Fundamentals of Caching Layered Data objects

Agrim Bari and Gustavo de Veciana
Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX, USA
Email: {agrim.bari, deveciana}@utexas.edu

George Kesidis
School of Electrical Engineering and Computer Science
The Pennsylvania State University
State College, PA, USA
Email: kesidis@gmail.com

***Abstract*—The effective management of the vast amounts of data processed or required by modern cloud and edge computing systems remains a fundamental challenge. This paper focuses on cache management for applications where data objects can be stored in layered representations. In such representations, each additional data layer enhances the "quality" of the object's version, albeit at the cost of increased memory usage. This layered approach is advantageous in various scenarios, including the delivery of zoomable maps, video coding, future virtual reality gaming, and layered neural network models, where additional data layers improve quality/inference accuracy. In systems where users or devices request different versions of a data object, layered representations provide the flexibility needed for caching policies to achieve improved hit rates, i.e., delivering the specific representations required by users. This paper investigates the performance of the Least Recently Used (LRU) caching policy in the context of layered representation for data, referred to as Layered LRU (LLRU). To this end, we develop an asymptotically accurate analytical model for LLRU. We analyze how LLRU's performance is influenced by factors such as the number of layers, as well as the popularity and size of an object's layers. For example, our results demonstrate that, in the case of LLRU, adding more layers does not always enhance performance. Instead, the effectiveness of LLRU depends intricately on the popularity distribution and size characteristics of the layers.**

***Index Terms*—Least Recently Used, Layered representations, Working set approximation.**

I. Introduction

Managing shared edge caching. Efficient management of shared memory systems for applications that require large volumes of data remains a significant challenge. These challenges are exacerbated when mobile applications with latency constraints leverage limited/costly edge caching resources but have limited or variable connectivity to the network edge. In such settings, ensuring that data is available when needed becomes even more critical.

Layered representations and applications. This paper focuses on applications where data objects can be stored and utilized in multiple versions, represented as *Layered Representations (LRs)*. Each version offers a trade-off between resource requirements (e.g. size) and the quality delivered to applications. LRs incrementally improve quality with each additional layer, providing flexibility for systems with diverse quality demands or resource limitations. LRs have a wide range of applications. In zoomable maps, they enable for varying levels of topographic detail. In scalable video coding, a base layer provides basic quality, while enhancement layers add finer details. For Virtual Reality (VR), progressive meshes [12] allow efficient storage and rendering of 3D models through hierarchical simplification. Similarly, in compact Neural Networks (NNs), a base model delivers lower accuracy, while additional layers improve weight fidelity and inference accuracy [14], [15].

Exploiting layered representations. Applications may request different versions of a data object for various reasons. First, limited computational resources on the end devices may necessitate versions requiring less compute, memory, or energy. Second, network conditions, such as low bandwidth, may favor versions that balance quality with efficient transmission. Third, applications may not always require the highest quality; for example, in VR gaming, a distant tree can be represented with a less detailed model. These scenarios demonstrate the flexibility of Layered Representations (LRs) in addressing diverse application requirements and constraints.

Alternative representations. There are alternative approaches to representing different versions of data objects, such as *Multiple Representations (MRs)* with or without transcoding [10], [19], [20]. In MR without transcoding, discrete and independent versions are created; for instance, in video streaming, this corresponds to encoding videos at different rates without layering. In contrast, MR with transcoding stores only the highest-quality version, from which lower-quality versions can be derived. Transcoding can be performed in real-time (online), generating lower versions on demand, or offline, where lower versions are precomputed for future requests.

LRU with LR. While considerable attention has been paid to the design and analysis of caching policies for MR, limited attention has been paid to the LR setting, which, as mentioned above, we expect to be of increasing relevance to emerging applications and caching at the network edge. In this paper, we focus on a disciplined study of LRU, which has been redesigned to leverage layered representation for data objects.

A. Related work

We review the caching policies relevant to our work. The foundational studies are summarized in [5], [13], and [11] provides analytical methods for performance evaluation. Caching policies aim to maximize the *cache hit rate*, the fraction of requests served directly from the cache.

