Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Modeling Perfect LFU on cache systems using Markov chain

Emese Sziklay

sziklayemese@inf.elte.hu

Eötvös Loránd University

Tamás Jursonovics

Deutsche Telekom (Germany)

Article

Keywords: cache modeling, Markov model, W-TinyLFU, hit rate, Content Delivery Network

Posted Date: March 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8887948/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Scientific Reports on November 3rd, 2026.
See the published version at https://doi.org/10.1038/s41598-026-76717-0.

Modeling Perfect LFU on cache systems using Markov chain

Emese Sziklay[1*] and Tamás Jursonovics[2]

[1*]Department of Data Science, Faculty of Informatics, Eötvös Loránd University, Pázmány Péter stny 1/C, Budapest, 1117, Hungary.
[2]Chapter Streaming & OTT Development, Deutsche Telekom, Deutsche-Telekom-Allee 9, Darmstadt, 64295, Germany.

*Corresponding author(s). E-mail(s): sziklayemese@inf.elte.hu;
Contributing authors: tamas.jursonovics@telekom.de;

Abstract

Accurately estimating the load on the origin server is crucial for the correct and efficient operation of a Content Delivery Network (CDN). Since CDNs rely heavily on caching, simulating caches with arbitrary topologies is essential. This paper presents a methodology for modeling the Perfect Least Frequently Used (PLFU) cache algorithm. While the literature has extensively documented various eviction strategies, including Least Recently Used (LRU), First-In First-Out (FIFO), and Random, there has been a lack of Markov models for frequency based caching algorithms. This work addresses that gap. The PLFU cache algorithm, being a stochastic process, can be represented as a Markov process after the cache has been in operation for a sufficient period. The proposed model enables the estimation of per-object miss rates for both single caches and cache networks, applicable directly to the PLFU algorithm and indirectly to Tiny-LFU strategies. Given that many CDNs employ LFU-like eviction policies, we show that our novel model provides practitioners and decision-makers with a practical tool for designing and deploying CDNs effectively.

Keywords: cache modeling, Markov model, W-TinyLFU, hit rate, Content Delivery Network

1 Introduction

With the rapid growth in online content consumption, the importance of content delivery networks (CDNs) has increased significantly. For efficient CDN management, caching popular content is crucial because it reduces the load on the origin server and user-perceived delays [1] by storing copies of objects in caches close to users. The caching strategy applied plays a crucial role in achieving optimal caching performance. A comprehensive survey by [2] classifies, compares, and presents the main features of existing strategies, along with their respective advantages and disadvantages.

The Least Frequently Used (LFU) cache strategy is classified as a frequency-based algorithm. LFU and its variants, such as Window-LFU [3] and Tiny-LFU [4], are widely deployed in web caching systems [5], and are therefore highly relevant to the modeling of Content Delivery Networks (CDNs).

Despite their practical importance, LFU-like algorithms have not yet been formally modeled in the literature. This work addresses this research gap by developing an analytical model for LFU-based cache replacement strategies. A model that accurately captures the main characteristics of cache algorithms can support the design of energy-efficient infrastructure and promote the sustainability of CDNs [6].

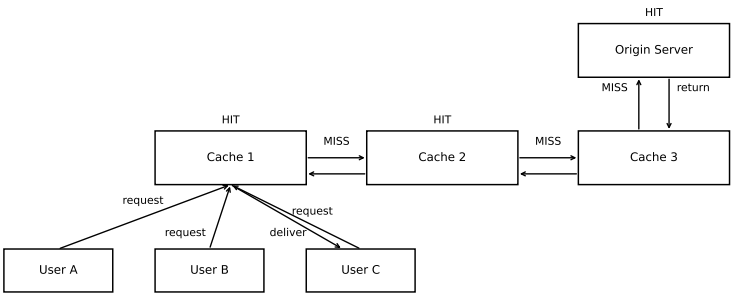


Fig. 1 Multi-cache architecture

Empirical studies [4, 7] have shown that web access patterns typically follow heavy-tailed distributions and are well approximated by Zipf-like or Pareto models, reflecting the high skewness of content popularity. Consequently, the Markov model we developed in this paper is evaluated using request traces generated from a Zipf distribution.

To this end, we generate a series of Zipf-distributed samples with a shape parameter of 1.1 and a wide range of object population sizes N. This setup enables the simulation of caching behavior and facilitates a quantitative comparison between the cache hit ratio (CHR) obtained from simulation and the corresponding CHR predicted by the analytical model. The choice of the Zipf exponent 1.1 is motivated by prior empirical

research, which analyzed user viewing patterns at a major European ISP and found this value to provide the best fit to observed traffic data [8]. The main contributions of this paper are summarized as follows:

- We develop a stochastic model of the Perfect LFU caching policy and show that its dynamics converge to a Markovian regime after a finite warm-up period.
- The proposed Markov model is validated not only on synthetic (Zipf-distributed) data but also on real traffic obtained from a major European ISP.
- We provide a finite-sample analysis and explicitly quantify the number of requests (warm-up period) required before the cache reflects the true popularity ranking.
- Using Hoeffding's and Bernstein's inequalities, we derive non-asymptotic lower bounds on the warm-up time needed to ensure that the top-n most popular items are correctly identified with high probability.
- We establish probabilistic guarantees for both ranking correctness among cached items and protection against cache pollution by less popular items.
- We show that incorporating variance information through Bernstein's inequality yields significantly tighter bounds than Hoeffding-based estimates.
- We support our claims through Monte Carlo simulations, demonstrating close agreement between theory and empirical convergence behavior.
- Our model enables principled CDN dimensioning, ensuring that origin load remains within safe operational limits while preventing unnecessary cache oversizing or redundant deployment.

The remainder of this paper is organized as follows. Section 2 reviews related work, with a focus on analytical models for caching nodes and systems under well-known caching strategies. Sections 3 and 4 present the preliminaries and the mathematical framework required to establish the Markovian properties of the proposed Perfect LFU algorithm. Section 5 provides a detailed description of the analytical model.

The evaluation of the model is presented in Section 6 for both a single cache node and a cache system, and includes a comparative analysis of Perfect LFU, LFU, and W-TinyLFU. Finally, Section 7 summarizes the main contributions of the paper and outlines directions for future work.

2 Related Work

A substantial body of research has emerged over recent decades on the modeling of cache replacement algorithms. Part of this literature focuses on the analysis of a single cache node, as exemplified by [9]. In contrast, other approaches address networks of caches with diverse topologies, as demonstrated in [10] and [11]. In this paper, the proposed Markovian model is applied both to an individual cache node and to a cache network, illustrated in Figure 1.

An overview of analytical techniques used for evaluating cache replacement strategies is provided in [12]. That study surveys a range of tools applied to caching algorithms, including Markovian modeling, and highlights that steady-state analyses are typically intractable for large cache sizes. In contrast, our work demonstrates that

the proposed Markovian framework remains effective even for extremely large systems, accommodating catalog sizes of 10 million objects and cache sizes exceeding 1 million.

Cache replacement algorithms such as FIFO, RANDOM, and CLIMB are analyzed in [13] using a mean-field model, where the stochastic cache dynamics are approximated by a system of deterministic ordinary differential equations (ODEs). Solving these ODEs yields closed-form expressions that approximate the miss probability of each individual item.

Many works such as [10], [11], [9], [14] detail the use of Markov chain to model the Least Recently Used (LRU) cache algorithm. In LRU caching, the state space S has $n+1$ states, $S = \{1, 2, \cdots, n, n+1\}$ where state $i, 1 \leq i \leq n$ represents the item's position in the LRU list of size n, and state $n+1$ indicates that the item is not cached. Such definition for the state space S would not be appropriate for modeling LFU-like cache algorithms because the number of frequencies of an item has been accessed determines the LFU cache policy. Rather, in our Markov model we defined the state space S as the various n-grouped combination of all available objects. In LRU modeling when a request for the item occurs, the object (if present) is moved to position 1; if it is absent so currently is in state $n+1$, it is inserted at position 1 with its request probability. When other objects are requested, the item may either remain in the same state or move down by one position, depending on its relative ordering in the LRU list. These transitions define an irreducible and aperiodic Markov chain for each object, whose stationary distribution gives the probability of occupying each cache position. In contrast, our Markov model developed for the Perfect LFU algorithms results a reducible Markov chain with three disjoint communicating classes as presented in Section 5. In the LRU cache modeling, the stationary probability of state $n+1$ provides the object's miss probability, and the overall cache hit rate can be derived by aggregating these per-object results.

In [9], each item is modeled as a continuous-time Markov chain (CTMC). The paper considers two cases: infinite and finite cache sizes. In the theoretical infinite-cache case, each object is represented by a two-state Markov chain, where states 0 and 1 indicate whether the object is absent from or present in the cache. The transition rates correspond to the object's request rate and expiry rate, respectively. In our model, however, we do not consider object expiry. Instead, the transition probabilities depend solely on the request rate, together with an additional constraint: once the cache warm-up period has elapsed, transitions in which a higher-ranked item would be replaced by a lower-ranked one have zero probability.

