The LCD interconnection of LRU caches and its analysis[*]

Nikolaos Laoutaris, Hao Che, Ioannis Stavrakakis

Abstract

In a multi-level cache, a hit at level l leads to the caching of the requested object in all intermediate caches on the reverse path (levels $l-1,\ldots,1$). This paper shows that a simple modification to this de facto behavior, in which only the $l-1$ level cache gets to store a copy, can lead to significant performance gains. The modified caching behavior is called *Leave Copy Down* (LCD); it has the merit of being able to avoid the amplification of replacement errors and also the unnecessary repetitious caching of the same objects at multiple levels. Simulation results against other cache interconnections show that LCD reduces the average hit distance. We construct an accurate approximate analytic model for the case of LCD interconnection of LRU caches. The developed model presents several novel techniques for the analysis of interconnected caches, and gives a clear insight as to why the LCD interconnection yields an improved performance. Although initially motivated from web caching, LCD and its analysis are more general in scope, and apply to other applications of caching as well.

1 Introduction

A cache is a fast access memory that mediates between a consumer of information, and a slower memory where information is stored on a permanent basis. The function of the cache is to maintain a set of most valuable objects, so that they may be accessed promptly, thus

[*]Nikolaos Laoutaris and Ioannis Stavrakakis are with the Dept. of Informatics and Telecommunications, University of Athens, 15784 Athens, Greece. E-mail: {laoutaris,ioannis}@di.uoa.gr. Hao Che is with the Dept. of Computer Science and Engineering, Univerity of Texas at Arlington, USA. E-mail: hche@cse.uta.edu.

avoid accessing the slower and/or remote (permanent) memory. Caching is one of the most pervasive and omnipresent ideas of computer science. It has been studied and applied in many different domains, such as: in computer architecture, to speed up the communication between the central processor unit (CPU) and the main memory (RAM); in operating systems, to perform paging, i.e., keep in the RAM the most valuable blocks from the permanent storage devices (hard disks); in distributed file systems, to keep frequently accessed files closer to the clients; in the world wide web, to allow clients to receive web content from local proxy servers, thus avoid accessing the remote origin servers through the network.

In several occasions caches are employed at multiple levels. Examples are, multi-level cache architectures in modern CPUs, multi-level caches in RAID disk arrays, multi-level caches in the world wide web. Under the modus operandi of such multi-level systems, requests are first received at the lowest level cache (the one closest to the client), and then routed upwards until they reach a cache that stores the requested object. A hit is said to have occurred in that case. Following the hit, the requested object is sent downwards on the reverse path to the client, and each cache on this path gets to store a local copy of the object.

Leaving copies everywhere on the reverse path (here after abbreviated LCE), has been considered as a de facto behavior. Despite the vast bibliography on caching, we are only aware of a few works that have questioned this de facto behavior [1, 2, 3]. This paper continues on this line of research, investigating whether caching a local copy in *all* intermediate caches on the reverse path is indeed a good idea, or are there reasons to revise it, and instead keep copies only in a *subset* of intermediate caches. Our answer to this question is that LCE is not always the best choice and that simple alternative algorithms can outperform it in a variety of common scenarios. In [3], we have proposed such an algorithm – which we call *Leave Copy Down* (LCD) – that appears to be superior to LCE and other potential algorithms, across a wide range of parameters. The operation of LCD is quite simple; instead of storing a copy in all intermediate caches, only the immediate downstream neighbor of the hit cache gets to store one. This way, objects move gradually from the origin server towards the clients, with each request advancing them by one hop. The conception of LCD and the initial discussion in [3] were motivated in the context of web caching; the algorithm, however, is not specific to web caching and may be applied to other applications of caching as well.

The current work focuses on LCD, and takes an analytic look at its workings, with the aim of deepening our understanding as to why this particular algorithm yields an improved performance. By developing an appropriate analytic performance evaluation model, it becomes clear why this is the case. The enhanced performance of LCD stems from: (1) its ability to avoid the *amplification of replacement errors* (it limits replacement errors locally to a single cache instead of allowing them to spread to an entire chain of caches); (2) its ability to provide for *exclusive caching* (allows each cache on a chain of caches to hold a potentially different set of objects, thus avoiding the repetitious replication of the same few objects).

The presented analysis of the LCD algorithm is a contribution to the analytic study of interconnected caches. Several approximation techniques are introduced in order to overcome problems such as the combinatorial hardness of analyzing LRU replacement, the correlation in the miss streams that flow from cache to cache, the coupling of cache states under LCD (the state of a cache depending on the state of its downstream neighbor and vice versa). By carefully combining the various approximation techniques, the resulting final analytic model is able to predict satisfactorily the performance of the real system. Apart from LCD, the presented analysis methodology can be applied to other interconnections as well, thus having a potentially broader scope.

The remainder of the paper is structured as follows. Section 2 introduces several cache interconnection algorithms (called *meta* algorithms), elaborates on the desired properties for such algorithms, and presents a performance comparison via simulation. Section 3 gives an overview of previous approaches for the analysis of LRU caching. Section 4 presents the basic theory for the analysis of isolated LRU caches; this theory is employed as a building block in later parts. Section 5 presents the analysis of LCD-interconnected tandems of LRU caches, and the required modifications to handle more general tree topologies. Section 6 demonstrates the accuracy of the final analytic model. Section 7 concludes the paper.

2 Meta algorithm for multi-level caches

The question of whether to cache an object at an intermediate cache is one that may be possed independently of the specific replacement algorithm operating on the cache. For this reason, the algorithms that are studied here may be characterized as *meta algorithms* for multi-level caches (or just meta algorithms) to differentiate them from the much discussed and well understood replacement algorithms, and to stress the fact that they operate independently of the latter. Meta algorithms have been employed in the past in the different, but related to caching, domain of self-organizing linear search (see Hester and Hirschberg [4] and references therein). Apart from the apparently different application domain, the meta algorithms employed in this work operate on groups of caches organized in tandems or trees, whereas earlier work on self-organizing linear search studied the operation of meta algorithms on isolated linear lists.

In the following, we consider multi-level caches operating under several different meta algorithms. In all cases it is assumed that the Least Recently Used (LRU) replacement algorithm runs in all caches. Due to the aforementioned independent operation of the meta algorithm from the employed replacement algorithm, it is believed that the presented results and conclusions should apply to some extent to other replacement algorithms as well (the reader is referred to Podlipnig and Böszörmenyi [5] for an up-to-date survey of replacement algorithms).

2.1 Description

This section describes three new meta algorithms, LCD, MCD, Prob, as well as the currently employed one, LCE, and two recently proposed ones, Filter (Che et al. [1]) and DEMOTE (Wong and Wilkes [2]).

2.1.1 Leave Copy Everywhere (LCE)

This is the standard mode of operation currently in use in most multi-level caches. When a hit occurs at a level l cache or the origin server, a copy of the requested object is cached in all intermediate caches (levels $l-1, \ldots, 1$) on the path from the location of the hit down to the requesting client.

2.1.2 Leave Copy Down (LCD)

Under LCD a new copy of the requested object is cached only at the $(l-1)$-level cache, i.e., the one that resides immediately below the location of the hit on the path to the requesting client. LCD is more "conservative" than LCE as it requires multiple requests to bring an object to a leaf cache, with each request advancing a new copy of the object one hop closer to the client.

2.1.3 Move Copy Down (MCD)

Similar to LCD with the difference that a hit at level l *moves* the requested object to the underlying cache. This requires that the requested object be deleted[1] from the cache where the hit occurred. No deletion of course takes place when the hit occurs at the origin server. The idea behind MCD is to reduce the number of replicas for the same object on the path between the requesting client and the origin server.

2.1.4 Prob(p)

Prob is a randomized version of LCE. Each intermediate cache on the path from the location of the hit down to the requesting client is eligible for storing a copy of the requested object. An intermediate cache keeps a local copy with probability p, thus invoking the replacement algorithm, and does not keep a copy with probability $1-p$. Prob(1) is identical to LCE.[2]

The operation of the above mentioned algorithms is illustrated in Fig. 1. Note that the three new meta algorithms require a very small amount of extra co-operation other than the minimum required to implement a multi-level cache, i.e., each cache to know its immediate ancestor so that it can forward requests upstream, and its immediate descendant(s) so that it

[1]The object does not have to be physically deleted from the cache. A better strategy is to set its timestamp to a very small value thus marking it for eviction upon the next miss of any other requested object. This has the advantage that in the case that the next request refers to this object, a hit will occur, whereas a miss would have occurred if physical deletion had taken place.

[2]Probabilistic algorithms have been recently employed in the domain of web caching, but in different contexts. Psounis and Prabhakar [6] show that replacement algorithms may be efficiently implemented by utilizing only samples from the cache (instead of the entire cache content) in order to identify a good eviction candidate. In Starobinski and Tse [7], randomized algorithms are used in order to handle objects with varying costs (fetch distance) and sizes.

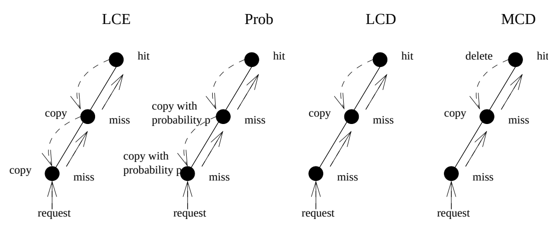

Figure 1: Operation of LCE, Prob, LCD, and MCD.

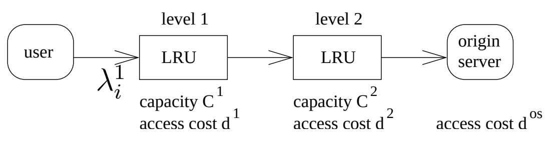

Figure 2: A two level LRU tandem.

can forward objects downstream.

2.1.5 Filter: a non-memoryless meta algorithm

A major distinctive factor of replacement algorithms is whether they are memoryless or non-memoryless. The LRU algorithm is characterized as memoryless because each permutation of the cache, following a request, utilizes no additional information from the memory; the permutation is executed by simply bringing the requested object to the top of the LRU list. Frequency based algorithms, however, need to maintain additional information in the memory to keep track of the number of requests for each object. Similar is the case with function-based replacement algorithms that use multi-criteria value functions (capturing multiple criteria like size, fetch distance, recency/frequency etc.) in order to make replacement decisions (such an example is GD-Size, see [5] for more). Such algorithms are characterized as non-memoryless.