The *Least Recently Used (LRU)* policy retains recently accessed objects, capitalizing on temporal locality. Under the Independent Reference Model (IRM), where data objects are requested independently of previous requests, analytical results for LRU include the invariant distribution for uniformly sized objects [16] and the working-set approximation [6], [7]. The working-set approximation, introduced in [6], provides a method to compute the cache hit rate, with its asymptotic accuracy formally proven in [7]. In this paper, we extend this analysis to layered representations.

Beyond the IRM model, competitive analyzes compare online policies, which lack future knowledge, to Belady's offline optimal policy [3]. LRU's competitive ratio scales linearly with cache size B. To improve LRU, e.g., the *Marker* algorithm [8] achieves a competitive ratio of $2H_B \approx 2\log(B)$, where H_B is the B-th harmonic number. No online policy can outperform H_B. Recent work [17] shows that combining Marker with future request predictions can further reduce the competitive ratio, depending on the accuracy of the prediction. The idea of segmented caches, which have in the past been used to more efficiently evict "one-hit wonders" (e.g., [18]), has more recently been combined with low-complexity FIFO policies [21]. In the following, we extend classical non-segmented LRU and leave to future work adaptations of our approach to segmented caching policies. An extended version of this work, which includes additional experiments and discussions, is available on arXiv [2].

B. Paper contributions and organization

The key contributions of this paper are now summarized. First, we redesign and analyze the Least Recently Used (LRU) caching policy for systems where data objects are available in *layered representations (LR)*. The proposed *Layered LRU (LLRU)* policy is simple, robust to dynamic request patterns, and well suited for caching in LR-based systems. Second, we introduce a new working-set approximation to compute hit probabilities for data objects under the LLRU policy, assuming an *Independent Reference Model (IRM)* for versioned data object requests. We demonstrate the asymptotic accuracy of this approximation for a fixed number of layers. Finally, we investigate the sensitivity of LLRU's performance to the size and popularity of layers and data object versions. This analysis provides insights into determining the optimal number of layers and identifying scenarios where additional layers enhance performance.

The paper is organized as follows. Section II presents the system model and the working set approximation for LLRU, along with a proof of its asymptotic accuracy. In Section III, we empirically validate the claims from Section II. Finally, Section IV concludes the paper.

II. System Model and analysis

A. Model for cache

The system consists of a cache server of capacity B bytes. The server stores various versions of data objects to serve near-future requests from a user population. Due to practical constraints, the cache capacity is typically not enough to store all versions of data objects.

B. Model for data objects and arrival requests

Let $\mathcal{D}$ be the set of data objects, with cardinality $D = |\mathcal{D}|$. Each data object $d \in \mathcal{D}$ can have multiple versions, all with the same total number of versions, indexed by $v \in \{1, 2, \ldots, V\}$, where V is the total number of versions. The arrival rate of requests for version v of data object d is denoted by $\lambda(d, v)$. The total arrival rate of requests for data object d is given by $\lambda(d) = \sum_{v=1}^{V} \lambda(d, v)$, and the overall arrival rate of requests from all users is $\lambda = \sum_{d=1}^{D} \lambda(d)$. The probability of a request for data object d is $q(d) = \lambda(d)/\lambda$, and the probability of a request for version v of data object d is $q(d, v) = \lambda(d, v)/\lambda$. Next, we describe our model for layered representations.

C. Model for Layered representations

In LRs, version v of a data object is represented by a set of consecutive layers $l \in \{1, 2, \ldots, v\}$, where the size of layer l for data object d is denoted by $\delta(d, l)$. Consequently, version v of data object d occupies a total space of $s_{\mathrm{LR}}(d, v) = \sum_{l=1}^{v} \delta(d, l)$ in the cache. We define $\gamma(d, l) = \sum_{v=l}^{V} \lambda(d, v)$ as the total request rate for layer l of data object d, accounting for requests for all versions that include this layer. The request probability for layer l of data object d is given by $p(d, l) = \gamma(d, l)/\lambda = \sum_{v=l}^{V} q(d, v)$.

D. Layered Least Recently Used (LLRU):

Property of LLRU. If layer $l + 1$ of a data object is present in the cache, then all lower layers $i \in \{1, 2, \ldots, l\}$ must also be in the cache.

LLRU manages the cache by evicting the least recently accessed layers of data objects. Let $a(d, l)$ denote the last access time of layer l for data object $d \in \mathcal{D}$. When an LLRU cache processes a request for (d, v) at time t, it operates as follows: a) During a cache hit/miss - Update $a(d, l) = t$ for all layers $l \leq v$, ensuring lower layers are accessed after higher layers, and b) If a cache miss occurs, evict layers of other cached data objects in increasing order of their last access times until sufficient space is available to store all layers $l \leq v$ of data object d.