For finite, practical caches in [9], each content item at each cache node is modeled by a Markov chain whose states represent the possible positions of the item within the LRU or FIFO stack, together with an additional state denoting that the item is not cached as described above. An iterative CTMC solution is used to estimate the hit probability of each object at time t, progressively refining the transition rates based on the updated state probabilities.

In [10], the LRU cache is modeled in a similar manner using Markov chains, and the stationary distribution is used to determine the miss rate, consistent with the approach in [9] described above. However, [10] and [11] extend the analysis to multi-cache systems. [10] argues that each node's behavior in a cache network depends on

both locally generated requests and requests forwarded from lower-level (child) caches after misses. [11] extends the previous work further to more complex schemes. Our model does not consider such a network topologies because they are not deployed in real CDNs. Rather we analyze our model on multi-caches where the caches are connected sequentially as shown in Figure 1. In [10] the combined inputs define an effective request probability for each node, which then serves as the input to its Markov chain. By propagating the miss probabilities upward through the network, the overall miss rate of the multi-cache system can be derived.

[14] proposes a Markov-chain-based analytical model (MACS) to estimate the hit rate in Content-Centric Networks (CCNs) for the LRU cache replacement algorithm combined with the leave-copy-everywhere (LCE) insertion policy.

[11] builds on the MACS model for LRU, as introduced in [14] and [10]. In [10], the model was extended to incorporate additional insertion policies, such as leave-copy-down (LCD) and $2Q$, the latter being similar to TinyLFU [4], where items must receive at least two accesses before entering the main cache. The work in [11] goes further by generalizing MACS to a caching strategy which is a theoretical scheme that places the most popular contents closest to clients.

The extant literature has primarily focused on estimating the cache hit ratio for eviction algorithms. To the best of our knowledge, our work is the only study that defines a Markov model for the perfect least frequently used (PLFU) eviction algorithm. This model may also serve as a basis for estimating the TinyLFU algorithm, as discussed in Section 7.4.

As discussed above, [11] present their Markov model for the LRU cache algorithm. This journal paper also considers the LFU cache eviction strategy and notes the following:

> The Least Frequently Used (LFU) algorithm, which requires a more complex replacement process, is also interesting. However, it presents several shortcomings related to the insertion of new popular contents or the purge of older ones. Moreover, the complexity of the LFU functioning makes it harder to be modeled and analyzed.

The present paper aims to address this challenge by showing that, although the Markov model developed for Perfect LFU requires an additional condition compared to models for LRU, FIFO, and other algorithms mentioned above, it remains effective for practical purposes, including CDN sizing decisions, while also contributing to considerations of environmental sustainability in CDNs.

3 Preliminaries

Users of a Content Delivery Network (CDN) request various types of content, such as media files and web pages, thereafter referred to as *objects* or *items*. Caching frequently requested objects is a fundamental mechanism for improving overall network performance, as it reduces access latency and bandwidth consumption at the ingress side.

To clearly state the assumptions and prerequisites for the subsequent sections, we summarize below the background model and notation used throughout this paper.

3.1 Request Model and Assumptions

User requests are assumed to follow the *Independent Reference Model* (IRM). Under this model, requests are statistically independent and identically distributed over time, and the probability of requesting a given object remains constant. Furthermore, the content catalog consists of a finite number of N objects.

Formally, at each discrete time instant t, where $t = 0, 1, 2, \ldots$, object $i \in \{1, \ldots, N\}$ is requested with probability p_i. It is also assumed that the popularity distribution of objects follows a Zipf distribution, whose probability mass function is given in (1):

$$p_i = P(X = i) = \frac{i^{-a}}{H_N(a)}, \quad \text{where} \quad H_N(a) = \sum_{k=1}^{N} \frac{1}{k^a} \tag{1}$$

The parameter $a > 0$ characterizes the skewness of the popularity distribution, and $H_N(a)$ denotes the N-th generalized harmonic number.

These modeling assumptions are widely adopted in the literature and have been employed in several prior studies, including [7, 13, 15, 16].

3.2 Cache Model and Notation

We begin by considering a single cache that stores a subset of requested items from a finite catalog of distinct media objects. In Section 7.2, we show that the proposed model readily extends from a single cache node to a cache network.

Let N denote the total number of distinct objects that may be requested. The cache has a finite capacity of n objects, with $n < N$. In practice, cache sizes in CDNs typically range between 10% and 20% of the catalog size N. All objects are assumed to be of identical size. This simplification does not affect the validity of our model, since although objects may have different sizes, their average size does not change over time and can therefore be approximated by a fixed value of 1. Accordingly, cache capacity is measured in terms of the number of objects stored.

At each discrete time step m, the cache is in a state determined by the set of objects currently stored. This is denoted by

$$X(m) = \text{set of objects contained in the cache at time } m.$$

Each incoming request is for an object $i \in \{1, 2, \ldots, N\}$, generated according to the popularity distribution defined in (1). For each request:

- If the requested object is already stored in the cache, a *cache hit* occurs.
- Otherwise, a *cache miss* occurs, and the cache state $X(m)$ is updated according to a replacement policy (e.g., LRU, LFU, FIFO, or Random).

The notation used throughout this paper is summarized in Table 1.

The Perfect LFU (PLFU) cache management process starts from an initial state $X(0)$, in which the cache is completely filled. The PLFU algorithm, as described in [8], maintains two distinct data structures: one that stores the request frequencies of objects currently in the cache (cache-records), and another that records the historical

Table 1 Summary of notation.

Symbol	Description
N	Total number of distinct objects (i.e., the catalog size)
n	Cache capacity (maximum number of objects stored)
$X(m)$	Cache state at time step m
m	Discrete time index (request number)
S	State space (all subsets of $\{1, \ldots, N\}$ of size n)
i	Index of a requested object ($i \in \{1, \ldots, N\}$)
p_i	Request probability of object i
f_i	Request frequency (number of requests for object i)
a	Zipf exponent

frequencies of previously evicted objects (evicted items record). The request frequency f_i of object i represents the total number of times object i has been requested.

During Perfect LFU operation, if the requested object is not found in the cache (i.e., a cache miss occurs), the classic LFU eviction rule is applied: the object with the lowest request frequency is removed, and both data structures are updated accordingly. Specifically, the evicted object's frequency is removed from the cache record and stored in the evicted-items record, while the requested object's frequency is removed from the evicted-items record, incremented by one, and added to the cache record. In the case of a cache hit, the Perfect LFU algorithm behaves identically to the in-memory LFU algorithm by incrementing the requested object's frequency in the cache record.

The cache state process $\{X(m)\}_{m\geq 0}$ forms a stochastic process driven by the random request sequence satisfying the Independent Reference Model (IRM) assumption also used in [16], [17]. The state space S is finite and discrete and consists of all possible subsets of n objects drawn from the catalog of N objects.

The cardinality of the state space is given by the number of combinations of N objects taken n at a time:

$$|S| = \binom{N}{n}.$$

Since the cache cannot store duplicate objects and the order of objects is irrelevant, each cache state corresponds to a unique combination. Thus, the state space can be written as

$$S = \{s_1, s_2, \ldots, s_{\binom{N}{n}}\}.$$

Under the definition above, the PLFU cache management process is *not* a Markov process. This is because the cache state at time m does not depend solely on $X(m-1)$, but also on the entire request history through object frequencies f_i.

In the following section, we introduce additional assumptions and modeling refinements that enable us to represent the PLFU algorithm as a Markov process.

4 Theoretical Foundations

In this section, we develop the theoretical framework required to establish that, after a sufficiently long time horizon, the Perfect LFU (PLFU) cache replacement algorithm

can be treated as a Markov process. Extending the notation summarized in Table 1, we introduce the following assumptions, definitions, and results.

Let $R_1, R_2, \ldots, R_t$ denote a sequence of independent requests drawn from a discrete probability distribution over a catalog of N distinct objects, with ordered probabilities

$$p_1 > p_2 > \cdots > p_N, \qquad \sum_{i=1}^{N} p_i = 1. \tag{2}$$

For a given sample window of length t, define the empirical request frequency of object i as

$$f_i = \sum_{k=1}^{t} \mathbf{1}\{R_k = i\},$$

which satisfies $f_i \sim \text{Binomial}(t, p_i)$ and $\mathbb{E}[f_i] = tp_i$.

We define the probability gaps between the n most popular objects as

$$\Delta_i := p_i - p_n, \quad i < n, \qquad \Delta_{\text{in}} := \min_{1 \le i \le n-1} \Delta_i = p_{n-1} - p_n > 0, \tag{3}$$

and introduce the separation gap between the least popular object stored in the cache and the most popular object outside the cache:

$$\Delta_{\text{out}} := p_n - p_{n+1} > 0. \tag{4}$$

Since $p_1 > p_2 > \cdots > p_N > 0$, both the internal smallest top–n gap and the boundary gap are positive.

We consider the event that all cached objects are requested more frequently than any non-cached object:

$$E \;=\; \left\{ \min_{1 \le i \le n} f_i > \max_{j > n} f_j \right\}, \tag{5}$$

which ensures that the cache contains exactly the set of n most popular objects.