The aforementioned characterization may be applied to the field of meta algorithms as well. All the aforementioned meta algorithms may be characterized as memoryless because they do not require any extra information other than the actual state of the hierarchy. Recently, Che et al. [1] proposed a new meta algorithm for hierarchical caches, aiming at improving the overall hit ratio. This algorithm, that will here after be referred to as Filter, appears to be non-memoryless.

Under the Filter algorithm, a hit for object i at level l leads to the caching of i in an intermediate cache m, when m satisfies the following condition: $\tau_m^{-1} < \lambda_i$: τ_m is said to be the *characteristic time* of cache m; λ_i is the frequency that object i is requested at the client level. Each cache m is seen as a low pass filter with a cutoff frequency equal to τ_m^{-1} and, thus, a multi-level cache behaves like a tandem of low pass filters having different cutoff frequencies. Additionally, when an object is evicted from a cache at level l, Filter forces its caching at level $l + 1$ if not already cached there. Filter incurs additional complexity as compared to the previous memoryless meta algorithms, as it needs to estimate the request frequency of each requested object, and disseminate this information to all the caches along with the request. This extra processing and the information that must be maintained make Filter a non-memoryless meta algorithm.

2.1.6 DEMOTE

Wong and Wilkes [2] have proposed a simple inter-cache co-operation mechanism that they call DEMOTE. A cache instead of just evicting an object, it demotes it, i.e., sends it for caching at the above level, where it is inserted to the head of the LRU list (a similar mechanism is included in the Filter algorithm of Che et al.). Additionally, when a cache transmits an object to a downstream cache, it moves its local copy to the tail of its LRU list (to be evicted with the next replacement); this is similar to the MCD algorithm (Sect. 2.1.3). The goal of DEMOTE is to avoid the duplication of the same objects at multiple levels (more discussion on this important issue in Sect. 2.2.2).

2.2 Design Principles

The three new meta algorithms, LCD, MCD, and Prob, aim at improving the performance of a multi-level cache in terms of the expected distance to reach a cache hit. To achieve this goal they take advantage of the following design principles.

2.2.1 Avoid the amplification of replacement errors

On-line replacement algorithms are bound to commit replacement errors as compared to the OPT replacement strategy of Belady that replaces the object with the maximum forward distance until the next request. OPT is of course not realizable in practice, as it requires knowledge of future requests. Similarly, when considering independent identically distributed requests – the so called independent reference model (IRM) – replacement errors occur when a less popular object causes the eviction of a more popular one.

Thus, any causal replacement algorithm is committing errors as compared to the optimal, and these errors lead to inferior performance. The effect of replacement errors becomes even more critical when considering multi-level, rather than isolated caches. In an L-level multi-level cache that operates under the LCE meta algorithm, a request for an unpopular object may lead to its caching in all L caches on the path from the requesting client up to the root cache, and by doing so commit up to L replacement errors. Leaving a copy in all the intermediate caches is, in effect, leading to the *amplification of replacement errors*. The proposed algorithms try to reduce the extent of this amplification by reducing the number of copies that are cached with each request.

In the particular case of web caching, replacement errors are brought to their extreme, due to the existence of a very high percentage of objects that are requested only once. These so called $one - timer$ objects usually amount up to 45% of the total requests and 75% of the total distinct objects present in the measured workloads [8, 9]. Caching an one-timer object is the worst type of replacement error that can occur as it is guaranteed that the one-timer will not be requested again thus leading to waste of storage capacity. The aforementioned high percentages of one-timers clog an entire multi-level cache that operates under LCE with useless objects. The proposed LCD and MCD meta algorithms guarantee that the one-timers cannot affect any cache other than the root cache. Thus they completely filter-out one-timers for all but one cache in the hierarchy. Prob, likewise, filters out most of the one-timers by using a small cache probability p ($p = 0.2$ is used in Sect.2.3, see [3] for more results).

2.2.2 Achieve cache exclusivity

Cache hierarchies that operate under LCE end up duplicating the same objects at multiple levels. Their performance deteriorates to approximately the performance of the largest cache in the hierarchy, whereas it should ideally be approaching the performance of a cache that is as large as the sum of all the caches in the hierarchy. The problem of making cache hierarchies *exclusive* – i.e., forcing them to store disjoint sets of objects at different levels – was studied recently by Wong and Wilkes [2] who proposed the DEMOTE algorithm. Our LCD meta algorithm shares some resemblance with [2], as it too caters to exclusivity, and it also strives for minimum added complexity (as opposed to other works that keep track of individual cache contents, and make completely coordinated caching/replacement decisions [10, 11], at the expense of a significant amount of added complexity in terms of state and communication). However, our approach towards exclusivity is completely different.

LCD takes an *active* approach towards exclusivity, whereas DEMOTE is potentially *passive* in its workings. To understand these characterizations, see that LCD attempts to prevent valueless objects from reaching the (valuable) leaf cache, whereas the DEMOTE architecture permits them to get at the leaf cache with a single request (just like LCE), and then attempts to achieve exclusivity by not evicting them at all levels, but rather giving them additional chances through the demote operations. All together, LCD attempts to achieve exclusivity through admission control prior to caching (hence being active), while DEMOTE attempts to achieve exclusivity through replacement (hence being passive). The passive operation has its cost, as it allows valueless object to linger at the lower levels of the hierarchy, depriving valuable cache space from the most valuable objects. Also, the demote operation consumes bandwidth, not only to transmit object downwards, but also to transmit objects upwards for the demote operations.

A second important difference relates to the performance under multiple clients. In the architecture of Wong and Wilkes, an intermediate cache that has transmitted an object to a downstream client, marks it for eviction upon the next replacement (much like our MCD algorithm). This is completely justified when considering a hierarchy for a single client, e.g., a memory hierarchy for a single CPU, but can be counter intuitive in a hierarchy with multiple clients, e.g., a tree-shaped interconnection of web caches, servicing multiple client institutions at the leaf levels. In the second case, removing an object from an intermediate cache after its transmission to a single client, prohibits other clients that have not yet cached it, from receiving it promptly from the intermediate cache, rather than from the origin server. Such a behavior wipes out a potentially important gain from servicing the so called *cold misses*, and often leads to performance that is even worse than that of the standard LCE algorithm. Wong and Wilkes comment on this matter that *"the clear benefits from single-client workloads are not so easily repeated in the multi-client case"*. For the latter case, they propose variations of the DEMOTE policy, that, however, make it more complex, and more important, require considerable amount of fine-tuning to specific operating conditions.

Handling multiple clients is inherent to the operation of LCD and no similar problems arise. An object that resides in an intermediate cache may flow towards multiple downstream caches without leaving its position. It is evicted from the intermediate cache, only if requests stop reaching it there, not as a consequence of being sent to a downstream cache (as is the case with DEMOTE). Thus the demand driven operation of LCD (copy to multiple clients when requested) coupled with a simple replacement logic (evict only when request rates fall and avoid interfering with the state of the LRU list (as done by DEMOTE)), seem to be more natural choices for handling multiple clients. Indeed, in all synthetic and trace-driven experiments that appear in [3], which are under multiple clients arranged in a tree topology, LCD always performs best.

2.3 Performance of different meta algorithms

This section presents an initial performance evaluation of the various meta algorithms. It does not strive for an exhaustive comparison under all possible workloads, but rather to enhance the experimental results that were presented in [3], and justify our interest in analyzing the LCD meta algorithm in this article.

The performance evaluation is conducted through synthetic simulation under Zipf-like requests and a two-level tandem topology (Fig. 2). A Zipf-like distribution is a power-law, dictating that the ith most popular object is requested with a probability K/i^a, where $K = (\sum_{j=1}^{N} \frac{1}{j^a})^{-1}$; N denotes the number of distinct objects. The skewness parameter a captures the degree of concentration of requests; values approaching 1 mean that few distinct objects receive the vast majority of requests, while small values indicate progressively uniform popularity.

The Zipf-like distribution is representative of workloads that lead to high hit ratios. It is under such workloads that caching becomes more effective. The Zipf-like model is recognized as a good model for characterizing the popularity of various types of measured workloads, such as web objects [12] and multimedia clips [13]. The popularity of P2P [14] and CDN content has also been shown to be quite skewed towards the most popular documents, thus approaching a Zipf-like behavior. Recently, evidence of Zipf-like behavior has been observed in the distribution of Gnutella queries [15].

The average hit distance is used as the performance metric. A simple “hop-count” notion

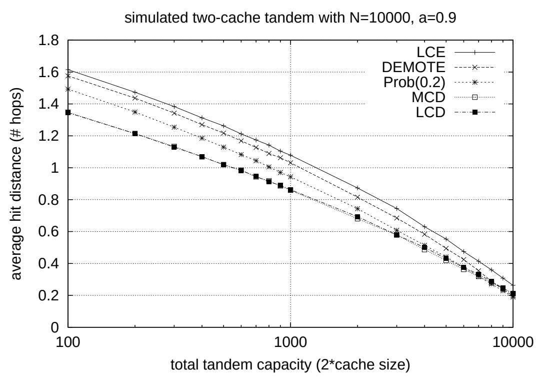

Figure 3: Average fetch distance (hit) in a two-level cache tandem under various meta algorithms

of distance is employed, thus it is assumed that the client is co-located with the level 1 cache ($d^1 = 0$), while the level 2 cache and the origin server are one and two hops away, respectively ($d^2 = 1$, $d^{\mathrm{os}} = 2$).

Figure 3 depicts the average hit distance under the meta algorithms of Sect. 2.1. The Filter algorithm is not included in the comparison, as it is the only one that requires laborious per-object frequency estimation (in [3] it has been shown experimentally that LCD outperforms Filter in the studied scenarios). The x-axis depicts the total capacity of the tandem in unit-sized objects (equally sized caches assumed). LCD and MCD yield almost the same performance that is at least 20% better than LCE and at least 15% better than DEMOTE in the initial range that reaches up to total tandem capacity of 1000 objects (representing 10% of all available objects). Most caching systems operate with relative capacity below 10%. In that range, LCD and MCD yield significant performance improvements.

As the available storage capacity increases, LCD, MCD, Prob, and DEMOTE progressively converge to almost the same performance, that is constantly better than that of LCE. Focusing on LCD and DEMOTE, we observe that whereas the two perform nearly as good under high availability of storage, LCD becomes clearly better under a low availability of storage. This can be explained intuitively as follows. Under abundant storage, the most popular objects will be cached at level 1 with a high probability despite the replacement errors, thus cache exclusivity becomes the dominant factor for improving the performance; both LCD and DEMOTE cater to exclusivity. Under limited storage, however, valuable objects often get replaced in favor of valueless ones. In that case, it is the avoidance of such replacement errors that dominates the performance, and exclusivity comes second. LCD avoids replacement errors at the valuable level 1 cache by requiring multiple requests to cache an object, whereas DEMOTE permits such errors to occur by requiring just a single request to cache a valueless object at level 1.