Instead of using access times, the LLRU cache-eviction order can be maintained by using a doubly-linked list.

E. Working-set approximation for LLRU

We introduce a working-set approximation for the LLRU policy when data objects are stored in LRs. The accuracy of this approximation is demonstrated in two ways: analytically, as the number of data objects approaches infinity, and empirically, through simulations.

Consider a system where time is divided into discrete slots. We assume that the request arrival process for each data object d and layer l follows a Bernoulli process with parameter $p(d, l)$. This means the probability of a request for data object d and layer l in a time slot is $p(d, l)$, and such events occur independently across time slots.

1) Characteristic Time: Suppose there is a request for data object d and layer l at time zero. Let $T_f(i,k)$ denote the time of the first request for data object $i \neq d$ and layer k, where $k \in \{1,2,\ldots,V\}$. Similarly, let $T_n(d,m)$ denote the time of the next request for data object d and layer m, where $m \leq l$. Under the Bernoulli arrival process, these times are geometrically distributed, i.e., $T_f(d,l) \sim \text{Geo}(p(d,l))$ and $T_n(d,l) \sim \text{Geo}(p(d,l))$.

At time $t > 0$, the total size of requested data objects and layers up to time t, excluding requests for data object d and layer l, is given by:

$$S_{-(d,l)}(t) = \sum_{\substack{i=1\\ i\neq d}}^{D} \sum_{k=1}^{V} \delta(i,k) \mathbf{1}\left\{T_f(i,k) < t\right\} + \sum_{k=1}^{l-1} \delta(d,k) \mathbf{1}\left\{T_n(d,k) < t\right\}, \quad (1)$$

where $\delta(i,k)$ represents the size of layer k for data object i.

The characteristic time $T_{-(d,l)}(B)$, a random variable, is defined as the minimum time $t > 0$ at which the working-set size excluding data object d and layer l exceeds B:

$$T_{-(d,l)}(B) = \min\{t > 0 : S_{-(d,l)}(t) \geq B\}. \quad (2)$$

A request for data object d and layer l at time $T_n(d,l)$ is a cache hit if the working-set size remains below B, i.e., $S_{-(d,l)}(T_n(d,l)) < B$, or equivalently, if $T_n(d,l) < T_{-(d,l)}(B)$. This relationship is expressed as:

$$\{S_{-(d,l)}(T_n(d,l)) < B\} = \{T_{-(d,l)}(B) > T_n(d,l)\}. \quad (3)$$

The hit probability for data object d and layer l is then:

$$\begin{aligned} h(d,l) &= \mathbb{P}\left(T_{-(d,l)}(B) > T_n(d,l)\right) & (4)\\ &= \mathbb{E}\left[1 - (1-p(d,l))^{T_{-(d,l)}(B)-1}\right]. & (5) \end{aligned}$$

Since $T_{-(d,l)}(B)$ corresponds to the time when the working-set size first reaches B, we have:

$$B = \sum_{\substack{i=1\\ i\neq d}}^{D} \sum_{k=1}^{V} \delta(i,k) \mathbb{E}\left[1-(1-p(i,k))^{T_{-(d,l)}(B)-1}\right] + \sum_{k=1}^{l-1} \delta(d,k) \mathbb{E}\left[1-(1-p(d,k))^{T_{-(d,l)}(B)-1}\right]. \quad (6)$$

We use two common approximations from the literature to simplify hit probability calculations; see [4], [9] for details.

Approximation 1: For $D \gg 1$, the characteristic time $T_{-(d,l)}(B)$ becomes concentrated around its mean value. Therefore, $T_{-(d,l)}(B)$ can be approximated by a deterministic value $t_{-(d,l)}(B)$ for data object d and layer l.