In addition, we define the event that the least popular cached item is correctly identified, i.e., item n attains the smallest request frequency among the true top-n objects:

$$A \;=\; \{\forall i < n : f_i - f_n > 0\}. \tag{6}$$

Together, the events E and A ensure that the set of cached objects coincides with the true set of the n most popular objects, and that the n-th ranked object has the smallest observed request frequency among the cached items. These events play a central role in characterizing the asymptotic behavior of PLFU and in establishing Markovianity after sufficient observation time. The number of requests required until this property holds is referred to as the *warm-up period* of the cache.

Corollary 1 establishes the existence of a finite warm-up period after which the Perfect LFU algorithm can be treated as a Markov process.

Corollary 1 (Almost-Sure Stabilization of Empirical Top-n and Bottleneck Item) *Let* $1 \leq n < N$ *and suppose*

$$p_1 > p_2 > \cdots > p_N > 0.$$

Then, almost surely, there exists a finite time T *such that for all* $t \geq T$*:*

$$\min_{i \leq n} f_i(t) = f_n(t) \tag{7}$$

and the empirical top-n frequency set equals the true top-n frequency set.

Proof We have

$$\Delta_{\text{in}} := \min_{1 \leq i \leq n-1} (p_i - p_n) = p_{n-1} - p_n > 0, \qquad \Delta_{\text{out}} := p_n - p_{n+1} > 0.$$

Set

$$\varepsilon := \frac{1}{4} \min\{\Delta_{\text{in}},\ \Delta_{\text{out}}\} > 0.$$

By the Strong Law of Large Numbers, for each $k = 1, \ldots, N$,

$$\frac{f_k(t)}{t} \xrightarrow{\text{a.s.}} p_k \qquad (t \to \infty),$$

so almost surely there exists a finite T such that for all $t \geq T$ and all k,

$$\left| \frac{f_k(t)}{t} - p_k \right| < \varepsilon.$$

Now consider any $i < n$. Then

$$\frac{f_i(t)}{t} - \frac{f_n(t)}{t} = \left(\frac{f_i(t)}{t} - p_i \right) + (p_i - p_n) + \left(p_n - \frac{f_n(t)}{t} \right) > -\varepsilon + \Delta_{\text{in}} - \varepsilon = \Delta_{\text{in}} - 2\varepsilon > 0,$$

because $\varepsilon < \Delta_{\text{in}}/4$. Hence $f_i(t) > f_n(t)$ for all $i < n$.

Similarly, for any $j > n$,

$$\frac{f_n(t)}{t} - \frac{f_j(t)}{t} \geq -\varepsilon + \Delta_{\text{out}} - \varepsilon = \Delta_{\text{out}} - 2\varepsilon > 0,$$

since $\varepsilon < \Delta_{\text{out}}/4$. Hence $f_n(t) > f_j(t)$ for all $j > n$.

Combining the two displays we obtain, for all $t \geq T$,

$$f_i(t) > f_n(t) > f_j(t) \qquad \forall\ i < n,\ \forall\ j > n.$$

Therefore $f_n(t) = \min_{i \leq n} f_i(t)$ and every top-n empirical frequency exceeds every outsider's frequency; equivalently, the empirical top-n set equals the true top-n set. This completes the proof. □

Next, we determine the length of the warm-up period, i.e., the number of requests required for the Perfect LFU process to enter its Markovian regime.

Our analysis relies on Hoeffding's inequality [18] and Bernstein's inequality [19]. For completeness, these results are stated in the Appendix as two lemmas, which are then used to establish our main results in Theorem 2 and Theorem 3. These theorems provide lower bounds on t that guarantee $\Pr(E \cap A) \geq 1 - \alpha$ where the right-hand side represents the confidence level and the events E and A are defined in (5) and (6), respectively. The Appendix also includes the proofs of these theorems.

Let's examine what the results are for the warm-up period in various catalog sizes, N. The formula obtained for t in Theorem 2 can be used to calculate the number of

Table 2 Cache Warming-Up Times (# of Requests)

N	Hoeffding Bound	Bernstein Bound	Monte Carlo Bound
100	2.24×10^7	2.34×10^6	8.38×10^4
200	5.08×10^8	6.52×10^7	8.75×10^5
400	1.15×10^{10}	1.59×10^9	7.19×10^6
800	2.58×10^{11}	3.61×10^{10}	6.63×10^7
1600	5.69×10^{12}	7.88×10^{11}	6.10×10^8
3200	1.24×10^{14}	1.67×10^{13}	5.61×10^9
6400	2.66×10^{15}	3.49×10^{14}	5.16×10^{10}
10000	1.90×10^{16}	2.45×10^{15}	1.42×10^{11}

requests needed for the cache to separate the frequencies of the top n items from all the others and guarantee at the same time that the empirical frequency of the nth ranked item is the smallest amongst the top n with a confidence level $1 - \alpha = 0.9$ as shown in the first column of Table 2.

We believe that Hoeffding's lower bound for the number of requests obtained in Theorem 2, as derived from Lemma 1, provides a rather conservative estimation of the warm-up period. In other words, the number of requests required for the cache to separate the most popular item is much smaller for zipf-distributed samples.

Remark 1
- The Hoeffding bound (Theorem 2) is simpler but more conservative; Bernstein (Theorem 3) is typically tighter because it incorporates variance.
- Both bounds scale as $t = O\big(\Delta^{-2} \log(\frac{N}{\alpha})\big)$, where Δ denotes the relevant gap ($\Delta_{\min}$ or Δ_{out}). Small gaps require large t.
- To get a numerical t in practice one may split α non-uniformly (e.g. give more budget to the harder event), or compute the two terms in each theorem and choose the maximum.
- These theorems guarantee simultaneously that (i) the true top-n will be the empirical top-n (5), and (ii) among the true top-n the item n remains the empirical minimum (6), with total failure probability at most α.

Estimating the required number of requests via Monte Carlo simulation.

Theorem 2 and Theorem 3 provide rigorous lower bounds on the number of requests t required to recover the true top-n items with high probability. However, these bounds are generally conservative due to the use of union bounds and worst-case inequalities. In practice, the actual number of samples needed to achieve a desired confidence level $1-\alpha$ may be significantly smaller, especially for distributions with uneven probabilities such as Zipf.

To obtain a more realistic estimate of t, we can use a Monte Carlo simulation approach. Specifically, for a fixed number of requests t, we repeatedly draw t independent samples from the target zipf distribution, compute the empirical frequencies $\hat{f}_i$, and check whether the event $A \cap E$ as defined in (5) and (6) occurs.

By performing many repetitions, we can estimate the probability $\Pr(A \cap E)$ as the fraction of trials in which the top-n items are correctly recovered. We then gradually increase t until the estimated probability reaches or exceeds the desired confidence level $1-\alpha$. The pseudo code of this Monte Carlo simulation is provided in Algorithm 1.

Algorithm 1 Monte Carlo Estimation of Sample Complexity

Require: Target probabilities $p_1 > \cdots > p_N > 0$, top-n size n, desired confidence level $1-\alpha$, number of Monte Carlo repetitions R, initial sample size $t_{\min}$, increment Δt

Ensure: Estimated minimal sample size $\hat{t}$ such that $\Pr(A \cap E) \gtrsim 1 - \alpha$

$t \leftarrow t_{\min}$
repeat
  $success \leftarrow 0$
  for $r = 1$ to R **do**
    Draw t i.i.d. samples from $\{1, \ldots, N\}$ according to $\{p_i\}$
    Compute empirical frequencies $\hat{f}_i$ for $i = 1, \ldots, N$
    Let $\mathcal{T} \leftarrow$ indices of the top-n empirical frequencies
    if $\mathcal{T} = \{1, \ldots, n\}$ **then** ▷ Condition A: empirical top-n matches true top-n
      if $\hat{f}_n = \min_{i \in \mathcal{T}} \hat{f}_i$ **then** ▷ Condition E: n-th item minimal among top-n
        $success \leftarrow success + 1$
      end if
    end if
  end for
  $\hat{P} \leftarrow success/R$ ▷ estimated probability of $A \cap E$
  if $\hat{P} < 1 - \alpha$ **then**
    $t \leftarrow t + \Delta t$
  end if
until $\hat{P} \geq 1 - \alpha$
return $\hat{t} \leftarrow t$

This Monte Carlo procedure provides a practical and data-driven way to estimate the empirical sample complexity, complementing the theoretical bounds derived via Hoeffding's and Bernstein's inequalities.

In our experiments, we set the number of Monte Carlo trials to $R = 20{,}000$ and the confidence level to $1 - \alpha = 0.9$. We chose an initial warm-up time $t_{\min}$ based on the heuristic

$$t_{\min} = \frac{n}{p_n},$$

where p_n denotes the request probability of the n-th ranked item, and n is the cache size. Note that n was chosen as a fraction of the catalog size, specifically $n = 0.15\,N$, so we did not evaluate all combinations of n and N.

The value of $t_{\min}$ serves only as a starting point for the search; in practice, the actual warm-up time t is expected to be larger. Algorithm 1 incrementally increases t until the following two conditions are satisfied in at least $1 - \alpha$ fraction of the trials:

1. The empirical top-n items exactly match the true top-n items, formaulated in (5).
2. The n-th ranked item is the minimum among the empirical counts of the top-n items, formulated in (6).