Another interesting observation is that whereas LCD and MCD perform almost the same here, LCD becomes clearly better under tree topologies [3]. This is attributed to the fact that MCD suffers from not allowing multiple downstream caches to share an object from an intermediate level cache (this issue has been discussed in Sect. 2.2.2 in the context of cache exclusivity). The fact that LCD is as good as MCD under tandem topologies, and better under tree topologies, has been one of the primary reason for our interest in analyzing LCD.

In the following sections, the LCD interconnection of LRU caches is mapped to an efficient (approximate) analytic model. The results from this analytic model give further insights to the performance gains of LCD, and to a large extent explain the observed experimental results. For further simulation results, including more synthetical workloads as well as trace-driven workloads and different topologies (trees), the reader is referred to [3].

3 Analytic models of LRU in the literature

To put the presented LCD/LRU analytic model for interconnected caches into perspective, we first review previous attempts to model LRU caching. We focus on analyses assuming independent identically distributed requests (the aforementioned IRM), that attempt to derive the expected behavior of LRU in steady-state; the reader is referred to Motwani and Raghavan [16] for analyses from the perspective of theoretical computer science, aiming at establishing worst case performance bounds for the (on-line) LRU with respect to the (off-line) optimal replacement policy.

It seems that King [17] was the first to derive the steady-state behavior of LRU under IRM. Initial attempts employed a Markov chain to model the contents of a cache operating under LRU. Unfortunately, such attempts give rise to huge Markov chains, having $C!\binom{N}{C}$ states (N number of objects in total, C capacity of the cache); numerical results for such chains can only be derived for very small N and C. More efficient steady-state formulas have been derived by avoiding the use of Markov chains, and instead making combinatorial arguments; see Koffman and Denning [18], and Starobinski and Tse [7] for such approaches that, however, still require exponential complexity to be evaluated numerically. Flajolet et al. [19] have presented integral expressions of LRU's hit ratio, that may be approximated through numerical integration at complexity $O(NC)$. Dan and Towsley [20] have derived an efficient $O(NC)$ iterative method for the approximation of LRU's hit ratio. Jalenković [21] has provided a closed form expression of hit ratio for the particular case of Zipf-like requests with skewness parameter $\alpha > 1$, for the asymptotic case, $N \to \infty$. The same author has shown that the hit ratio of LRU under Zipf-like requests is asymptotically insensitive for large caches, i.e., $C \to \infty$, to the statistical correlations of the request arrival process [22]. Thus the hit ratio under IRM is, in that case, similar to that under correlated requests.

Compared to the aforementioned works, Che at al. [1], and the current work, study the more difficult problem of interconnected LRUs. The added difficulty relates to the need of characterizing the miss process that flows to the upstream cache, in addition to the steady-state hit probabilities, which has been the only focus of prior works, e.g., [19], [20]; extending these works to handle multiple caches is not straight forward. The initial work of Che at al. [1] studies hierarchies that operate under LCE, whereas the current work studies hierarchies that operate under LCD. In the latter case, the analysis becomes even harder, due to the *coupling* of cache states under LCD (the state of the underlying cache depends on the state of the above one and vise versa), whereas under LCE, the hierarchy can be analyzed by a one-pass bottom up algorithm as there are no bidirectinal state dependencies.

4 The Che et al. approximation for individual LRU caches

This section presents the approximate analytic model for individual LRU caches that has been proposed recently by one of the authors (H. Che) in [1]. The model and its concepts will be used as building blocks at various occasions during the analysis of interconnected LCD/LRU caches that follows in subsequent sections. To this end, the presentation is adapted to the requirements of the current work. At certain points, it even adds details which do not appear

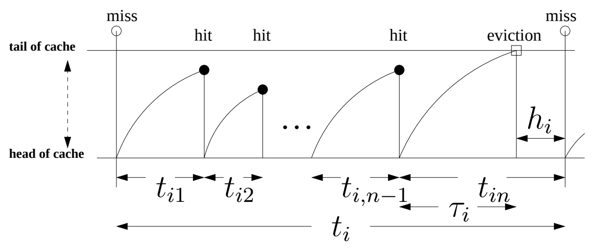

Figure 4: Time diagram of the variables involved in the Che et al. approximation.

in the original paper, with the aim to assist the reader. The first part of the section contains an elaborate version of the original analysis, while the second part contains a new, simpler way of deriving some of the results.

4.1 Analysis

Consider an LRU cache with capacity for C unit-sized objects, and a set of N distinct unit-sized objects. Assume that the request arrival process for object i, $1 \leq i \leq N$, follows a Poisson process with mean rate λ_i. The corresponding inter-arrival time, r, between successive requests for object i is thus exponential, having distribution $P_i(t) = P\{r \leq t\} = 1 - e^{-\lambda_i t}$ and density $\phi_i(t) = \lambda_i e^{-\lambda_i t}$. Under these assumptions, the following analytic model allows for the derivation of $\tilde{\pi}_i$, $1 \leq i \leq N$ – the probability of finding object i in the cache at an arbitrary observation time in steady-state.

Let t_i denote a random variable capturing the time between two subsequent misses for object i. This random variable can be written as the sum of n other random variables t_{ij}, $1 \leq j \leq n$, which are described as follows: t_{ij}, $1 \leq j \leq n-1$, is the inter-arrival time between two successive hits for i; t_{in}, is the inter-arrival time between the last hit for i and the subsequent miss. Figure 4 is a time diagram illustrating the aforementioned variables (see also Table 1 for a summary of notation). The exact distribution of t_{ij} for object i can be expressed as a combinatorial function involving the arrival processes of the other objects. The essence of the Che et al. approximation is to avoid the use of combinatorial formulas for the expression of t_{ij}, and instead write it in a much less complicated way by employing a *mean value approximation.*

Let τ_i denote the maximum inter-arrival time between two adjacent request for i that lead to hits; it will be referred to as the *characteristic time* of object i. In essence, τ_i is a random variable, but in the context of the Che et al. approximation it is considered a constant. The rationale behind this approximation is that as the request rate for the other objects increases, τ_i tends to fluctuate less and less around its mean value. This assumption is supported by numerical studies and the results appearing in [1]. In the same work, it is argued that the characteristic time for a given object may be obtained by solving the following equation for its unique solution τ_i:

$$\sum_{j=1, j\neq i}^{N} P_j(\tau_i) = C \tag{1}$$

The idea behind eq. (1) is to count the number of distinct/unique requests for other objects (non-tagged), that occur after the tagged object is brought to the head of the LRU list at $t = 0$, given that the tagged object is not requested. As each such request causes an insertion to the head of LRU, thereby pushing the tagged object one place back towards the tail of LRU, it is eventually evicted when C such requests have occurred; τ_i is the expected duration for this to happen. Since request inter-arrivals are exponential, thus memoryless, we can disregard the elapsed time for non-tagged request inter-arrivals prior to $t = 0$. Thus eq. (1) gives an exact expression for the characteristic time.

The following proposition establishes that the presented analysis may be applied as long as $N > C$, i.e., when the cache cannot hold all the objects, which is the usual case.

Proposition 1 *Equation (1) has exactly one real solution in $\mathbb{R}^+$ as long as $N > C$.*

Proof: Denote $f(\tau_i) = \sum_{j=1,j\neq i}^{N} P_j(\tau_i) - C$. The following hold true: (1) $f(\tau_i)$ is continuous and non-decreasing in $\mathbb{R}^+$ as it is the sum of $N-1$ functions $P_j(\tau_i)$, which are continuous and non-decreasing themselves (because they are exponential distributions); (2) $f(0) = -C < 0$, due to $P_j(0) = 1 - e^{-\lambda_j 0} = 0$, $1 \leq j \leq N, j \neq i$; (3) $\lim\limits_{\tau_i \to \infty} f(\tau_i) = N - C > 0$, due to $\lim\limits_{\tau_i \to \infty} P_j(\tau_i) = 1$, $1 \leq j \leq N, j \neq i$ (P_j's are distributions thus approach 1 towards ∞). From the three arguments and the intermediate value theorem, it follows that $f(\tau_i)$ has exactly one real solution in $\mathbb{R}^+$. $\square$

Under the aforementioned mean value approximation, t_{ij}'s become independent random variables whose distributions can be written in a straightforward manner by conditioning with respect to τ_i as follows: inter-arrivals that are shorter than τ_i involve two hits (as in the case of $t_{i1}, \ldots, t_{i,n-1}$); inter-arrivals that are longer than τ_i involve a hit and a subsequent miss (as in the case of t_{in}). The exact distributions are given below.

- Distribution of t_{ij}, $1 \leq j \leq n-1$ (having duration smaller than τ_i): All follow a common distribution denoted $P_{i-}(t)$, and defined as follows:

$$P_{i-}(t) = P\{t_{ij} \leq t\} = P\{r \leq t | r \leq \tau_i\} = \frac{P\{r \leq t, r \leq \tau_i\}}{P\{r \leq \tau_i\}} = \frac{P_i(t)u(\tau_i - t) + P_i(\tau_i)(1 - u(\tau_i - t))}{P_i(\tau_i)} \tag{2}$$

  where r is exponential following $P_i(t)$. The function $u(t)$ is the unit-step function, which is defined as follows: $u(t) = 1$ for $t \geq 0$ and 0 otherwise. Figure 5 gives a schematic explanation of the derivation of $P\{r \leq t, r \leq \tau_i\}$ which is involved in the derivation of $P_{i-}(t)$. The "$-$" sign in the subscript of $P_{i-}(t)$ is a mnemonic for indicating variables that are smaller (hence the minus sign) than a constant (here τ_i).