Approximation 2: The dependence of $t_{-(d,l)}(B)$ on (d,l) can be ignored for all data objects and layer. This is works when $p(d,l)$ is relatively insignificant to 1, and becomes exact if request probabilities are equiprobable. In summary, the working-set approximation for LLRU is as follows. Let $t^*(B)$ be such that:

$$B = \sum_{d=1}^{D} \sum_{l=1}^{V} \delta(d,l) \left(1-(1-p(d,l))^{(t^*(B)-1)}\right) \quad (7)$$

Then the hit probability for data object $d \in \mathcal{D}$ and layer $l \in \{1,2,\ldots,V\}$ is given by

$$h(d,l) = \left(1-(1-p(d,l))^{(t^*(B)-1)}\right). \quad (8)$$

This hit probability for data object d and layer l is equal to hit probability for data object d and version v, where $v = l$ because of the property of LLRU. The results for a time-slotted system can be extended to continuous time, where the request arrival process for data object d and layer l is a Poisson process with parameter $\gamma_{d,l}$. The hit probability for data object d and layer l is given by

$$h(d,l) = 1 - e^{-\gamma_{d,l} t^*(B)}, \quad (9)$$

where $t^*(B)$ is such that:

$$B = \sum_{d=1}^{D} \sum_{l=1}^{V} \delta(d,l) \left(1 - e^{-\gamma_{d,l} t^*(B)}\right). \quad (10)$$

In the next section, we show the asymptotic accuracy of working-set approximation.

F. Asymptotic accuracy of working-set approximation

We extend the analysis from [7] to incorporate layers into the construction, focusing on LR for this part. Specifically, we consider a system of caches where the request probability for data objects and the working-set size scale as a function of D. In this framework, each data object is assumed to have V fixed layers (or versions).

Let F be a smooth, monotone increasing function with domain [0, 1], such that $F(0) = 0$ and $F(1) = 1$. We define the request probability for data object d and version v as D scales in the following manner:

$$q^{(D)}(d,v) = \left(F(d/D) - F((d-1)/D)\right) g(v; d/D) \quad (11)$$

where $g(v; d/D)$ denotes the request probability for version v of data object d and $\sum_{v=1}^{V} g(v; d/D) = 1$ for all data objects. Based on the definition of F and g, we have $\sum_{d=1}^{D} \sum_{l=1}^{V} q^{(D)}(d,v) = 1$ and $q^{(D)}(d,v) \geq 0$. Thus, $q^{(D)}(d,v)$ is a probability distribution determined by F and g. We use $\delta^{(D)}(d,l)$ to denote the size of layer l for data object d and $p^{(D)}(d,l) = \sum_{v=l}^{V} q^{(D)}(d,v)$ denotes the request probability for layer l of data object d.

We define $b = B/D$, which scales as a function of D, and develop the notion of characteristic time in the same way as in the previous section. We assume a system with time-slots and request arrival process for data object d and layer l is a Bernoulli process with parameter $p^{(D)}(d,l)$ and such events occur independently over time-slots.

Suppose there is a request for data object d and layer l at time zero. Let $T_f^{(D)}(i,k)$ be the time of first request for data object $i \neq d$ and layer k, where $k = \{1, 2, \dots, V\}$. We use $T_n^{(D)}(d,m)$ to denote the time of next query for data object d and layer m, where $m \leq l$. Under the Bernoulli arrival process model, these times are geometrically distributed, i.e., $T_f^{(D)}(d,l) \sim \mathrm{Geo}(p^{(D)}(d,l))$ or $T_n^{(D)}(d,l) \sim \mathrm{Geo}(p^{(D)}(d,l))$. At time $t > 0$, the total size of different data objects and layer requested upto time t (i.e., working-set size), excluding requests for data object d and layer l is

$$S_{-(d,l)}^{(D)}(t) = \sum_{k=1}^{l-1} \delta^{(D)}(d,k) \mathbf{1}\left\{T_n^{(D)}(d,k) < t\right\} + \sum_{\substack{i=1 \\ i \neq d}}^{D} \sum_{k=1}^{V} \delta^{(D)}(i,k) \mathbf{1}\left\{T_f^{(D)}(i,k) < t\right\}, \quad (12)$$

with

$$\mathbb{E}\left[S_{-(d,l)}^{(D)}(t)\right] = \sum_{k=1}^{l-1} \delta^{(D)}(d,k) \left(1 - (1 - p^{(D)}(d,k))^{(t-1)}\right) + \sum_{\substack{i=1 \\ i \neq d}}^{D} \sum_{k=1}^{V} \delta^{(D)}(i,k) \left(1 - (1 - p^{(D)}(i,k))^{(t-1)}\right). \quad (13)$$