The resulting warm-up times are summarized in Table 2 and illustrated in Figure 2. Monte Carlo simulations show that initializing the algorithm at $t_{\min}$ reliably yields a warm-up period in which the empirical counts of the top-n items exceed those of all remaining catalog items, and the nth-ranked item in the cache has the lowest frequency among the top n. This corresponds to the event $A \cap E$ defined in Theorem 2 and Theorem 3. Across different catalog sizes N, the procedure consistently attains the target confidence level $(1-\alpha)$, confirming its effectiveness as a practical choice of warm-up duration for heavy-tailed workloads. One may question whether there exists a less conservative theorem than Theorem 2 and Theorem 3 that provides a lower bound closer to that obtained via Monte Carlo simulation. Addressing this question will be the subject of our future work.

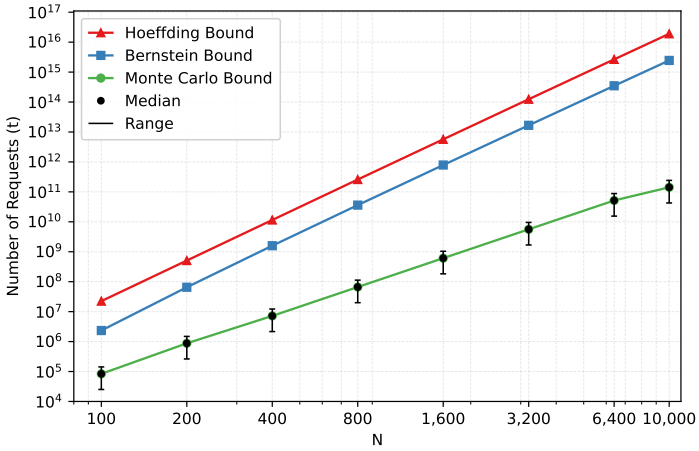


Fig. 2 Cache Warming-Up Times: Theoretical Bounds vs Monte Carlo at 0.9 confidence level

In this setting, assuming that m exceeds either the theoretical bounds of Theorem 2 or Theorem 3, or the practical bound obtained by Algorithm 1, the Perfect LFU cache algorithm can be accurately modeled as a Markovian process.

Specifically, the theorems above provide a $1-\alpha$ guarantee that, even though the frequencies of the cached top-n items may not be internally ordered, and the remaining items may also be unordered, no outside object can overtake a cached one once $m \geq t$. Consequently, upon each new arrival, only the n-th ranked item may be evicted. Therefore, the set of cached contents at time m depends only on the previous cache state, implying the Markov property:

$$P(X_{m+1} = j \mid X_m = i, X_{m-1} = i_{m-1}, \ldots, X_0 = i_0) = P(X_{m+1} = j \mid X_m = i). \quad (8)$$

Equation (8) expresses that the conditional distribution of the next cache state depends only on the current configuration and not on the entire history [20]. The warm-up time therefore marks the point after which Perfect LFU behaves as a stationary Markov chain.

Since the state space S is finite, the process $\{X(m)\}_{m \geq t}$ constitutes a discrete-time Markov chain with transition matrix T [21, 22], where each entry P_{ij} represents the probability of transitioning from state i to state j.

5 Methods

As previously indicated and summarized in Table 1, the state space S of a Perfect LFU cache with capacity n operating on a catalog of N objects consists of all unordered subsets of size n drawn from $\{1, 2, \ldots, N\}$. Thus,

$$|S| = \binom{N}{n}.$$

We index objects by decreasing popularity; that is, object 1 is the most frequently requested item, object 2 the second most frequent, and so on until object N, the least popular. Let $p_1 > p_2 > \cdots > p_N$ denote the corresponding request probabilities.

Each cache state is represented by an ordered tuple

$$(i_1, i_2, \ldots, i_n), \quad \text{with } i_1 < i_2 < \cdots < i_n,$$

which indicates the set of cached objects sorted by rank. In what follows, we present the Markov model of the Perfect LFU cache for a small example with $N = 4$ and $n = 2$. The associated transition matrix T is

$$T = \begin{array}{c} \\ (1,2) \\ (1,3) \\ (1,4) \\ (2,3) \\ (2,4) \\ (3,4) \end{array} \begin{array}{c} \begin{array}{cccccc} (1,2) & (1,3) & (1,4) & (2,3) & (2,4) & (3,4) \end{array} \\ \begin{pmatrix} p_1+p_2 & p_3 & p_4 & 0 & 0 & 0 \\ p_2 & p_1+p_3 & p_4 & 0 & 0 & 0 \\ p_2 & p_3 & p_1+p_4 & 0 & 0 & 0 \\ p_1 & 0 & 0 & p_2+p_3 & p_4 & 0 \\ p_1 & 0 & 0 & p_3 & p_2+p_4 & 0 \\ 0 & p_1 & 0 & p_2 & 0 & p_3+p_4 \end{pmatrix} \end{array} \tag{9}$$

Matrix T is indexed by cache states, and T_{ij} denotes the probability—conditional on the warm-up period having elapsed—that the cache transitions from state i to state j at the next request. For example, the value $p_1 + p_2$ in the first row and column indicates that when the cache contains objects $(1, 2)$, and either object 1 or object 2 is requested, the cache state remains unchanged. The remaining nonzero entries in the first row represent the probabilities that object 3 or 4 is requested, causing the least-frequently used object to be replaced.

As seen in (9), many entries of the matrix are identically zero. In particular, the entire upper-right 3×3 block vanishes, reflecting the fact that states $(2, 3)$, $(2, 4)$, and

$(3,4)$ cannot be reached from any of $(1,2)$, $(1,3)$, or $(1,4)$. This sparsity arises from the frequency-ordering property established in Corollary 1. Specifically, once the cache has run beyond the warm-up time t, the condition

$$f_2(m) > f_3(m) \quad \text{for all } m \geq t$$

holds with high probability. By Corollary 1, this implies

$$f_1(m) > f_3(m), \quad f_1(m) > f_4(m), \quad f_2(m) > f_4(m),$$

so that objects 3 and 4 can never overtake objects 1 or 2. Consequently, once objects 1 and 2 are cached, no lower-ranked object can enter the cache. Hence, all transitions that would insert an inferior object into a cache configuration containing strictly superior ones are assigned probability zero.

For example, in the fourth row of matrix T, the probability of a transition from $(2,3)$ to $(1,3)$ is zero, since this would require item 3 to be retained while evicting the more popular item 2, which contradicts the LFU eviction rule once the frequency ordering has stabilized.

More generally, from any state $(i_1, \ldots, i_n)$ sorted by rank, the cache may only transition to states that differ by replacement of the object with the largest index. Thus, each row of the transition matrix contains at most $N - n + 1$ nonzero entries, corresponding to (i) a cache hit, which leaves the state unchanged, or (ii) replacement of the least popular cached object.

The structure of T also reveals that the Markov chain is *reducible*. In the present example, the state space decomposes into three disjoint communicating classes. In general, the chain decomposes into multiple classes indexed by the smallest-ranked element of each state.

Formally, the full state space may be enumerated as

$$\begin{aligned} &(i_1, i_2, \ldots, i_n),\ (i_1, i_2, \ldots, i_{n+1}),\ \ldots,\ (i_1, i_2, \ldots, i_N), \\ &(i_1, i_3, \ldots, i_{n+1}),\ \ldots,\ (i_{N-n+1}, \ldots, i_N), \end{aligned} \tag{10}$$

where each tuple denotes a sorted selection of n objects. Among these, the states of the form

$$(i_1, i_2, \ldots, i_n),\ (i_1, i_2, \ldots, i_{n+1}),\ \ldots,\ (i_1, \ldots, i_N) \tag{11}$$

form the recurrent (closed) communicating class [20, 23]. All other states are transient.

Accordingly, the transition matrix admits the canonical decomposition

$$T = \begin{bmatrix} T_1 & 0 \\ A & P \end{bmatrix}, \tag{12}$$

where T_1 governs transitions among recurrent states, P governs the transient class, and A encodes transitions from transient to recurrent states.

For the example in (9), the submatrices are given explicitly by

$$T_1 = \begin{bmatrix} p_1+p_2 & p_3 & p_4 \\ p_2 & p_1+p_3 & p_4 \\ p_2 & p_3 & p_1+p_4 \end{bmatrix}, \quad A = \begin{bmatrix} p_1 & 0 & 0 \\ p_1 & 0 & 0 \\ 0 & p_1 & 0 \end{bmatrix},$$

and

$$P = \begin{bmatrix} p_2+p_3 & p_4 & 0 \\ p_3 & p_2+p_4 & 0 \\ p_2 & 0 & p_3+p_4 \end{bmatrix}.$$

This decomposition reflects the long-term behavior of the Perfect LFU cache: once the system enters the recurrent class corresponding to the truly most popular objects, it will remain there indefinitely with probability one.