  The corresponding density function is:

$$\begin{aligned} f_{i-}(t) &= \frac{dP_{i-}(t)}{dt} = \frac{(P_i(t))' \cdot u(\tau_i - t) + P_i(t) \cdot (u(\tau_i - t))' + P_i(\tau_i) \cdot (u(t - \tau_i))'}{P_i(\tau_i)} \\ &= \frac{\phi_i(t) \cdot u(\tau_i - t) - P_i(t) \cdot \delta(\tau_i - t) + P_i(\tau_i) \cdot \delta(t - \tau_i)}{P_i(\tau_i)} \\ &= \frac{\lambda_i e^{-\lambda_i t} \cdot u(\tau_i - t) - (1 - e^{-\lambda_i t}) \cdot \delta(\tau_i - t) + (1 - e^{-\lambda_i \tau_i}) \cdot \delta(t - \tau_i)}{1 - e^{-\lambda_i \tau_i}} \end{aligned} \tag{3}$$

  Function $\delta(t)$ is the Dirac delta function, $\delta(t) = 1$ for $t = 0$ and 0 otherwise. Let $F_{i-}(s)$ denote the Laplace transform of $f_{i-}(t)$.

$$F_{i-}(s) = \frac{\lambda_i}{1 - e^{-\lambda_i \tau_i}} \cdot \frac{1 - e^{-\tau_i(\lambda_i + s)}}{\lambda_i + s} \tag{4}$$

- Distribution of t_{in} (having duration larger than τ_i): It follows distribution $P_{i+}(t)$ which is as follows:

$$P_{i+}(t) = P\{t_{in} \leq t\} = P\{r \leq t | r > \tau_i\} = \frac{P\{r \leq t, r > \tau_i\}}{P\{r > \tau_i\}} = \frac{(P_i(t) - P_i(\tau_i))u(t - \tau_i)}{1 - P_i(\tau_i)} \tag{5}$$

  Figure 6 gives a schematic explanation of the derivation of $P\{r \leq t, r > \tau_i\}$ which is involved in the derivation of $P_{i+}(t)$. The "$+$" sign in the subscript of $P_{i+}(t)$ is a mnemonic for indicating variables that are larger (hence the plus sign) than a constant (here τ_i).

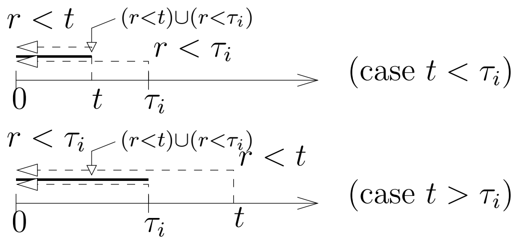

Figure 5: Schematic of the derivation of $P\{r < t, r < \tau_i\}$. The thick solid lines indicate the values of the random variable r that belong to the union $(r < t) \cup (r < \tau_i)$. $P\{r < t, r < \tau_i\}$ is the probability that an outcome belongs to this union.

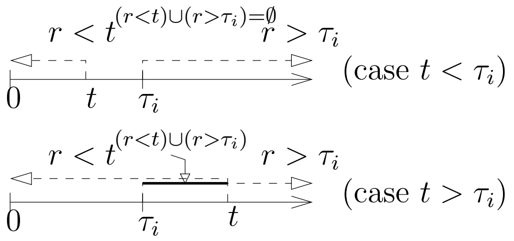

Figure 6: Schematic of the derivation of $P\{r < t, r > \tau_i\}$. The thick solid line indicates the values of the random variable r that belong to the union $(r < t) \cup (r > \tau_i)$. $P\{r < t, r > \tau_i\}$ is the probability that an outcome belongs to this union.

The corresponding density function is:

$$\begin{aligned} f_{i+}(t) &= \frac{dP_{i+}(t)}{dt} = \frac{(P_i(t))' \cdot u(t-\tau_i) + (P_i(t) - P_i(\tau_i)) \cdot (u(t-\tau_i))'}{1 - P_i(\tau_i)} \\ &= \frac{\phi_i(t) \cdot u(t-\tau_i) + (P_i(t) - P_i(\tau_i)) \cdot \delta(t-\tau_i)}{1 - P_i(\tau_i)} \\ &= \frac{\lambda_i e^{-\lambda_i t} \cdot u(t-\tau_i) + (e^{-\lambda_i \tau_i} - e^{-\lambda_i t}) \cdot \delta(t-\tau_i)}{e^{-\lambda_i \tau_i}} \end{aligned} \tag{6}$$

and its Laplace transform,

$$F_{i+}(s) = \frac{\lambda_i e^{-\tau_i s}}{\lambda_i + s} \tag{7}$$

Since $t_i = \sum_{j=1}^{n} t_{ij}$, the density function of $t_i|n$ is given by the convolution of (6) and the $(n-1)$th convolution of (3), i.e., $f_i(t|n) = f_{i+}(t) \otimes f_{i-}^{(n-1)}(t)$, where: $f_{i-}^{(n-1)}(t) = f_{i-}(t) \otimes \ldots \otimes f_{i-}(t)$ ($(n-1)$th convolution). The corresponding Laplace transform is $F_i(s|n) = (F_{i-}(s))^{n-1} F_{i+}(s)$. The density function of t_i can be written by removing the dependence on n.

$$f_i(t) = \sum_{n=1}^{\infty} f_i(t|n) \cdot (P_i(\tau_i))^{n-1} \cdot (1 - P_i(\tau_i)) \tag{8}$$

and its Laplace transform,

$$\begin{aligned} F_i(s) &= \int_0^{\infty} \sum_{n=1}^{\infty} f_i(t|n) \cdot (P_i(\tau_i))^{n-1} \cdot (1 - P_i(\tau_i)) \cdot e^{-st} dt \\ &= (1 - P_i(\tau_i)) \sum_{n=1}^{\infty} (P_i(\tau_i))^{n-1} \int_0^{\infty} f_i(t|n) \cdot e^{-st} dt \\ &= (1 - P_i(\tau_i)) \cdot \sum_{n=1}^{\infty} (P_i(\tau_i))^{n-1} \cdot F_i(s|n) = \frac{\lambda_i}{\lambda_i + s \cdot e^{\tau_i (s + \lambda_i)}} \end{aligned} \tag{9}$$

The average miss interval for object i, $T_i = E\{t_i\}$, is readily available by taking the derivative of $-F_i(s)$ and evaluating it at $s = 0$:

$$T_i = -F_i'(0) = \lambda_i^{-1} e^{\lambda_i \tau_i} \tag{10}$$

Now $\tilde{\pi}_i$ can be written as follows:

$$\tilde{\pi}_i = \frac{T_i - E\{h_i\}}{T_i} = 1 - \frac{E\{h_i\}}{T_i} = 1 - \frac{\lambda_i^{-1}}{\lambda_i^{-1} e^{\lambda_i \tau_i}} = 1 - e^{-\lambda_i \tau_i} \tag{11}$$

The variable h_i denotes the residual time from the time that i gets evicted from the cache, up to the next request for i (that results in a miss, see Fig. 4). Due to the fact that request inter-arrivals are exponentially distributed, $E\{h_i\} = \lambda_i^{-1}$.

Finally, it may be verified that the density of inter-miss intervals, $f_i(t)$, does not have a simple representation (see that the $F_i(s)$ of (9) does not have a compact inverse representation). It has been shown in [1] that a good compact approximation of $f_i(t)$ may be achieved by a truncated exponential density, i.e., $f_i(t) = \sigma_i \cdot e^{-\sigma_i (t - \tau_i)} \cdot u(t - \tau_i)$, where $\sigma_i = \frac{1}{T_i - \tau_i}$.

4.2 A simpler derivation of stationary behavior

In this section we show that if the objective is to study only the parts of stationary behavior that relate to the most commonly employed performance metric – the expected hit ratio over all requests – then a much simpler analysis becomes possible. To derive the hit ratio, one does not need to know the probability of finding an object in the cache at arbitrary times ($\tilde{\pi}_i$), but only know this probability upon the time instances that this object is requested; we denote this (conditional) probability for object i as π_i, and refer to it as the hit-ratio of i. We shall show that π_i can be derived much simpler than $\tilde{\pi}_i$.

See that between two successive misses for object i a total of n requests for it have occurred; of these $n - 1$ are hits and 1 is a miss. Thus, $\pi_i = \frac{E\{n\}-1}{E\{n\}}$. From (8) it is obvious that n is geometrically distributed with (success) parameter $(1 - P_i(\tau_i))$. Thus, $E\{n\} = 1/(1 - P_i(\tau_i))$ and hence,

$$\pi_i = 1 - 1/E\{n\} = P_i(\tau_i) = 1 - e^{-\lambda_i \tau_i} \tag{12}$$

Comparing this result with eq. (11) we realize that $\pi_i = \tilde{\pi}_i$, i.e., the conditional probability upon request arrivals times for i, and the unconditioned one at arbitrary times are, in fact, the

Notation	Definition
N, C	total # objects, cache capacity
π_i, λ_i	request probability, rate, for i
$P_i(t), \phi_i(t)$	exponential CDF and PDF with param. λ_i
t_i, T_i, τ_i	inter-miss interval for i, expected value of t_i, characteristic time for i
$t_{ij}, 1 \le j \le n-1$, t_{in}	hit-hit interval for i, hit-miss interval for i
$P_{i-}(t)$, $f_{i-}(t)$, $F_{i-}(s)$	CDF, PDF, Laplace transform, of $t_{ij}, 1 \le j \le n-1$
$P_{i+}(t)$, $f_{i+}(t)$, $F_{i+}(s)$	CDF, PDF, Laplace transform, of t_n
$f_i(t)$, $F_i(s)$	PDF, Laplace transform, of t_i
$\tilde{\pi}_i$, π_i	hit prob. for i at arbitrary times, and upon requests for i
^{1}, ^{2}	notational modifiers used to refer to level 1 or level 2
t_i^{off}, T_i^{off}, t_i^{on}, T_i^{on}	duration and expected value of OFF, ON states
$P_{i+}^{\text{on}}(t)$, $f_{i+}^{\text{on}}(t)$, $F_{i+}^{\text{on}}(s)$	CDF, PDF, Laplace transform, of miss-miss interval for i during ON state
$P_{i-}^{\text{on}}(t)$, $f_{i-}^{\text{on}}(t)$, $F_{i-}^{\text{on}}(s)$	CDF, PDF, Laplace transform, of miss-hit interval for i during ON state
ν_i^{on}	percentage of miss-miss intervals for i during ON state
$P_i^{\text{off}}(t)$, $\tilde{\pi}_i^{\text{off}}$	CDF of t_i^{off}, percentage of time on OFF state
$P_i^2(t)$	CDF for inter-arrivals for i at level 2
d^1, d^2, d^{os}	distance of level 1, level 2, origin server

Table 1: Notation summary. The notational modifiers ^{1}, ^{2} are used to refer to corresponding quantities of level 1 and level 2.

same. This is a consequence of the Poisson arrival process. Thus if the interest is only on the hit ratio, the current derivation is preferable as it is obviously much simpler (does not involve the various density functions and their transforms). The more elaborate analysis, however, has the advantage of characterizing the miss process also, which is required when studying interconnected caches (where the miss process at one cache contributes to the arrival process of an upstream one).