Similarly, we can find the working-set size at time t and its expectation is given by:

$$\mathbb{E}\left[S^{(D)}(t)\right] = \sum_{d=1}^{D} \sum_{l=1}^{V} \delta^{(D)}(d,l) \left(1 - (1 - p^{(D)}(d,l))^{(t-1)}\right). \quad (14)$$

Lemma 1. In the independent reference model with D data objects and V layers, the variance in the size of the working set is bounded by:

$$\mathrm{V}\left(S^{(D)}(t)\right) \leq \left(\frac{D \cdot V}{4} + D \cdot V \cdot (V-1)\right) \cdot \left(\delta_{\max}^{(D)}\right)^2,$$

where $\delta_{\max}^{(D)} = \max_{d \in \mathcal{D}, l \in \{1,2,\dots,V\}} \delta^{(D)}(d,l)$.

Proof. Let $X^{(D)}(d,l;t)$ be a random variable which is 1 if data object d and layer l is in the cache at time t, and 0 otherwise. The working set at time t is given by:

$$S^{(D)}(t) = \sum_{d=1}^{D} \sum_{l=1}^{V} \delta^{(D)}(d,l) X^{(D)}(d,l;t), \quad (15)$$

then the variance $\mathrm{V}\left(S^{(D)}(t)\right)$ in the size of working set is

$$\mathrm{V}\left(S^{(D)}(t)\right) = \mathrm{V}\left(\sum_{d=1}^{D} \sum_{l=1}^{V} \delta^{(D)}(d,l) X^{(D)}(d,l;t)\right). \quad (16)$$

Since

$$\mathrm{V}\left(X^{(D)}(d,l;t)\right) \leq 1/4,$$
$$\mathrm{Cov}\left(X^{(D)}(i,l;t), X^{(D)}(d,l;t)\right) \leq 0, i \neq d$$
$$\mathrm{Cov}\left(X^{(D)}(d,l;t), X^{(D)}(d,k;t)\right) \leq 1, l \neq k,$$

where Cov is the covariance. We find that

$$\mathrm{V}\left(S^{(D)}(t)\right) \leq \left(\frac{D \cdot V}{4} + D \cdot V \cdot (V-1)\right) \cdot \left(\delta_{\max}^{(D)}\right)^2 \quad (17)$$

□

We define Riemann integrable Δ satisfying $\Delta(d/D, l) = \delta^{(D)}(d,l)$ for all D, d and l for the theorem below.

Theorem 1 (Asymptotic hit probability)**.** *Consider the system of caches which scales as a function of D. For large D, the hit probability for data object d and layer l, $h^{(D)}(d,l)$, is approximated by*

$$h^{(D)}(d,l) = \left(1 - (1 - p^{(D)}(d,l))^{(t^*(B)-1)}\right) \quad (18)$$

where $t^*(B)$ is such that:

$$B = \sum_{d=1}^{D} \sum_{l=1}^{V} \delta^{(D)}(d,l) \left(1 - (1 - p^{(D)}(d,l))^{(t^*(B)-1)}\right). \quad (19)$$

In the limit $D \to \infty$, the hit probability for data object d and layer l is given by:

$$h(d,l) := \lim_{D \to \infty} h^{(D)}(d,l) = 1 - e^{-\tau^*(b) F'(d) \sum_{v=l}^{V} g(v;d)} \quad (20)$$

where $\tau^*(b)$ is such that:

$$b = \lim_{D \to \infty} \mathbb{E}\left[\frac{S^{(D)}(D\tau)}{D}\right] \quad (21)$$

and

$$\lim_{D \to \infty} \mathbb{E}\left[\frac{S^{(D)}(D\tau)}{D}\right] = \int_0^1 \sum_{l=1}^{V} \Delta(x,l) dx - \int_0^1 \sum_{l=1}^{V} \Delta(x,l) e^{-\tau^*(b) F'(x) \sum_{v=l}^{V} g(v;x)} dx. \quad (22)$$

Proof Sketch. This proof is inspired by [7]. Due to space limitations we refer the reader to Theorem 1 in [7] for the intermediate steps of the proof.