5.1 Steady State of the PLFU Cache Management

In the previous section, it was assumed that the PLFU cache algorithm has been operating for a sufficiently long time, that is, beyond the warm-up period. Consequently, for the request probability distribution with

$$p_1 > p_2 > \cdots > p_N > 0.$$

the uniform–separation property stated in Corollary 1 applies, implying that the empirical frequencies satisfy

$$f_i(m) > f_j(m) \quad \text{for all } i \le n,\ j > n, \quad \text{and } \min_{i \le n} f_i(t) = f_n(t) \text{ for all } m \ge t,$$

with high probability. Under these conditions, the cache dynamics evolve as a time-homogeneous Markov chain.

A time-homogeneous Markov chain admits a stationary distribution $\mathbf{s}$ satisfying

$$\begin{aligned} \mathbf{s}\,T &= \mathbf{s}, \\ \mathbf{s} = [s_1, s_2, \ldots, s_{\binom{N}{n}}], \qquad \sum_{i=1}^{\binom{N}{n}} s_i &= 1. \end{aligned} \tag{13}$$

Transposing (13) yields

$$T^T \mathbf{s} = \mathbf{s}, \tag{14}$$

which implies that $\mathbf{s}$ is the right eigenvector of T^T corresponding to the unity eigenvalue.

As established previously, the transition matrix T admits the canonical decomposition

$$T = \begin{bmatrix} T_1 & 0 \\ A & P \end{bmatrix},$$

where T_1 represents transitions within the recurrent class, P corresponds to transient states, and A captures transitions from transient to recurrent states. Transposing and substituting into (14) gives

$$\begin{bmatrix} T_1^T & A^T \\ 0 & P^T \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix}. \tag{15}$$

Expanding the matrix equation in (15), we obtain the equivalent relations in (16)–(17).

$$T_1^T \mathbf{s}_1 + A^T \mathbf{s}_2 = \mathbf{s}_1, \tag{16}$$

$$P^T \mathbf{s}_2 = \mathbf{s}_2. \tag{17}$$

Matrix P is a sub-stochastic transition matrix associated with transient states. Therefore, the spectral radius of P^T is strictly less than one [24], which implies that (17) admits only the trivial solution $\mathbf{s}_2 = \mathbf{0}$. Consequently, (16) reduces to

$$T_1^T \mathbf{s}_1 = \mathbf{s}_1. \tag{18}$$

Thus, the stationary distribution $\mathbf{s}$ is of the form

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{0} \end{bmatrix}. \tag{19}$$

where $\mathbf{s}_1$ is the eigenvector of T_1^T associated with eigenvalue one.

Since T_1 represents a closed irreducible class, the Perron-Frobenius theorem guarantees [24] that this eigenvector is unique and strictly positive. Therefore, the stationary distribution assigns nonzero probability exclusively to recurrent states, while all transient states have zero steady-state probability.

As a result, the effective dimension of the state space is reduced from $\binom{N}{n}$ to $N - n+1$. Accordingly, the original transition matrix T reduces to a $(N-n+1)\times(N-n+1)$ matrix T_1 capturing the long-term cache dynamics.

For illustration, consider $N = 10{,}000$ objects and cache size $n = 1{,}500$. The full state space contains $\binom{10000}{1500} \approx 7.085 \times 10^{1834}$ states, which is computationally intractable. After reduction, the stationary distribution is instead computed from an 8501×8501 eigenvector problem for the unity eigenvalue, which is feasible with numerical methods. We also evaluated a catalog of 10 million objects and a cache of 1.5 million; steady state was reached in less than 2.5 hours on our server, demonstrating practicality for real-world CDN deployment.

6 Findings

The Markov model developed in the previous section yields the steady-state distribution over all feasible cache configurations. Due to the structural properties of the PLFU algorithm after the warm-up period, it follows that only $N - n + 1$ states have

nonzero stationary probability, where N denotes the number of objects and n the cache size. These states correspond precisely to the recurrent class identified in (11); all remaining states are transient and thus have zero probability in the steady state.

To compute the miss probability of each object, we exploit the resulting stationary distribution. We illustrate the procedure using the example in (9), where $N = 4$ and $n = 2$. In this case, the recurrent states are $(1, 2)$, $(1, 3)$, and $(1, 4)$, with steady-state probabilities $\pi_{(1,2)}$, $\pi_{(1,3)}$, and $\pi_{(1,4)}$, respectively.

The miss probability of an object equals its request probability multiplied by the probability that the object is not present in the cache. Because object 1 belongs to every recurrent state, we immediately have

$$P(\text{miss of object } 1) = 0.$$

Object 2 is absent from states $(1, 3)$ and $(1, 4)$; hence,

$$P(\text{miss of object } 2) = \big(\pi_{(1,3)} + \pi_{(1,4)}\big)\, p_2 = \big(1 - \pi_{(1,2)}\big)p_2.$$

Proceeding analogously,

$$P(\text{miss of object } 3) = \big(1 - \pi_{(1,3)}\big)p_3, \qquad P(\text{miss of object } 4) = \big(1 - \pi_{(1,4)}\big)p_4,$$

where p_i denotes the request probability of object i.

For a catalog of N objects and cache size n, the recurrent states are

$$(1, 2, \ldots, n-1, n),\ (1, 2, \ldots, n-1, n+1),\ \ldots,\ (1, 2, \ldots, n-1, N), \tag{20}$$

with steady-state probabilities $\pi_{(1,2,\ldots,n-1,i)}$ for $i = n, \ldots, N$. Consequently, the first $n - 1$ objects never incur a miss,

$$P(\text{miss of object } i) = 0, \qquad i = 1, \ldots, n-1.$$

For objects $i \geq n$,

$$P(\text{miss of object } i) = \big(1 - \pi_{(1,2,\ldots,n-1,i)}\big)\, p_i, \qquad i = n, \ldots, N, \tag{21}$$

where $\pi_{(1,2,\ldots,n-1,i)}$ is obtained from the Markov steady state in (20). Expression (21) provides the closed form miss probability of object i. In the CDN setting, the server load follows by summing (21) over all objects.

7 Experiments

7.1 Single-Node Cache Simulation

To validate the Markov model developed in Section 5, we first simulate a single Perfect LFU (PLFU) cache. For each catalog size $N \in \{100, 200, 400, 800\}$, independent request traces were generated according to a Zipf distribution with exponent 1.1.

Each trace contains a sufficiently large number of requests to exceed the warm-up period associated with the given catalog size N and cache size n. The duration of the warm-up period was determined using the Monte Carlo simulation described in Algorithm 1. Although only Zipf-distributed workloads are considered in the simulations, the analytical model applies to any monotone request distribution, as guaranteed by Corollary 1.

The Perfect LFU algorithm was simulated for each catalog size and cache fraction. During the warm-up period (the first half of each trace), the PLFU cache recorded item request frequencies, while hit and miss events were not counted. After the warm-up period expired (the second half of the trace), hit and miss events were accumulated, effectively modeling a cache operating in steady state, as is typical for long-running CDN deployments.

For each simulation run and cache fraction, we computed the per-object miss probability. Figure 3 compares the simulation results with the analytical predictions of the Markov model derived in (21). The results demonstrate excellent agreement between simulation and analysis across all considered values of N and cache fractions.

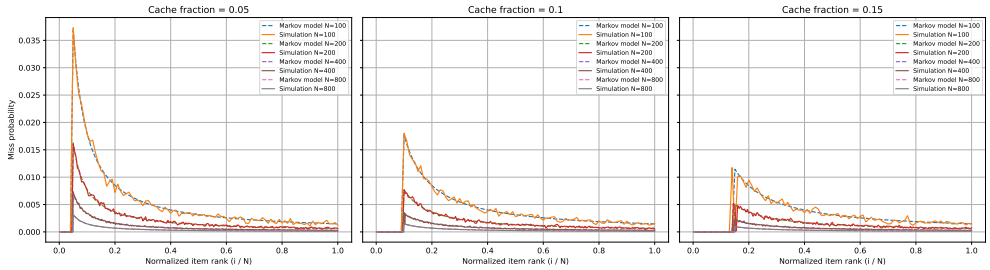


Fig. 3 PLFU simulation versus Markov model predictions for single-node cache.

Figure 4 shows the same results after vertical normalization by the Riemann integral over $[0, 1]$, demonstrating that the model consistently captures the relative miss probabilities across all sample sizes.

7.1.1 Cache Hit Ratio Comparisons

Extending the previous set of catalog sizes to
$N \in \{100, 200, 400, 800, 1600, 3200, 6400, 10{,}000\}$, and considering the same cache fractions of 5%, 10%, and 15% of the catalog size, we computed the cache hit ratio (CHR) for the simulated LFU and TinyLFU cache algorithms. These results were then compared with the CHR predicted by our Markov model, as illustrated in Figure 5.