5 Analysis of a two-level LCD/LRU tandem

This section develops an approximate analytic model for LCD interconnected LRU caches.

5.1 Model description and assumptions

Figure 2 illustrates a two-level LCD/LRU tandem. User requests are received at the level 1 cache. Misses flow to the level 2 cache, and eventually to the origin server if neither cache holds the requested object. Both caches operate under LRU replacement and the LCD meta algorithm. The effect of the LCD meta algorithm on this particular tandem is that an object missing from both caches needs at least two requests to be brought to the level 1 cache; the first one brings it to the level 2 cache, and a subsequent one (assuming that it still resides at level 2) brings it to level 1.

As in Sect. 4, the aggregate request arrival process at the level 1 is assumed to be Poisson with mean rate λ^1, thus the per-object arrival processes are also Poisson with rates $\lambda_i^1 = p_i^1\lambda^1$; p_i^1 denotes the request probability of the ith most popular object. Objects are of unit size, and cache capacities are integral. We analyze the tandem by starting from the level 1 cache (the leaf) and then moving to the level 2 cache (the root).

5.2 Level 1 cache (leaf)

To analyze the leaf cache we focus on object i, and identify the two states that completely characterize the caching behavior as pertaining to this object:

OFF state : During the OFF state, object i is either cached at level 1, or has been evicted but not requested yet. Thus all requests for an object that is in its OFF state lead to hits and no request overflows to the second level. The characterization OFF refers to the fact that the miss process that leads to the second level is turned off for the particular object. Let t_i^{off} denote the duration of OFF.

ON state : The ON state follows the OFF state. During ON, object i is not cached at level 1, thus all requests flow to level 2. If i is cached at level 2, then it is brought to level 1 thus ending the current ON state. ON here refers to the fact that the miss process is turned on, feeding level 2 with requests. Let t_i^{on} denote the duration of ON.

Figure 7 illustrates the two states, demonstrating the fact that the behavior with respect to object i is fully described by a succession of OFF-ON cycles. Following an OFF-ON cycle for the tagged object, the system returns to the beginning of a new OFF state for this object. Thus it suffices to study this cycle to completely characterize the system. Notice that an ON state might have zero duration. This is the case when the miss at the end of the OFF

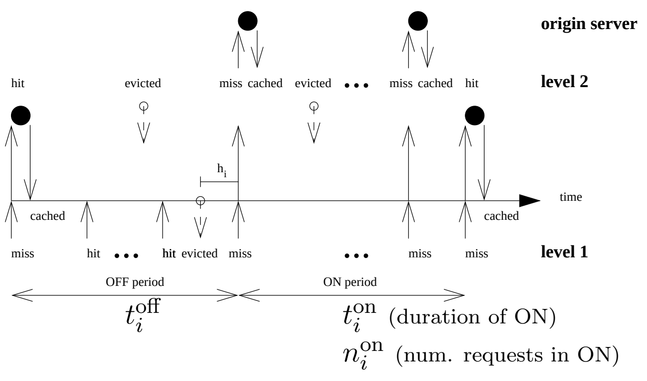

Figure 7: Time diagram of a two-level LCD/LRU tandem.

state finds the level 2 cache with a copy of i which is brought to level 1, thus instantaneously returning it to the beginning of a new OFF period.

5.2.1 Duration of OFF

The OFF interval has a duration t_i^{off} and involves two level 1 misses for i, at the beginning and the end, and a number of level 1 hits that come in-between. The distribution of t_i^{off} may be obtained by following the exact same steps as with a single LRU cache, as elaborated in Sect. 4. Thus the expected duration of OFF for object i is,

$$T_i^{\text{off}} = (\lambda_i^1)^{-1} e^{\lambda_i^1 \tau_i^1} \tag{13}$$

The LCD algorithm, however, affects the way that the characteristic time τ_i^1 is computed. In the case of an isolated cache, a miss leads directly to the caching of the requested object and this fact is inherent in eq. (1) that gives the characteristic time in the isolated case. The LCD case, however, is different. A miss at level 1 leads to local caching of the requested object only when this object resides at level 2; the object is not cached at level 1 if it only resides at the origin server at the time of the miss. To find the characteristic time, we employ a mean value approximation that is similar in conception to eq. (1), but is adapted to the LCD case:

$$\sum_{j=1, j\neq i}^{N} \pi_j^1 \cdot P_j^1(\tau_i^1) + (1-\pi_j^1) \cdot \pi_j^2 \cdot P_j^1(\tau_i^1) = C^1 \tag{14}$$

A verbal description of (14) would be that τ_i^1 is the interval in which exactly C^1 distinct objects j, $j \neq i$, that are either cached at the leaf or miss at the leaf but are cached at the root, are requested and, thus, cause the eviction of i. By observing equation (14) one may see that LCD enlarges the characteristic time τ_i^1 as compared to the case of pure LRU. This can be easily understood by noting that $\pi_j^1 + (1 - \pi_j^1) \cdot \pi_j^2$ i.e., the sum of the multipliers of $P_j^1(\tau_i^1)$'s in (14) is less than 1, whereas the single multiplier of $P_j(\tau_i)$ in (1) is equal to 1.

A second important observation is that whereas eq. (1) gives an exact expression for the expected time for C^1 distinct/unique non-tagged insertions to occur at the head of LRU, eq. (14) gives an approximate expression. This is because due to LCD, non-tagged requests do not necessarily trigger insertions of non-tagged objects. Thus although request inter-arrivals are memoryless, inter-insertion intervals are not. Thus the starting point $t = 0$ for counting for C^1 distinct/unique insertion of non-tagged objects, matters. Disregarding the elapsed time of on-going non-tagged inter-insertions at $t = 0$ makes eq. (14) an approximate expression for τ_i^1. This approximation is made to preserve the tractability of the model and, as will be shown in Sect. 6, leads to accurate results.

5.2.2 Duration of ON

The miss at the end of the OFF period signifies the beginning of the ON period. If the requested object resides at level 2, then it is immediately transmitted to level 1, and the ON period terminates instantaneously. Otherwise, the object descends from the origin server to the level 2 cache,[3] and it takes additional requests for the same object to get to level 1. The number of additional requests depends on whether the next request for the same object at level 1 (which is a miss), occurs before the object is evicted from level 2. Once again, it is through the use of the characteristic time that the duration of the ON period is computed. The rationale is as follows. To have n requests for i in the ON period, it will require that $n - 1$ inter-arrivals for i at level 1 (which flow to level 2 as they are misses) be larger[4] than the characteristic time at level 2 (τ_i^2), and the last one be smaller than the characteristic time at level 2. This is much like an inverse analogy[5] of the situation discussed in Sect. 4, with the

[3]Since the origin server always holds the requested object, it only takes a single miss to bring an object to level 2.

[4]Thus each time the first request of an inter-arrival brings i to level 2 from the origin server, but the next request comes too late, when i has been evicted from level 2. Thus the second request instead of bringing i to level 1, it just brings it back to level 2 from the origin server.

[5]Here $n - 1$ intervals are larger than the char. time, and 1 is smaller, whereas in Sect. 4, $n - 1$ were smaller and 1 larger.

additional difference that inter-arrivals of one level are compared to the characteristic time of a different level (the above one), whereas in the original case, inter-arrivals and characteristic time refer to the same cache.

By conditioning on whether an inter-arrival for i at level 1 is larger or smaller than τ_i^2, we get the following results (look at the corresponding derivations of Sect. 4 and the notation summary on Table 1 for details):

$$\begin{aligned}
P_{i-}^{\mathrm{on}}(t) &= P\{r \le t | r \le \tau_i^2\} = \frac{P_i(t)u(\tau_i^2 - t) + P_i(\tau_i^2)(1 - u(\tau_i^2 - t))}{P_i(\tau_i^2)} \\
f_{i-}^{\mathrm{on}}(t) &= \frac{\lambda_i^1 e^{-\lambda_i^1 t} \cdot u(\tau_i^2 - t) - (1 - e^{-\lambda_i^1 t}) \cdot \delta(\tau_i^2 - t) + (1 - e^{-\lambda_i^1 \tau_i^2}) \cdot \delta(t - \tau_i^2)}{1 - e^{-\lambda_i^1 \tau_i^2}} \\
F_{i-}^{\mathrm{on}}(s) &= \frac{\lambda_i^1}{1 - e^{-\lambda_i^1 \tau_i^2}} \cdot \frac{1 - e^{-\tau_i^2(\lambda_i^1 + s)}}{\lambda_i^1 + s} \\
P_{i+}^{\mathrm{on}}(t) &= P\{r \le t | r > \tau_i^2\} = \frac{(P_i(t) - P_i(\tau_i^2))u(t - \tau_i^2)}{1 - P_i(\tau_i^2)} \\
f_{i+}^{\mathrm{on}}(t) &= \frac{\lambda_i^1 e^{-\lambda_i^1 t} \cdot u(t - \tau_i^2) + (e^{-\lambda_i^1 \tau_i^2} - e^{-\lambda_i^1 t}) \cdot \delta(t - \tau_i^2)}{e^{-\lambda_i^1 \tau_i^2}} \\
F_{i+}^{\mathrm{on}}(s) &= \frac{\lambda_i^1 e^{-\tau_i^2 s}}{\lambda_i^1 + s}
\end{aligned} \tag{15}$$

Following the aforementioned description of an ON period of n requests, we can write:

$$\begin{aligned}
F_i^{\mathrm{on}}(s|n) &= (F_{i+}^{\mathrm{on}}(s))^{n-1} F_{i-}^{\mathrm{on}}(s) && \text{(Laplace transform of } t_i^{\mathrm{on}}|n) \\
f_i^{\mathrm{on}}(t) &= \sum_{n=1}^{\infty} f_i^{\mathrm{on}}(t|n) \cdot (1 - P_i(\tau_i^2))^{n-1} \cdot P_{ki}(\tau_i^2) && \text{(density function of } t_i^{\mathrm{on}}) \\
F_i^{\mathrm{on}}(s) &= \frac{\lambda_i^1 (1 - e^{-\tau_i^2(s + \lambda_i^1)})}{\lambda_i^1 - (s + \lambda_i^1) e^{\tau_i^2(s + \lambda_i^1)}} && \text{(Laplace transform of } t_i^{\mathrm{on}}) \\
T_i^{\mathrm{on}} &= E\{t_i^{\mathrm{on}}\} = -\acute{F}_i^{\mathrm{on}}(0) = \frac{e^{\tau_i^2 \lambda_i^1}}{\lambda_i^1 (e^{\tau_i^2 \lambda_i^1} - 1)} && \text{(expected duration of ON period)}
\end{aligned} \tag{16}$$

