2026; revised March 2026; accepted April 2026