Figure 5 confirms that the CHRs predicted by the Markov model closely match those obtained from a single-sample simulation. In fact, across all cache fractions, the difference between the TinyLFU hit ratio and the Markov model prediction decreases with increasing catalog size and stabilizes for large N. For $N \geq 3200$, the TinyLFU–Markov gap remains consistently small, typically within 2–3% across all

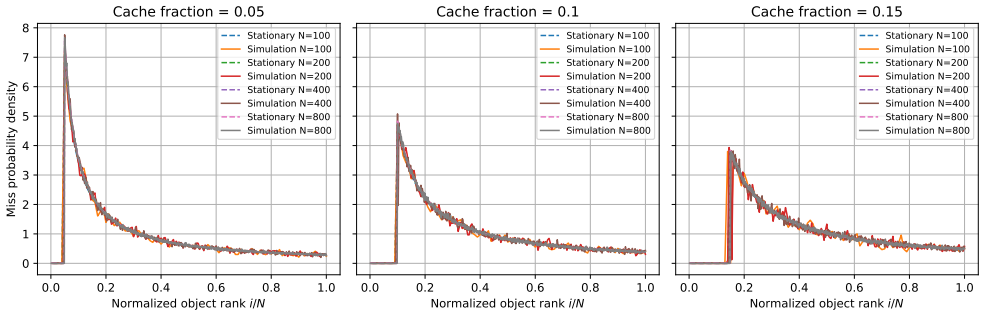


Fig. 4 Normalized PLFU simulation and Markov model predictions for single-node cache.

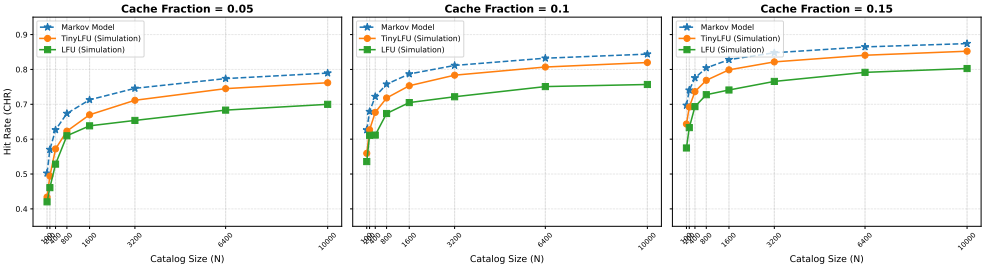


Fig. 5 Cache Hit Rate Comparison: Simulation vs Markov Model Predictions

cache fractions. This indicates that the Markov model provides a stable and predictable approximation of TinyLFU performance in large-scale regimes. Since the Markov model slightly overestimates the cache hit ratio, it should be interpreted as an optimistic benchmark. In practice, when using the Markov model to estimate the cache hit ratio of a real CDN deployment where TinyLFU (or its variants) is employed, applying a simple correction—such as reducing the predicted hit ratio by approximately 5%—yields a robust and realistic estimate. This margin safely accounts for the observed discrepancy and enables reliable performance evaluation and capacity planning in operational CDN environments.

7.2 Multi-Cache System Simulation

We further evaluate the Markov model in a three-level cache system, as illustrated in Figure 1. A single Zipf-distributed request trace over 100 objects, with exponent 1.1, is generated and sequentially processed by caches of sizes 5, 10, and 15. The miss stream produced by each cache is used as the input to the subsequent cache in the hierarchy.

For each cache level, we compute the per-object miss probabilities and compare them with the corresponding predictions of the Markov model. The results of this comparison are shown in Figure 6.

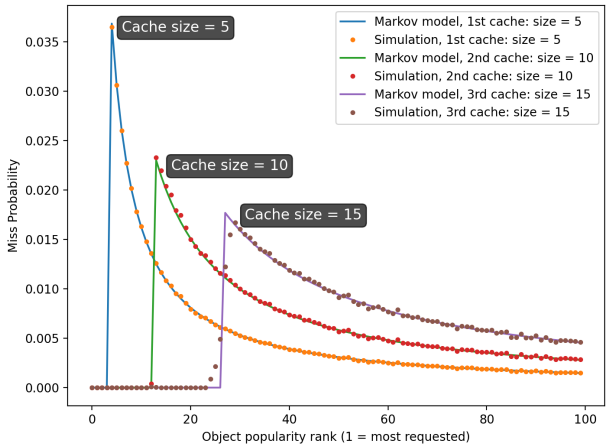


Fig. 6 PLFU miss simulation versus Markov model predictions for a multi-cache system.

To model this system analytically, the first cache is treated in the same manner as in the single-node simulation. For the second cache, the input request distribution is constructed from the normalized miss probabilities of the first cache. This procedure is repeated for the third cache.

7.3 Validate the model on real media objects

So far, we have demonstrated that the Markov model developed in Section 5 provides an excellent fit to the per-object miss probability on synthetic data, and, as shown in Section 7.2, is capable of accurately estimating the hit rate of the entire CDN.

To validate the model in a realistic setting, we were provided with aggregated and normalized popularity statistics of TV channels over a two-hour period by a major European ISP. The data consist solely of rank-order popularity values and do not include any individual-level access logs or personally identifiable information. The resulting rank-order probability distribution is shown in Fig. 7. In this context, the media objects correspond to TV channels, identified by channel numbers. As noted in Section 1, such data is well approximated by a Zipf distribution with shape parameter 1.1 [8]. Accordingly, we used this parameter when generating Zipf-distributed samples to validate the Markov model developed for the Perfect LFU cache algorithm.

Naturally, other parameter values—or even different probability distributions—could also be used, provided they satisfy (2), since the consequences of Corollary 1 imply that the Perfect LFU cache algorithm can be modeled by the proposed Markov model, as detailed in Section 4.

We evaluated cache sizes of 10, 15, 20, 25, and 30, with the results summarized in Table 3. The close agreement between the Markov model predictions and the empirical Perfect LFU results confirms the accuracy of the proposed model when applied to real-world data.

Table 3 Cache Hit Rate Comparison on Real Data.

Cache Size	Markov Model	Perfect LFU
10	0.5572	0.5674
15	0.6561	0.6639
20	0.7276	0.7340
25	0.7865	0.7911
30	0.8324	0.8359

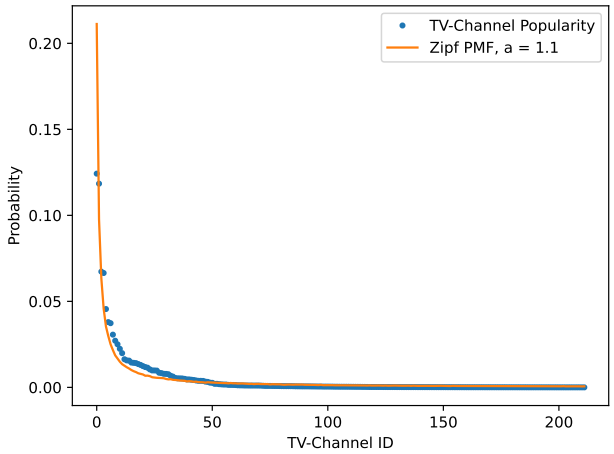


Fig. 7 Empirical rank-order distribution of TV channel popularity versus the theoretical Zipf PMF with parameter $a = 1.1$

7.4 Using our model on W-TinyLFU

Given the widespread use of the Window-TinyLFU (W-TinyLFU) cache algorithm in Content Delivery Networks (CDNs), it is relevant to examine the potential application of the developed Markov model in this context. TinyLFU and its variant, W-TinyLFU, serve as practical benchmarks, as [25] notes that the open-source Caffeine Java 8 cache [26] implements W-TinyLFU, which has been deployed—either directly or via Caffeine—in storage systems such as Cassandra and LinkedIn's feed. This prevalence highlights the importance of considering W-TinyLFU when evaluating and validating the Markov model against realistic CDN workloads.

PLFU has been shown to outperform traditional in-memory LFU in terms of hit rate; however, as the number of objects grows, PLFU's advantage diminishes: its cache hit ratio approaches that of LFU, and the metadata overhead becomes a storage burden [8]. Comparing PLFU, LFU, and W-TinyLFU provides context for the Markov model's performance. Figure 5 shows that W-TinyLFU hit rates consistently lie slightly below those of PLFU and the Markov model, though the gap narrows

with increasing object count. For a realistic CDN scenario with 10,000 objects, the deviation is only 2.3%, indicating that the Markov model yields an optimistic yet reliable estimate. Assuming a modest 5% underestimation of hit rates provides a safe approximation of origin server load, supporting efficient CDN design without risking overload.

8 Conclusion

This paper develops a rigorous analytical framework for modeling frequency-based caching policies by introducing a steady-state Markov chain formulation of the Perfect LFU (PLFU) algorithm. We show that once empirical access frequencies stabilize, the resulting transition matrix admits a canonical decomposition that reduces the state space from $\binom{N}{n}$ possible cache configurations to only $N - n + 1$ recurrent states. This structural reduction enables exact steady-state analysis even for extremely large content catalogs, including those with tens of millions of objects.

Using this framework, we derive closed-form expressions for per-object miss probabilities and overall cache hit ratios, and validate the analytical results through extensive simulations and real-world data obtained from a major European ISP. Beyond PLFU, we empirically demonstrate that the proposed model yields conservative yet practically meaningful performance estimates for classical LFU and the TinyLFU algorithm under realistic Zipf-distributed workloads. As a result, the framework serves not only as an exact analytical tool for PLFU, but also as a reliable upper bound on the hit rate—or equivalently, a lower bound on the miss rate and network load—for modern LFU-based caching systems deployed at scale, such as content delivery networks (CDNs).