5.2.3 Stationary behavior at level 1

It is now possible to write the probability of finding object i at level 1 (at both request and arbitrary epochs):

$$\begin{aligned}\pi_i^1 = \tilde{\pi}_i^1 &= \frac{T_i^{\text{off}} - E\{h_i\}}{T_i^{\text{off}} + T_i^{\text{on}}} \\ &= \frac{T_i^{\text{off}} - (\lambda_i^1)^{-1}}{T_i^{\text{off}} + T_i^{\text{on}}} \\ &= \frac{(\lambda_i^1)^{-1} \cdot e^{\lambda_i^1 \tau_i^1} - (\lambda_i^1)^{-1}}{(\lambda_i^1)^{-1} \cdot e^{\lambda_i^1 \tau_i^1} + (\lambda_i^1)^{-1} (e^{\lambda_i^1 \tau_i^2} - 1)^{-1} e^{\lambda_i^1 \tau_i^2}} \\ &= \frac{e^{\lambda_i^1 \tau_i^1} - 1}{e^{\lambda_i^1 \tau_i^1} + (e^{\lambda_i^1 \tau_i^2} - 1)^{-1} e^{\lambda_i^1 \tau_i^2}}\end{aligned} \tag{17}$$

Observe that the steady-state probability of finding i at level 1 depends on the characteristic time at level 1 (τ_i^1), as well as on the characteristic time at level 2 (τ_i^2), whereas in an isolated LRU cache (or in a level 1 cache of a hierarchy that operates under LCE) it is only the local characteristic time that matters.

5.3 Level 2 cache (root)

The study of the level 2 cache starts with the characterization of the request arrival process of the tagged object i.

5.3.1 The arrival process at level 2

The sole contributor to this arrival process is the miss process from level 1. Referring back to Fig. 7, it may be seen that the inter-arrivals at level 2 are either inter-miss intervals of the OFF period (each OFF period at level 1 creates one inter-arrival for level 2), or inter-miss intervals during the ON period at level 1. Inter-arrivals of the first kind (OFF period) can be modeled by a truncated exponential distribution, like the one used in Sect. 4.1. Inter-arrivals of the second kind (ON period), follow conditional exponential distributions (the condition expressed with respect to τ_i^2). We, therefore, approximate the distribution of r_i^2, the inter-arrivals for i at level 2, by the following *mixture* distribution:

$$P_i^2(t) = P\{r_i^2 < t\} = \tilde{\pi}_i^{\text{off}} \cdot P_i^{\text{off}}(t) + (1 - \tilde{\pi}_i^{\text{off}}) \cdot (\nu_i^{\text{on}} \cdot P_{i+}^{\text{on}}(t) + (1 - \nu_i^{\text{on}}) \cdot P_{i-}^{\text{on}}(t)) \tag{18}$$

where,

$$\begin{aligned}
\tilde{\pi}_i^{\text{off}} &= \frac{T_i^{\text{off}}}{T_i^{\text{off}} + T_i^{\text{on}}} = \frac{e^{\lambda_i^1 \tau_i^1}}{e^{\lambda_i^1 \tau_i^1} + (e^{\lambda_i^1 \tau_i^2} - 1)^{-1} e^{\lambda_i^1 \tau_i^2}} \\
P_i^{\text{off}}(t) &= (1 - e^{-\sigma_i^{\text{off}}(t - \tau_i^1)}) \cdot u(t - \tau_i^1) \quad \text{with} \quad \sigma_i^{\text{off}} = \tfrac{1}{T_i^{\text{off}} - \tau_i^1} \\
\nu_i^{\text{on}} &= \frac{E\{n_i^{\text{on}}\} - 1}{E\{n_i^{\text{on}}\}} = e^{-\lambda_i^1 \tau_i^2} \quad \left(\text{where} \quad E\{n_i^{\text{on}}\} = \sum_{n_i^{\text{on}}=1}^{\infty} (1 - P_i^1(\tau_i^2))^{n_i^{\text{on}} - 1} \cdot P_i^1(\tau_i^2) = \frac{1}{P_i^1(\tau_i^2)} \right)
\end{aligned} \tag{19}$$

$\tilde{\pi}_i^{\text{off}}$ denotes the percentage of time that level 1 spends in the ON state with respect to object i. $P_i^{\text{off}}(t)$ is a truncated exponential distribution that is used to approximate the actual distribution of OFF intervals. ν_i^{on} denotes the percentage of requests that go into the ON period, without succeeding to bring i to level 1. Figure 7 shows that n_i^{on} requests for i go into an ON period, of which only the last one brings i to level 1. It is easy to see that n_i^{on} is geometrically distributed with parameter $P_i^1(\tau_i^2)$, hence $E\{n_i^{\text{on}}\} = 1/P_i^1(\tau_i^2)$. A percentage ν_i^{on} of the total requests that go into ON are exponential intervals that are known to be larger than τ_i^2; their distribution, $P_{i+}^{\text{on}}(t)$, has been given in (15). The remaining percentage, $1 - \nu_i^{\text{on}}$, involves exponential intervals that are smaller than τ_i^2; their distribution, $P_{i-}^{\text{on}}(t)$, has been given in (15) (such intervals bring i to level 1 thus ending the ON period).

5.3.2 The characteristic time at level 2

As far as the caching behavior is concerned, the level 2 cache operates in the usual LRU manner, i.e., a miss leads to the immediate caching of the requested object. This happens because the upstream neighbor of level 2 is the origin server which, by holding all the objects, leads to the immediate caching at level 2 of all misses that occur at that level. Due to the existence of the origin server, the LCD meta algorithm reduces to the normal caching operation for the level 2 cache. Therefore, the characteristic time at level 2, τ_i^2, may be approximated (request inter-arrivals are no longer exponential) by solving:

$$\sum_{j=1, j \neq i}^{N} P_j^2(\tau_i^2) = C^2 \tag{20}$$

5.3.3 Stationary behavior at level 2

The stationary probability of finding object i at level 2, upon a request for it at level 2, π_i^2, can be obtained by applying the method of Sect. 4.2, thus

$$\pi_i^2 = \frac{E\{n_i^2\} - 1}{E\{n_i^2\}} = 1 - \frac{1}{E\{n_i^2\}} = 1 - \frac{1}{1/(1 - P_i^2(\tau_i^2))} = P_i^2(\tau_i^2) \tag{21}$$

Here, n_i^2, denotes the number of requests for i that must occur from one miss until the next one, at level 2; it is geometrically distributed with (success) parameter $(1 - P_i^2(\tau_i^2))$. Using (18) to write $P_i^2(\tau_i^2)$ (and noting that $P_{i-}^{\text{on}}(\tau_i^2) = 1$ and $P_{i+}^{\text{on}}(\tau_i^2) = 0$) and then substituting $\tilde{\pi}_i^{\text{off}}$, $P_i^{\text{off}}(\tau_i^2)$ and ν_i^{on} from (19), equation (21) becomes:

$$\begin{aligned}\pi_i^2 &= \tilde{\pi}_i^{\text{off}} \cdot P_i^{\text{off}}(\tau_i^2) + (1 - \tilde{\pi}_i^{\text{off}}) \cdot (1 - \nu_i^{\text{on}}) \\ &= \frac{e^{\lambda_i^1 \tau_i^1} \cdot (1 - e^{-\sigma_i^{\text{off}}(\tau_i^2 - \tau_i^1)}) \cdot u(\tau_i^2 - \tau_i^1)}{e^{\lambda_i^1 \tau_i^1} + (e^{\lambda_i^1 \tau_i^2} - 1)^{-1} e^{\lambda_i^1 \tau_i^2}} + \left(1 - \frac{e^{\lambda_i^1 \tau_i^1}}{e^{\lambda_i^1 \tau_i^1} + (e^{\lambda_i^1 \tau_i^2} - 1)^{-1} e^{\lambda_i^1 \tau_i^2}}\right)(1 - e^{-\lambda_i^1 \tau_i^2})\end{aligned} \tag{22}$$

For the level 2 cache we can again derive the miss process by applying the analysis of Sect. 4. The method is the same: define the conditional distributions, take their transforms, condition with respect to n_i^2, remove the dependence on n_i^2 to get the Laplace transform of the inter-miss distribution, find a compact inversion formula and use it for the arrivals at the next level. For all the aforementioned quantities, closed-form expressions may be derived.

Having derived π_i^1 and π_i^2, it becomes possible to compute the average hit distance in the tandem (2 caches plus the origin server):

$$E\{\text{hit distance}\} = \sum_{i=1}^{N} p_i^1 \cdot (d^1 \pi_i^1 + d^2 (1 - \pi_i^1) \pi_i^2 + d^{\text{os}} (1 - \pi_i^1)(1 - \pi_i^2)) \tag{23}$$

5.4 Decoupling the tandem

Although closed form expressions have been derived for π_i^1 and π_i^2, it is not possible to use them directly, as the two caches are *coupled* under LCD, i.e., the state of one depends on the state of the other and vise versa. There also exist local couplings.

If the tandem were to involve LRU caches operating under LCE, then it would be possible to work in a bottom up manner and derive the steady-state at each level. The LCD algorithm, however, brings additional complications. To derive the steady-state behavior at level 1 under

LCD, one must know the steady-state behavior at level 2 also. This is because an object insertion after a miss at level 1, depend under LCD on whether the requested object exists at level 2. In various stages during the analysis of level 1, the steady-state behavior of level 2 had to be involved: (i) the derivation of the characteristic time at level 1 involved the π_j^2's (eq.(14)); (ii) the analysis of the ON period was based on conditioning on τ_i^2's (eqs. (15),(16)). As a result π_i^1 (eq. (17)) depends on level 2 explicitly, as it involves τ_i^2, and also implicitly, as its own τ_i^1 is the solution of eq. (14) that involves π_i^2. There also exist local couplings between τ_i^1's and π_i^1's due to eqs. (14) and (17).

The level 2 cache, naturally depends on level 1, as its arrival process is the miss process of the downstream cache. To resolve the couplings, we resort to an iterative method which works as follows. It starts with an initially uncoupled tandem (i.e., it assumes that the tandem operates under LCE). The LCE tandem can be worked in a bottom up manner and thus get the characteristic times and steady-state hit probabilities at both levels. Then it starts an iteration where the LCD formulas are employed, and the LCE results provide the starting point. A stopping rule is defined for convergence. The method is described below.