Using Mean Value Theorem, Lemma 10 from [7], and Convergence of Riemann Integrals, we obtain Eq. 22. Let $\tau^*(b)$ denote the unique solution to Eq. 21. For finite $D \gg 1$, Eq. 21 can be approximated by:

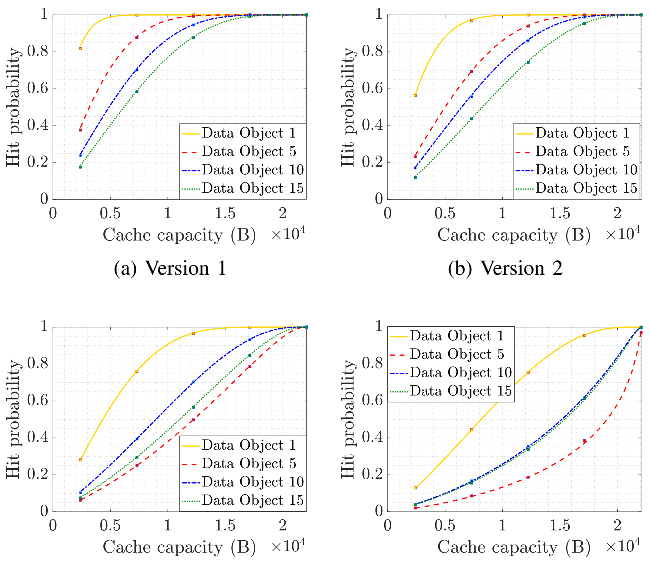
$$B = \sum_{d=1}^{D} \sum_{l=1}^{V} \delta^{(D)}(d,l) \left(1 - (1 - p^{(D)}(d,l))^{(D\tau-1)}\right) \quad (23)$$

for $B = Db$. We define $t^*(B) = D\tau^*(b)$ as the unique solution for the above equation. Note that as $D \to \infty$,

$$\frac{S_{-(d,l)}^{(D)}(D\tau)}{D} \sim \frac{S^{(D)}(D\tau)}{D}$$

Now using Lemma 1 and convergence in probability we have:

$$\lim_{D \to \infty} \mathbb{P}\left(S_{-(d,l)}^{(D)}(D\tau) \geq B\right) = \lim_{D \to \infty} \mathbb{P}\left(S^{(D)}(D\tau)/D \geq b\right) = u(\tau - \tau^*(b)) \quad (24)$$



(a) Version 1 (b) Version 2

(c) Version 3 (d) Version 4

Fig. 1. Hit probability against cache capacity for selected data objects under LLRU caching policy.

By Palm's theorem [1], the stationary LRU miss probability for data object d and layer l is

$$1 - h^{(D)}(d,l) = \mathbb{P}\left(S^{(D)}_{-(d,l)}(T_n^{(D)(d,l)}) \geq B\right)$$

$$= \sum_{t=1}^{\infty} \mathbb{P}\left(S^{(D)}_{-(d,l)}(t) \geq B | T_n^{(D)(d,l)} = t\right) \mathbb{P}\left(T_n^{(D)(d,l)} = t\right)$$

For $B = Db$ and $D \gg 1$, we use $t = D\tau$ and Eq. 24 to simplify above to get:

$$1 - h^{(D)}(d,l) = (1 - p^{(D)}(d,l))^{D\tau^*(b)-1}$$

for all data objects d and layer l. We can replace $D\tau^*(b)$ with $t^*(B)$. As $D \to \infty$, using Lemma 10 from [7] or point wise limits for right hand side, we obtain

$$1 - h(d,l) = e^{-\tau^* F'(d) \sum_{v=l}^{V} g(v;d)}. \qquad (25)$$

□

III. Numerical Evaluation and Simulation Results

In this section, we perform extensive numerical evaluations based on the working set approximation. The aim is to characterize the fundamental tradeoffs underlying the caching of data objects with layered representations.