In addition, we investigate the cache warm-up period, defined as the number of requests required for access frequencies to stabilize. We establish two conservative theoretical lower bounds on the warm-up time using concentration-based arguments, and complement them with Monte Carlo estimates. Across all parameter regimes considered, the empirical warm-up times are several orders of magnitude smaller than the theoretical bounds. As summarized in Table 2, moderate configurations exhibit gaps of a few orders of magnitude, while even in large-scale settings—where theoretical bounds reach 10^{15}–10^{16}—the observed stabilization time remains limited to approximately 10^{11} requests. These results highlight the conservative nature of the existing theoretical guarantees.

Several directions for future work naturally follow from this study. First, an important theoretical challenge is to develop tighter analytical bounds on the cache warm-up period that more accurately reflect the empirical stabilization behavior, improving upon results such as Theorem 3. Second, while the present framework provides provable bounds on hit and miss rates, future research will focus on refining these estimates by deriving tighter bounds or closed-form expressions for cache load. In particular, extending the analysis to multi-cache systems and real-world CDN deployments—including architectures based on Varnish cache technology and the TinyLFU caching strategy—remains an important and practically relevant direction.

Declarations

This study was funded by T-Labs.

Appendix A Hoeffding's Inequality

Lemma 1 (Hoeffding's Inequality) Let $X_1, X_2, \ldots, X_t$ be independent random variables such that $a_m \leq X_m \leq b_m$ almost surely for each m. Define the sample mean $\bar{X}_t = \frac{1}{t}\sum_{m=1}^{t} X_m$ and let $\mu = \mathbb{E}[\bar{X}_t] = \frac{1}{t}\sum_{m=1}^{t} \mathbb{E}[X_m]$. Then, for all $\varepsilon > 0$,

$$\Pr(\bar{X}_t - \mu \geq \varepsilon) \leq \exp\left(-\frac{2t^2\varepsilon^2}{\sum_{m=1}^{t}(b_m - a_m)^2}\right),$$

Appendix B Bernstein's Inequality

Lemma 2 (Bernstein's Inequality) Let $X_1, X_2, \ldots, X_t$ be independent random variables and define their centered versions

$$Y_m = X_m - \mathbb{E}[X_m], \qquad m = 1, \ldots, t.$$

Assume that each Y_m is almost surely bounded as

$$|Y_m| \leq M.$$

Let the total variance be

$$\sigma^2 = \sum_{m=1}^{t} \mathrm{Var}(X_m).$$

Define the sample mean $\bar{X}_t = \frac{1}{t}\sum_{m=1}^{t} X_m$ and its expectation

$$\mu = \mathbb{E}[\bar{X}_t] = \frac{1}{t}\sum_{m=1}^{t} \mathbb{E}[X_m].$$

Then, for all $\varepsilon > 0$,

$$\Pr(\bar{X}_t - \mu \geq \varepsilon) \;\leq\; \exp\left(-\frac{t\varepsilon^2}{2(\sigma^2/t) + \frac{2}{3}M\varepsilon}\right),$$

Appendix C Hoeffding bound

Theorem 2 (Hoeffding combined bound) *Fix a total failure probability $\alpha \in (0,1)$. If*

$$t \;\geq\; \max\left\{\frac{2}{\Delta_{\text{in}}^2}\ln\Big(\frac{2(n-1)}{\alpha}\Big),\;\; \frac{2}{\Delta_{\text{out}}^2}\ln\Big(\frac{2(N-n)}{\alpha}\Big)\right\},$$

then with probability at least $1-\alpha$ both

$$\min_{1\leq i\leq n} f_i > \max_{j>n} f_j \qquad \textit{and} \qquad f_n = \min_{1\leq i\leq n} f_i$$

hold (equivalently, the empirical top-n equals the true top-n, and item n is the empirical minimum of the top-n).

Proof Split the total failure probability as $\alpha = \alpha_{\text{top}} + \alpha_{\text{out}}$; for simplicity pick $\alpha_{\text{top}} = \alpha_{\text{out}} = \alpha/2$.

(1) No higher-ranked top item falls below item n. For fixed $i < n$ define

$$D_i := f_i - f_n = \sum_{k=1}^{t} Y_k^{(i)}, \qquad Y_k^{(i)} := \mathbf{1}\{\text{trial } k \text{ is } i\} - \mathbf{1}\{\text{trial } k \text{ is } n\}.$$

Each $Y_k^{(i)} \in [-1, 1]$ and $\mathbb{E}[D_i] = t\Delta_i$. By Hoeffding's inequality for bounded summands,

$$\Pr(D_i \le 0) = \Pr\left(D_i - \mathbb{E}[D_i] \le -t\Delta_i\right) \le \exp\Big(-\frac{(t\Delta_i)^2}{2t}\Big) = \exp\Big(-\frac{t\Delta_i^2}{2}\Big).$$

Union-bounding over $i = 1, \ldots, n-1$ and using $\Delta_i \geq$
$Delta_{\text{in}}$ gives

$$\Pr\left(\exists i < n : \ D_i \le 0\right) \le (n-1)\exp\Big(-\frac{t\Delta_{\text{in}}^2}{2}\Big).$$

Requiring this to be at most $\alpha_{\text{top}} = \alpha/2$ yields the first condition

$$t \ge \frac{2}{\Delta_{\text{in}}^2} \ln\Big(\frac{2(n-1)}{\alpha}\Big).$$

(2) No outsider overtakes a top-n item. It suffices to ensure the largest outsider f_{n+1} is below the threshold halfway between the means of f_n and f_{n+1}. Let

$$\theta := \frac{t(p_n + p_{n+1})}{2} = t\Big(p_{n+1} + \frac{\Delta_{\text{out}}}{2}\Big).$$

For any outsider $j > n$ we have $p_j \le p_{n+1}$ and, by Hoeffding applied to sum of Bernoulli indicators,

$$\Pr(f_j \ge \theta) = \Pr\left(f_j - tp_j \ge \theta - tp_j\right) \le \exp\Big(-\frac{2(\theta - tp_j)^2}{t}\Big) \le \exp\Big(-\frac{2(t\Delta_{\text{out}}/2)^2}{t}\Big) = \exp\Big(-\frac{t\Delta_{\text{out}}^2}{2}\Big).$$

Union-bounding over all $N - n$ outsiders gives

$$\Pr\left(\exists j > n : \ f_j \ge \theta\right) \le (N-n)\exp\Big(-\frac{t\Delta_{\text{out}}^2}{2}\Big).$$

Requiring this to be at most $\alpha_{\text{out}} = \alpha/2$ yields the second condition

$$t \ge \frac{2}{\Delta_{\text{out}}^2} \ln\Big(\frac{2(N-n)}{\alpha}\Big).$$

Combining the two conditions and taking the maximum completes the proof: with the displayed t both failure events have probability at most $\alpha/2$, so the union has probability at most α and thus $\Pr(E \cap A) \ge 1 - \alpha$. □

Appendix D Bernstein bound

Theorem 3 (Bernstein Combined Bound) *Fix a significance level $\alpha \in (0, 1)$. Let $p_1 > p_2 > \cdots > p_N$ with $\sum_{i=1}^{N} p_i = 1$, and let the cache store $n < N$ items. Define*

$$\Delta_{\text{in}} := p_{n-1} - p_n, \qquad \Delta_{\text{out}} := p_n - p_{n+1},$$

$$V_{\text{top}} := \max_{i<n} \left(p_i + p_n - (p_i - p_n)^2\right), \quad V_{\text{out}} := \max_{j>n} p_j(1 - p_j) \le p_{n+1}(1 - p_{n+1}).$$

If the number of requests t satisfies

$$t \ \ge \ \max\left\{ \frac{\ln \frac{2(n-1)}{\alpha}\left(2V_{\text{top}} + \frac{2}{3}\Delta_{\text{in}}\right)}{\Delta_{\text{in}}^2}, \ \frac{\ln \frac{2(N-n)}{\alpha}\left(2V_{\text{out}} + \frac{2}{3}\Delta_{\text{out}}\right)}{\Delta_{\text{out}}^2} \right\},$$

then the probability that the top-n items are correctly ranked and no outsider overtakes a cached item satisfies

$$\Pr(A \cap E) \ge 1 - \alpha.$$

Proof We split the total error probability as

$$\alpha = \alpha_{\text{top}} + \alpha_{\text{out}},$$

and assign $\alpha/2$ to each event via the union bound.

(1) Preservation of order among the top-n items.