Initialization : Solve an uncoupled tandem for $\tau_i^1(0)$, $\pi_i^1(0)$, $\tau_i^2(0)$, $\pi_i^2(0)$ using the method of Sect. 4 (see also [1]).

Iteration n : For iteration n, $n \geq 1$, perform the following steps:

Step 1 : Solve for $\tau_i^1(n)$, $\pi_i^1(n)$, $\tau_i^2(n)$, $\pi_i^2(n)$ using the results of iteration $(n-1)$, and the following formulas (which are adapted versions of formulas already presented):

$$\sum_{j=1,j\neq i}^{N} \pi_j^1(n-1)\cdot P_j^1(\tau_i^1(n)) + (1-\pi_j^1(n-1))\cdot \pi_j^2(n-1)\cdot P_j^1(\tau_i^1(n)) = C^1 \qquad \text{(solve for } \tau_i^1(n))$$

$$\pi_i^1(n) = \frac{e^{\lambda_i^1\tau_i^1(n)}-1}{e^{\lambda_i^1\tau_i^1(n)} + (e^{\lambda_i^1\tau_i^2(n-1)}-1)^{-1}e^{\lambda_i^1\tau_i^2(n-1)}}$$

$$\sum_{j=1,j\neq i}^{N} P_j^2(\tau_i^2(n)) = C^2 \quad \text{(solve for } \tau_i^2(n);\ \text{use } P_j^2(t) \text{ from (18) with } \tau_i^1=\tau_i^1(n), \tau_i^2=\tau_i^2(n-1))$$

$$\pi_i^2(n) = \frac{e^{\lambda_i^1\tau_i^1(n)}(1-e^{-\sigma_i^{\text{off}}(\tau_i^2(n)-\tau_i^1(n))})u(\tau_i^2(n)-\tau_i^1(n))}{e^{\lambda_i^1\tau_i^1(n)} + (e^{\lambda_i^1\tau_i^2(n)}-1)^{-1}e^{\lambda_i^1\tau_i^2(n)}} + \left(1-\frac{e^{\lambda_i^1\tau_i^1(n)}}{e^{\lambda_i^1\tau_i^1(n)} + (e^{\lambda_i^1\tau_i^2(n)}-1)^{-1}e^{\lambda_i^1\tau_i^2(n)}}\right)(1-e^{-\lambda_i^1\tau_i^2(n)}) \tag{24}$$

Step 2 : Convergence test:

$$\max(\boldsymbol{e^1(n)}, \boldsymbol{e^2(n)}) \leq \epsilon \tag{25}$$

where $\boldsymbol{e^1(n)} = |\boldsymbol{\pi^1(n)} - \boldsymbol{\pi^1(n-1)}|$ and $\boldsymbol{e^2(n)} = |\boldsymbol{\pi^2(n)} - \boldsymbol{\pi^2(n-1)}|$. The max function returns the largest element in any of the two vectors (each one having elements $|\pi_i^x(n) - \pi_i^x(n-1)|$, $1 \leq i \leq N$). The constant ϵ denotes the (user-defined) tolerance for the convergence of the iterative method. If condition (25) holds true, then set: $\pi_i^1 = \pi_i^1(n)$, $\pi_i^2 = \pi_i^2(n)$, $1 \leq i \leq N$, and exit the iteration. Else, set: $\tau_i^1(n-1) = \tau_i^1(n)$, $\pi_i^1(n-1) = \pi_i^1(n)$, $\tau_i^2(n-1) = \tau_i^2(n)$, $\pi_i^2(n-1) = \pi_i^2(n)$, and perform another iteration.

Finalization Compute the average hit distance in the tandem through eq. (23).

5.5 Handling multiple clients - an LCD/LRU tree

The analysis of the LCD/LRU tandem in Sect. 5, assumes the existence of a single client that connects to the lowest level; this is the exact model of hierarchical memory. There are cases, however, that multiple clients, each with its own level 1 cache, connect to a shared level 2 cache; such an example is a hierarchical web cache. In the latter case, caches form a tree topology, rather than a tandem. How could such a topology be analyzed?

For that purpose, it is possible to use a slightly modified version of the analysis of Sect. 5 for tandems. The key observation is that in a tree, level 1 caches remain unaffected, and it is only the level 2 cache that is affected from the presence of multiple clients. Still, only a slight modification to the analysis of the level 2 cache is required; the arrival process at level 2 (Sect.5.3.1) needs to be adapted to multiple clients. Equation (15) in [1] gives the expression for the aggregate inter-arrival distribution at the root under LCE and multiple miss streams (each miss stream approximated by a truncated exponential distribution). The same equation applies to the LCD case, with the only difference be that here the individual miss streams are approximated by the mixture distribution of eq. (18), rather than by a truncated exponential distribution. An analysis similar to the tandem case can then be carried out.

5.6 Summary of the analysis and discussion

This section summarizes the concepts and tools utilized in the analysis of LCD/LRU. Despite that LRU itself is a complicated system (from an analytical point of view), and the fact that LCD brings additional complications to its analysis, the presented model is able to predict accurately the performance of LCD/LRU as will be shown in Sect. 6. The ability to do so, owes to the use of a number of carefully selected techniques that approximate satisfactorily the real system, while remaining tractable. These key techniques are summarized bellow:

- The mean value approximation is used to overcome the combinatorial hardness of analyzing an isolated LRU. By considering the characteristic time to be a constant, the per-object hit probabilities are expressed in a simple manner, without however sacrificing the important dependencies between different object, which are retained through eq. (1) that caters to the individual request probabilities.

- Object requests are i.i.d. random variables at the leaf level, whereas they become correlated at the root level (due to the mediation of the leaf cache). This is a common feature in many interconnected systems (e.g., in networks of queues, which in fact are simpler than networks of LRU's as the former involve only one type of costumer, whereas the latter involve multiple types). In the case of the LCE interconnection, the correlated request process for a particular object i at level 2, is approximated by an uncorrelated one. This is done by approximating the inter-arrivals by a truncated exponential distribution. By using the characteristic time of level 1 as the truncation constant, it becomes possible to retain the most important characteristics of the miss process that flows to level 2. This idea is also employed in the analysis of LCD, in order to capture the distribution of inter-arrivals at the two ends of an OFF period.

- The LCD algorithm adds a significant amount of complication as it does not permit all misses to be cached instantly, but rather places conditions for this to happen; such conditions involve information on the state of the above level. The key concepts for handling LCD are: (1) the identification of the OFF-ON repeated cycles; (2) the adoption of a modified expression for the equation that gives the characteristic time of level 1.

- Having written the characteristic times and hit probabilities at both levels, it remains to overcome the coupling due to the bidirectional dependency between the two levels. This is handled by the iterative procedure of Sect. 5.4 that starts with an initially uncoupled system and progressively adds the true dependencies that exist. Such methods have been used in the past for the study of coupled queues, e.g., in Modiano and Ephremides [23].

Although the analysis has been centered around the LCD algorithm, the presented framework might have a broader application. Other types of inter-cache dependencies can be modelled by retaining most of the analysis, and adapting only the distribution of the ON period and the equation for the characteristic time at level 1. The ON period must capture the amount of time that an object is not allowed to descend to level 1 due to dependencies with other caches. The equation for the characteristic time must reflect the effects of these dependencies in the expected amount of time that is required to evict an object from the level 1 cache.

6 Numerical Results

To assess the accuracy and the predictive power of the analytic model for the LCD/LRU tandem, analytic numerical results are compared against simulation results. The comparison is carried out under Zipf-like requests. The topology is as shown in Fig. 2, and the access distances are $d^1 = 0$, $d^2 = 1$, $d^{\mathrm{os}} = 2$ (level 1/level 2/origin server). Such a selection of distances amount to a hop-count distance, where the clients are co-located with the level 1 cache. The average hit distance is computed using eq. (23).

We have chosen to test the accuracy of the analytic model under Zipf-like requests, because it is under such highly skewed distributions that it becomes challenging to estimate the performance analytically. Should the popularity of objects be uniform, it would be possible to substitute all p_i^1's, $1 \leq i \leq N$, with a single value $p = 1/N$, and thus carry out a much simpler analysis. Our approach was not to make any simplifying assumptions regarding popularity during the analysis, thus construct a model that is more versatile.

Figure 8 depicts the average hit distance in a two-level LCD/LRU tandem with $N = 100$, $C^1 = C^2 = C$, and Zipf-like requests ($a = 0.6$ and $a = 0.9$), as given by simulation and analysis. Even for such a small population of objects, and small cache sizes, the analytic

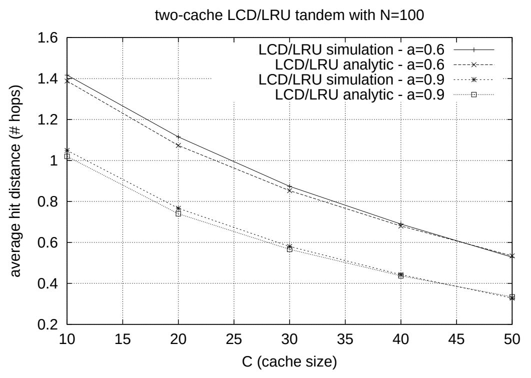

Figure 8: Comparison of simulation and analytic results for a two-cache LCD/LRU tandem under Zipf-like requests (two different values for the skewness parameter a).

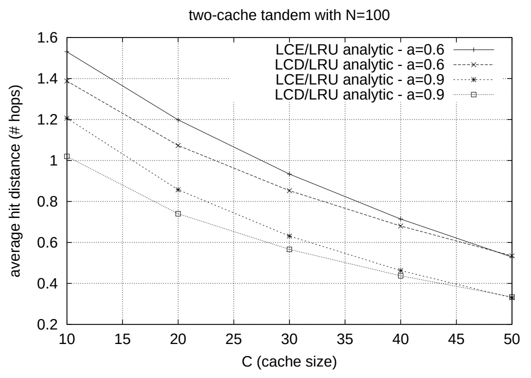

Figure 9: Comparison of LCE/LRU and LCD/LRU two-cache tandem under Zipf-like requests (two different values for the skewness parameter a).

results are always within 2-3% of the simulation results.