A. How accurate is the working set approximation for LLRU?

Setting. We consider a caching system with $D = 100$ data objects, each having $V = 4$ layered versions. The request probability $q(d)$ follows a Zipf distribution with parameter 0.8, while the request probability $q(d,v)$ for version v is uniformly selected from $(0, q(d))$, ensuring $\sum_{v=1}^{V} q(d,v) = q(d)$. Additionally, we impose $q(d,v_1) > q(d,v_2)$ for $v_1 < v_2$, reflecting the higher request frequency of lower versions. Requests for

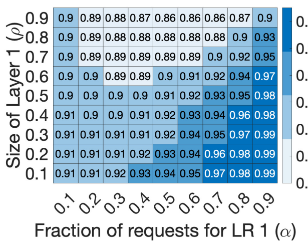
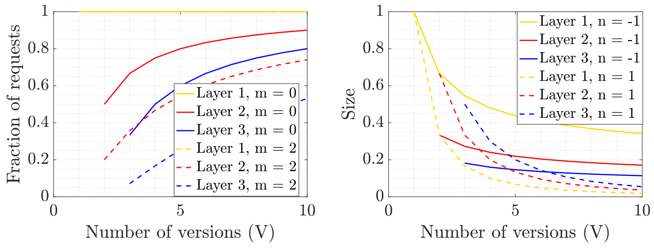


Fig. 2. Performance of LLRU for different size and popularity for a cache capacity of 20.



(a) Fraction of requests for layers. (b) Size of layers.

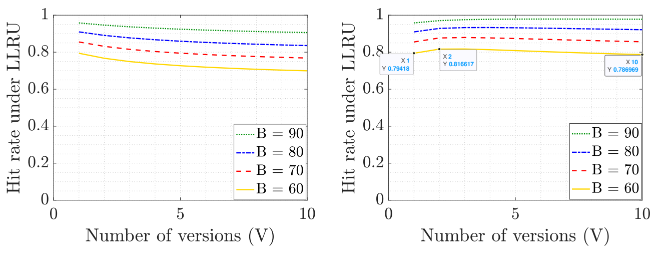
Fig. 3. Popularity and size characterization for different values of m and n as a function of number of versions.

object d and version v follow a Poisson process with rate $q(d,v)$. The size of each layer is uniformly chosen from $[1,240]$, ensuring a total object size of 240.

Results Discussion. Figure 1 shows the hit probability of different versions for objects ranked 1, 5, 10, and 15. Squares represent LLRU simulation results, obtained from sufficiently long runs for accuracy, while lines correspond to working set approximations. The near-perfect agreement across all object layers validates the approximation, which we use to explore further questions.

B. Impact of layers' sizes and popularity on performance

In this section, we study the impact of layer size and popularity of layers on the hit rate. More specifically, we will fix the size of layers and consider how to set the popularity of versions and thus layers that is optimal. Similarly, for a fixed popularity of versions, we address the optimal setting of the size of each layer.



(a) m = 0, n = -1 (b) m = 2, n = -1

Fig. 4. Comparison of LLRU with V vs. 1 version under LR where popularity and size scales as a function of V.

Setting. For this section, we have a caching system with $D = 100$ data objects and $V = 2$ versions. The request probability for data objects follows a Zipf distribution with parameter 0.8. For the case of 2 versions, we use α to denote the fraction of requests for Version 1 of data object d and $\rho = \delta(d, 1)$ to denote the size of Layer 1 of data object d.

Results discussion. *1) How to set the size and popularity when each data object has 2 version?*

We show the performance of the cache under the LLRU caching policy in Figure 2 for different values of the fraction of requests for Version 1 and the size of Layer 1. We observe that for a fixed fraction of requests for LR 1, as the size of Layer 1 decreases, the performance improvement is monotonically increasing. However, the same is not true for the fixed size of Layer 1 and the increasing fraction of requests for LR 1. Furthermore, if both the size and fraction of requests vary simultaneously, possibly along a diagonal, the hit rate does not follow a monotonic pattern. The last observation is that significant improvements occur with an increasing fraction of requests for LR 1 and a decreasing size of Layer 1.

C. Is it beneficial to increase the number of versions for a data object?

Setting. We consider 100 data objects with request probabilities following a Zipf distribution (parameter 0.8). Given V versions, the request probability of the v-th version of object d is: $q^{(V)}(d, v) = \frac{(V-v+1)^m}{\sum_{i=1}^{V}(V-i+1)^m}$. The size of version v is: $s_{\mathrm{LR}}^{(V)}(d, v) = \sum_{l=1}^{v} \delta^{(V)}(d, l)$, where $\delta^{(V)}(d, l) = \frac{l^n}{\sum_{i=1}^{V} i^n}$. We plot the request probability, $p^{(V)}(d, l) = \sum_{v=l}^{V} q^{(V)}(d, v)$, for the first three layers, along with $\delta^{(V)}(d, l)$ as a function of V in Figure 3a and Figure 3b. This is analogous to the case of partitioning an object into subobjects based on varying popularities, such as the difference in request probabilities for street views of a building versus its interiors.