Fix an index $i < n$. For trial $k = 1, \dots, t$, define the comparison random variable

$$Y_k^{(i)} := X_k^{(i)} - X_k^{(n)},$$

where $X_k^{(j)}$ is the indicator that request k is for item j. Then $Y_k^{(i)} \in \{-1, 0, 1\}$ and

$$\mathbb{E}[Y_k^{(i)}] = p_i - p_n =: \Delta_i.$$

Define the cumulative difference

$$D_i := \sum_{k=1}^{t} Y_k^{(i)}.$$

By linearity of expectation,

$$\mathbb{E}[D_i] = t\Delta_i.$$

Since the trials are independent and identically distributed under the independent reference model, the variables $Y_k^{(i)}$ are i.i.d., hence

$$\text{Var}(D_i) = \sum_{k=1}^{t} \text{Var}(Y_k^{(i)}) = t\ \text{Var}(Y_1^{(i)}).$$

A direct calculation gives

$$\text{Var}(Y_1^{(i)}) = \mathbb{E}[(Y_1^{(i)})^2] - (\mathbb{E}[Y_1^{(i)}])^2 = p_i + p_n - (p_i - p_n)^2.$$

Define the worst-case variance among top-n comparisons:

$$V_{\text{top}} := \max_{i<n} \big(p_i + p_n - (p_i - p_n)^2\big).$$

Thus for all $i < n$,

$$\text{Var}(D_i) \le tV_{\text{top}}.$$

Applying Bernstein's inequality with $s = t\Delta_i$, we obtain

$$\Pr(D_i \le 0) = \Pr(D_i - \mathbb{E}[D_i] \le -s) \le \exp\Big(-\frac{s^2}{2\,\text{Var}(D_i) + \frac{2}{3}s}\Big) \le \exp\Big(-\frac{t\Delta_i^2}{2V_{\text{top}} + \frac{2}{3}\Delta_i}\Big).$$

Let

$$\Delta_{\text{in}} := \min_{i<n} \Delta_i = p_{n-1} - p_n.$$

Applying a union bound over all $i = 1, \dots, n-1$, the probability that any top item falls below item n is at most $\alpha/2$ provided

$$t \ge \frac{\ln\big(\frac{2(n-1)}{\alpha}\big)\big(2V_{\text{top}} + \frac{2}{3}\Delta_{\text{in}}\big)}{\Delta_{\text{in}}^2}.$$

(2) Protection against outsiders.

Fix an item $j > n$ and define

$$X_j := \sum_{k=1}^{t} Z_k^{(j)},$$

where $Z_k^{(j)}$ indicates whether request k is for item j. Then

$$\mathbb{E}[X_j] = tp_j, \qquad \mathrm{Var}(X_j) = tp_j(1-p_j).$$

Define the uniform variance bound over outsiders:

$$V_{\mathrm{out}} := \max_{j>n} p_j(1-p_j) \le p_{n+1}(1-p_{n+1}).$$

Set the threshold

$$\theta := t\Big(p_{n+1} + \frac{\Delta_{\mathrm{out}}}{2}\Big) = \frac{t(p_n + p_{n+1})}{2}, \quad \Delta_{\mathrm{out}} := p_n - p_{n+1}.$$

Then $\theta - tp_j \ge t\Delta_{\mathrm{out}}/2$.

Applying Bernstein's inequality yields

$$\Pr(X_j \ge \theta) \le \exp\left(-\frac{(\theta - tp_j)^2}{2tV_{\mathrm{out}} + \frac{2}{3}(\theta - tp_j)}\right) \le \exp\left(-\frac{t\Delta_{\mathrm{out}}^2}{2(2V_{\mathrm{out}}) + \frac{2}{3}\Delta_{\mathrm{out}}}\right).$$

A union bound over the $N-n$ outsiders gives total error probability at most $\alpha/2$ provided

$$t \ge \frac{\ln\big(\frac{2(N-n)}{\alpha}\big)\big(2V_{\mathrm{out}} + \frac{2}{3}\Delta_{\mathrm{out}}\big)}{\Delta_{\mathrm{out}}^2}.$$

Finally, requiring both bounds to hold yields

$$\Pr(A \cap E) \ge 1 - \alpha,$$

which concludes the proof.

□

References

[1] Davison, B. D. A web caching primer. *IEEE internet computing* **5**, 38–45 (2001).

[2] Podlipnig, S. & Böszörmenyi, L. A survey of web cache replacement strategies. *ACM Computing Surveys (CSUR)* **35**, 374–398 (2003).

[3] Karakostas, G. & Serpanos, D. N. Doe, J. (ed.) *Exploitation of different types of locality for web caches.* (ed.Doe, J.) *Proceedings of the 7th International Symposium on Computers and Communications (ISCC 2002)*, 207–212 (IEEE, 2002).

[4] Einziger, G., Friedman, R. & Manes, B. Tinylfu: A highly efficient cache admission policy. *ACM Transactions on Storage (ToS)* **13**, 1–31 (2017).

[5] Karakostas, G. & Serpanos, D. Practical lfu implementation for web caching. *Technical Report TR-622-00* (2000).

[6] Zhou, Q., Wang, S., Ma, X. & Xu, W. Digital technologies and corporate green innovation: opening the "black box" of resource orchestration mechanisms. *Sustainability Accounting, Management and Policy Journal* **15**, 884–912 (2024). URL https://doi.org/10.1108/SAMPJ-09-2023-0639.

[7] Hasslinger, G., Heikkinen, J., Ntougias, K., Hasslinger, F. & Hohlfeld, O. Klein, A. & Wunder, G. (eds) *Optimum caching versus lru and lfu: Comparison and combined limited look-ahead strategies.* (eds Klein, A. & Wunder, G.) *Proceedings of the 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2018)*, 1–6 (IEEE, 2018).

[8] Sziklay, E. & Jursonovics, T. Illés, Z., Verma, C., Gonçalves, P. J. S. & Singh, Y. (eds) *Energy efficiency of cache eviction algorithms for zipf-distributed objects.* (eds Illés, Z., Verma, C., Gonçalves, P. J. S. & Singh, Y.) *Proceedings of International Conference on Recent Innovations in Computing*, 231–244 (Springer Nature Singapore, Singapore, 2025).

[9] Gomaa, H., Messier, G. G., Williamson, C. & Davies, R. Estimating instantaneous cache hit ratio using markov chain analysis. *IEEE/ACM transactions on Networking* **21**, 1472–1483 (2012).

[10] Ben-Ammar, H., Hadjadj-Aoul, Y., Rubino, G. & Ait-Chellouche, S. Bennis, M. & Gesbert, D. (eds) *A versatile markov chain model for the performance analysis of ccn caching systems.* (eds Bennis, M. & Gesbert, D.) *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2018)*, 1–6 (IEEE, 2018).

[11] Ben-Ammar, H., Hadjadj-Aoul, Y., Rubino, G. & Ait-Chellouche, S. On the performance analysis of distributed caching systems using a customizable markov chain model. *Journal of Network and Computer Applications* **130**, 39–51 (2019).

[12] Hasslinger, G., Okhovatzadeh, M., Ntougias, K., Hasslinger, F. & Hohlfeld, O. An overview of analysis methods and evaluation results for caching strategies. *Computer Networks* **228**, 109583 (2023).

[13] Gast, N. & Van Houdt, B. Transient and steady-state regime of a family of list-based cache replacement algorithms. *Queueing Systems* **83**, 293–328 (2016).

[14] Ammar, H. B., Chellouche, S. A. & Aoul, Y. H. Karakostas, G. & Serpanos, D. N. (eds) *A markov chain-based approximation of ccn caching systems.* (eds Karakostas, G. & Serpanos, D. N.) *Proceedings of the IEEE Symposium on Computers and Communications (ISCC 2017)*, 327–332 (IEEE, 2017).

[15] Rosensweig, E. J., Menasche, D. S. & Kurose, J. Greenberg, A. & Medard, M. (eds) *On the steady-state of cache networks.* (eds Greenberg, A. & Medard, M.) *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM 2013)*, 863–871 (IEEE, 2013).

[16] Breslau, L., Cao, P., Fan, L., Phillips, G. & Shenker, S. On the implications of zipf's law for web caching (1998).

[17] Hasslinger, G., Heikkinen, J., Ntougias, K., Hasslinger, F. & Hohlfeld, O. Klein, A. & Wunder, G. (eds) *Optimum caching versus lru and lfu: Comparison and*

combined limited look-ahead strategies. (eds Klein, A. & Wunder, G.) *Proceedings of the 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt 2018)*, 1–6 (IEEE, 2018).

[18] Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* **58**, 13–30 (1963).

[19] Boucheron, S., Lugosi, G. & Massart, P. *Concentration Inequalities: A Nonasymptotic Theory of Independence* (Oxford University Press, 2013). URL https://doi.org/10.1093/acprof:oso/9780199535255.001.0001.

[20] Ross, S. M. *Introduction to probability models* (Academic press, 2014).

[21] Wilkov, R. Analysis and design of reliable computer networks. *IEEE Transactions on Communications* **20**, 660–678 (1972).

[22] Dehling, H. G., Gottschalk, T. & Hoffmann, A. C. *Stochastic modelling in process technology* (Elsevier, 2007).

[23] Gebali, F. & Gebali, F. Reducible markov chains. *Analysis of Computer Networks* 157–189 (2015).

[24] Horn, R. A. & Johnson, C. R. *Matrix analysis* (Cambridge university press, 2012).

[25] Einziger, G., Eytan, O., Friedman, R. & Manes, B. Sullivan, M. & Krothapalli, N. (eds) *Adaptive software cache management.* (eds Sullivan, M. & Krothapalli, N.) *Proceedings of the 19th International Middleware Conference*, 94–106 (ACM, 2018).

[26] Manes, B. Caffeine: A high performance caching library for java 8. https://github.com/ben-manes/caffeine (2016). Accessed: 2025-12-16.