Figure 9 compares the performance of LCD/LRU against that of LCE/LRU, using analysis. For the later, the results of Sect. 4 are used (see also the original analysis in [1]). The figure shows that LCD/LRU has a performance that is constantly superior across the depicted spectrum of cache sizes. The gap, however, closes with increasing C. This is expected, as under a high availability of storage, the gain that LCD achieves by judiciously accepting objects in the level 1 cache, competes with the cost paid for requiring multiple requests to bring an object to level 1. See, however, that for LCE to reach up to the performance of LCD, the two caches must by able to collectively hold the entire object population (see that the lines in Fig. 9 intersect when each cache holds 50 object, in a population of 100). This means that as long as caches are not enormously large (to hold all the content), LCD/LRU performs better than LCE/LRU, as also shown by previous simulations.

Figure 10 attempts to explain the superiority of LCD over LCE, by looking closely at the objects that reside in each cache. The top left graph in Fig. 10, shows the per-object hit probabilities, for the two levels of an LCE/LRU tandem, under a relatively small cache size, $C = 5$. By observing the two curves, one may see that the two caches hold approximately the same set of objects, with the level 1 cache having a somewhat larger probability of caching some of the most popular ones; the two curves, however, are strikingly similar.

The top right graph of Fig. 10 shows the corresponding steady-state probabilities for LCD/LRU and $C = 5$. The situation here is completely different. The curves for level 1 and

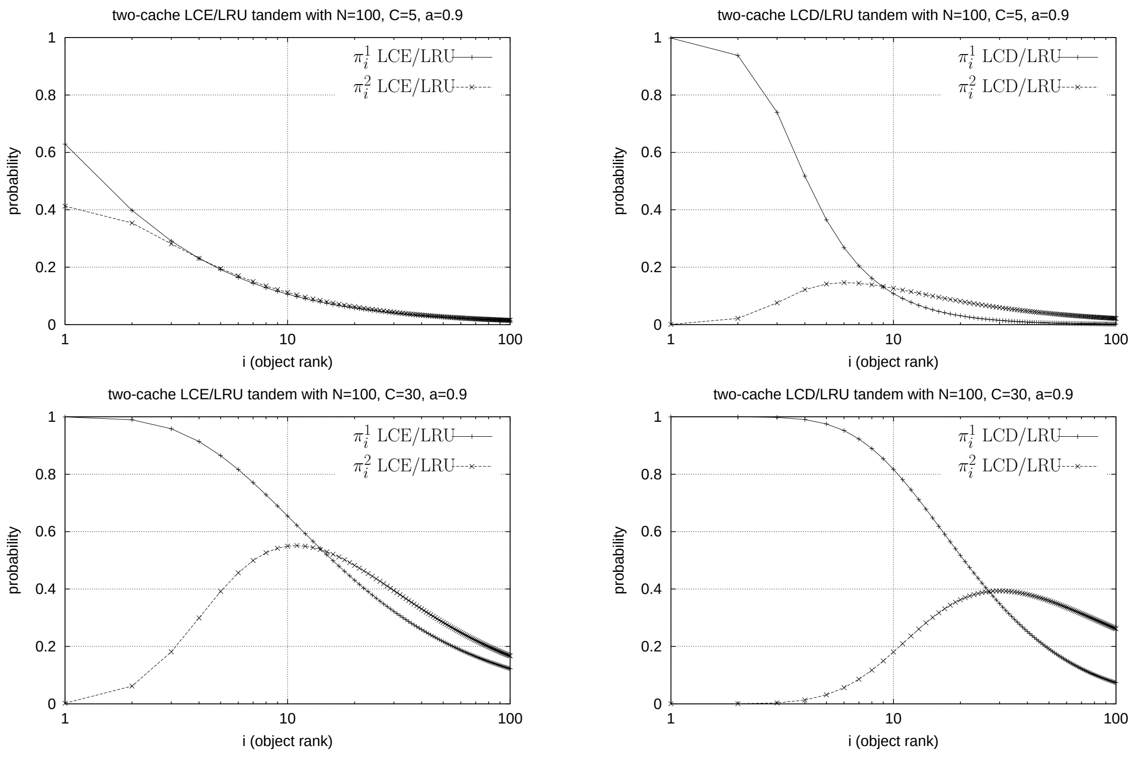

Figure 10: Level 1, π_i^1, and level 2, π_i^2, object hit ratios for LCE and LCD (for two different cache sizes, $C = 5$ and $C = 30$).

level 2 are distinctively different. The level 1 cache holds with a high probability the initial most popular object, and with a very low probability the objects that belong to the tail of the popularity distribution; see that objects 1 and 2 are almost guaranteed to be at level 1 under LCD, whereas they are at level 1 for half of the occasions under LCE (top left graph). The level 2 cache under LCD, behaves more rationally, granted the state of the underlying cache. It picks up the most popular objects, starting with those that miss frequently at level 1; Thus it caches most of the objects that belong to the middle section of the popularity distribution and avoids duplicating objects that have a high probability of being cached at level 1. It is interesting to note that under LCD, objects 1 and 2 are almost guaranteed to be at level 1 and also guaranteed not to be at level 2 (under LCE, objects 1 and 2 co-exist at both levels for much of the time). The behavior of LCD clearly demonstrates the property of cache exclusivity, as discussed in Sect. 2.2.2.

The bottom graphs of Fig. 10 depict the same results under a significantly larger cache size, $C = 30$ (meaning that the two caches can collectively hold up to 60% of all objects). The high availability of storage in this cases, leads to a caching behavior that is similar for both LCE and LCD. Having more storage at level 1, combined with the skewed demand, allows even LCE to have the popular object at level 1 in most occasions. This filters out most requests that go to level 2, thus approaching the behavior under LCD. Still, LCD does a better job in keeping the most popular objects only at level 1, and the subsequent most popular ones only at level 2. Increasing further the cache size makes the behavior of the two policies even more similar; this explains why LCE and LCD yield the same average hit distance for $C = 50$ (Fig. 9).

7 Conclusions

This paper has taken an analytic look at the LCD way of interconnecting LRU caches. Despite its simplicity, LCD has demonstrated surprisingly good performance under various workloads and interconnection topologies. Aiming at explaining the results of simulation experiments, we have constructed an analytic performance evaluation model. Despite the simplicity of the LCD algorithm itself, its exact analysis under LRU replacement is far beyond the borders of tractability. For this reason, we have employed a number of novel approximation techniques. The result is a tractable approximate analytic model that has the ability to accurately predict the performance of a real LCD/LRU multi-level cache. Using this model, we study the caching behavior at the different levels of a multi-level cache, and explain why LCD performs better than other algorithms.

References

[1] Hao Che, Ye Tung, and Zhijun Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, Sept. 2002.

[2] Theodore M. Wong and John Wilkes, "My cache or yours? making storage more exclusive," in *Proc. of the USENIX Annual Technical Conference*, Monterey, CA, June 2002, pp. 161–175.

[3] Nikolaos Laoutaris, Sofia Syntila, and Ioannis Stavrakakis, "Meta algorithms for hierarchical web caches," in *IEEE International Performance Computing and Communications Conference (IEEE IPCCC)*, Phoenix, Arizona, Apr. 2004.

[4] James H. Hester and Daniel S. Hirschberg, "Self-organizing linear search," *ACM Computing Surveys*, vol. 17, no. 3, pp. 295–311, Sept. 1985.

[5] Stefan Podlipnig and Laszlo Böszörmenyi, "A survey of web cache replacement strategies," *ACM Computing Surveys*, vol. 35, no. 4, pp. 374–398, 2003.

[6] Konstantinos Psounis and Balaji Prabhakar, "Efficient randomized web-cache replacement schemes using samples from past eviciton-times," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 441–454, Aug. 2002.

[7] David Starobinski and David N. C. Tse, "Probabilistic methods for web caching," *Performance Evaluation*, vol. 46, no. 2-3, pp. 125–137, 2001.

[8] Anirban Mahanti, Carey Williamson, and Derek Eager, "Traffic analysis of a web proxy caching hierarchy," *IEEE Network Magazine*, vol. 14, no. 3, pp. 16–23, May 2000.

[9] Anirban Mahanti, Derek Eager, and Carey Williamson, "Temporal locality and its impact on web proxy cache performance," *Performance Evaluation*, vol. 42, pp. 187–203, 2000.

[10] Xueyan Tang and Samuel T. Chanson, "Coordinated en-route web caching," *IEEE Transactions on Computers*, vol. 51, no. 6, pp. 595–607, June 2002.

[11] Kang-Won Lee, Sambit Sahu, Khalil Amiri, and Chitra Venkatramani, "Understanding the potential benefits of cooperation among proxies: Taxonomy and analysis," Tech. Rep. RC22173, IBM Research, Sept. 2001.

[12] Lee Breslau, Pei Cao, Li Fan, Graham Philips, and Scott Shenker, "Web caching and Zipf-like distributions: Evidence and implications," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, New York, Mar. 1999.

[13] Maureen Chesire, Alec Wolman, Geoffrey M. Voelker, and Henry M. Levy, "Measurement and analysis of a streaming-media workload," in *Proceedings of USITS*, 2001.

[14] Stefan Saroiu, Krishna P. Gummadi, Richard J. Dunn, Steven D. Gribble, and Henry M. Levy, "An analysis of internet content delivery systems," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002)*, Dec. 2002.

[15] Kunwadee Sripanidkulchai, "The popularity of gnutella queries and its implication on scalability," 2001, white paper online at `http://www-2.cs.cmu.edu/~kunwadee/research/p2p/gnu`.

[16] Rajeev Motwani and Prabhakar Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.

[17] W.C. King, "Analysis of paging algorithms," in *Proceedings of the IFIP 1971 Congress*, Ljubljana, 1972, pp. 485–490.

[18] E. G. Coffman and P. J. Denning, *Operating Systems Theory*, Prentice-Hall Inc., 1980.

[19] Philippe Flajolet, Danièle Gardy, and Loys Thimonier, "Birthday paradox, coupon collectors, caching algorithms and self-organizing search," *Discrete Applied Mathematics*, vol. 39, pp. 207–229, 1992.

[20] Asit Dan and Dan Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proceedings of ACM SIGMETRICS*, 1990, pp. 143–152.

[21] Predrag R. Jelenković, "Asymptotic approximation of the move-to-front search cost distribution and least-recently-used caching fault probabilities," *Annals of Applied Probability*, vol. 9, no. 2, pp. 430 – 464, 1999.

[22] Predrag R. Jelenković and Ana Radovanović, "Asymptotic insensitivity of least-recently-used caching to statistical dependency," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, CA, Apr. 2003.

[23] Eytan Modiano and Anthony Ephremides, "A method for delay analysis of interacting queues in multiple access systems," in *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Francisco, CA, Mar. 1993, pp. 447–454.