Results discussion. We compare LLRU performance with V vs. 1 version in Figure 4. Figures 4a and 4b depict the hit rate under LLRU as a function of V for different values of m and n. Our results show that both the popularity and size of layers must vary at a certain rate to improve the hit rate. In Figure 4a, the hit rate decreases monotonically with V, whereas increasing m for the same n introduces non-monotonic behavior, as seen in Figure 4b. This highlights the nuanced impact of V, popularity, and size characterization on hit rate. Additionally, we observe a monotonically increasing hit rate for all $m > 0$ and $n \geq 0$.

IV. Conclusion

The efficient management of the large amounts of data required by emerging delay-constrained applications, e.g., multiplayer VR gaming and NN-based inference, will require judicious use of caching that exploits, when appropriate, hierarchies of data object representations that enable tradeoffs between a data object's size and quality. To address this, in this paper, we have studied Layered Least Recently Used (LLRU) optimized for data objects with layered representations. To that end, we have developed a working set approximation for a practical policy. We exploit this approximation to investigate the impact that the incremental size of layers and the level of demand for different versions will play. The paper studies these impacts showing, e.g., when additional layers may be of value, and when they may indeed be counterproductive, towards enhancing performance.

Acknowledgment

This work was supported by National Science Foundation CNS- 2212202 Award.

References

[1] F. Baccelli and P. Brémaud. *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences, 2nd Ed.* Springer-Verlag, Berlin, 2003.

[2] Agrim Bari, Gustavo de Veciana, and George Kesidis. Fundamentals of caching layered data objects, 2025. [Online] Available:https://arxiv.org/abs/2504.01104.

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

[4] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: modeling, design and experimental results. *IEEE JSAC*, 20(7):1305–1314, 2002.

[5] A. Dan and D. Towsley. An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes. In *Proc. ACM SIGMETRICS*, 1990.

[6] Peter J. Denning and Stuart C. Schwartz. Properties of the working-set model. *Commun. ACM*, 15(3):191–198, mar 1972.

[7] R. Fagin. Asymptotic miss ratios over independent references. *Journal Computer and System Sciences*, 14(2):222–250, 1977.

[8] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[9] C. Fricker, P. Robert, and J. Roberts. A Versatile and Accurate Approximation for LRU Cache Performance. In *Proc. International Teletraffic Congress*, Krakow, Poland, 2012.

[10] Felix Hartanto, Jussi Kangasharju, Martin Reisslein, and Keith Ross. Caching video objects: Layers vs versions? *Multimedia Tools Appl.*, 31(2):221–245, nov 2006.

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

[12] H. Hoppe. Progressive meshes. In *Proc. ACM SIGGRAPH Conf. on Computer Graphics and Interactive Techniques*, page 99–108, 1996.

[13] Predrag R. Jelenković. Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. *The Annals of Applied Probability*, 9(2):430–464, 1999.

[14] B. Jiang and Y. Mu. Russian doll network: Learning nested networks for sample-adaptive dynamic inference. In *Proc. IEEE/CVF ICCV Workshops*, pages 336–344, 2021.

[15] Eunwoo Kim, Chanho Ahn, and Songhwai Oh. Nestednet: Learning nested sparse structures in deep neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8669–8678, 2018.

[16] W. King. Analysis of paging algorithms. *Proc. IFIP Congress*, pages 485–490, 1971.

[17] Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. *J. ACM*, 68(4), jul 2021.

[18] Memcached. https://memcached.org.

[19] A. Ortega, F. Carignano, S. Ayer, and M. Vetterli. Soft caching: web cache management techniques for images. In *Proc. First Signal Processing Society Workshop on Multimedia Signal Processing*, pages 475–480, 1997.

[20] T. Shanableh and M. Ghanbari. Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats. *IEEE Transactions on Multimedia*, 2(2):101–110, 2000.

[21] Y. Zhang, J. Yang, Y. Yue, Y. Vigfusson, and K.V. Rashmi. SIEVE is Simpler than LRU: an Efficient Turn-Key Eviction Algorithm for Web Caches. In *Proc. USENIX NSDI*, 2024.