Performance Evaluation 127–128 (2018) 194–211

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

Weighted fair caching: Occupancy-centric allocation for space-shared resources

Lianjie Shi, Xin Wang, Richard T.B. Ma *, Y.C. Tay

School of Computing, National University of Singapore, Singapore

ARTICLE INFO

Article history:
Available online 23 October 2018

Keywords:
Caching policy
Occupancy rate
Resource allocation

ABSTRACT

Traditional cache replacement policies such as LRU and LFU were often designed with the focus on efficiency and aimed at maximizing the hit rates. However, the resource owners of modern computing systems such as cloud infrastructures and content delivery networks often have new objectives such as fairness and revenue to be optimized rather than the overall hit rate. A general resource management framework that allows resource owners to determine various resource allocations is desirable. Although such a mechanism like Weighted Fair Queueing (WFQ) exists for indivisible time-shared resources such as CPU and network bandwidth, no such counterpart exists for space-shared resources such as cache and main memory. [1]

In this paper, we propose Weighted Fair Caching (WFC), a capacity-driven cache policy that provides explicitly tunable resource allocations for cache owners in terms of the occupancy rates of contents. Through analysis of the continuous-time Markov Chain model of cache dynamics, we derive the closed-form occupancy rates as a function of the weights of contents, and various properties such as monotonicity and scaling of WFC. We show that WFC can be used to provide fair sharing of cache space among contents, as well as class-based service differentiations. We evaluate the performance of WFC using real data traces from two major video providers. We find that, compared to traditional cache policies, WFC provides better fairness while sacrificing an acceptable amount of hit rates.

1. Introduction

Research on management and allocation of space-shared resources such as cache, memory and disk space have been centered around the placement and replacement of contents. In this paper, we mainly refer to space-shared resources as caches, and resource management mechanisms then appear as replacement policies. Since hit rate directly influences the latency or response time of content requests, traditional replacement policies such as least recently used (LRU) and least frequently used (LFU) were designed to maximize the overall hit rate of content requests. These designs and analysis were primarily focused on the system efficiency, assuming that the resource owner's goal is to optimize the system performance. Under the traditional replacement policies, the portion of resource occupied by consumers is usually not explicitly controllable by the resource owner, but more dependent on the consumers themselves. For instance, if the resource owner, i.e., the cache provider, has adopted LFU policy, some content that is requested more frequently than any one else

* Corresponding author.
E-mail addresses: shilian@comp.nus.edu.sg (L. Shi), xin.wang@comp.nus.edu.sg (X. Wang), tbma@comp.nus.edu.sg (R.T.B. Ma), dcstayyc@nus.edu.sg (Y.C. Tay).

1 This work was supported in part by Singapore MOE grant T1251RES1710.

https://doi.org/10.1016/j.peva.2018.09.011
0166-5316/

can stay in the cache almost permanently. In contrast, some other consumer, i.e., content provider, may never get the chance to keep its contents in the cache.

However, besides optimizing system efficiency, the resource owners usually have some new goals, e.g., maximizing revenues or achieving fairness, in many modern contexts, which require explicitly controllable quality of the caching service. For example, content delivery networks (CDNs) and access ISPs provide in-network caching services for their customers, whose willingness to pay and content characteristics might be different. Under this context, the resource owners' goal is often to maximize their revenues, under which differentiated cache resource is expected to be allocated to the consumers based on their willingness to pay and usage of cache space. For another example, enabled by virtualization technologies, the providers of cloud infrastructures can share resources among multiple paying customers. Under this multi-tenant context, a resource owner's goal might be to guarantee the fairness among its customers. In the allocation of time-shared resources such as network bandwidth, there have been well-defined notions such as max–min fairness and α-fairness, as well as discussions and debates about topics such as net neutrality. Things are not the same for space-shared resources, as the design of existing efficiency-targeted policies did not take fairness into consideration. In this paper, we refer to fairness as equal share of the resource being allocated to consumers if they pay the same amount to the resource owner. To achieve such fairness, the resource owner hopes to explicitly control the portion of resource, i.e., amount of cache space allocated to consumers.

Given the above, what needs to be designed to address these new goals boils down to a new resource management framework that looks beyond hit rate and allows the owner to explicitly allocate its limited resource among competing customers. Such frameworks, e.g., Weight Fair Queueing (WFQ) [1,2], exist for indivisible time-shared resources such as network bandwidth and CPU, but not for space-shared resources.

In this paper, we propose *Weighted Fair Caching* (WFC), a new resource management framework that allows cache owners to achieve various new goals such as providing fairness or controlling resource allocation among contents. WFC is a type of *capacity-driven* policy under which content evictions occur only upon cache misses and thus cache slots are always fully utilized. WFC also belongs to the family of probabilistic Markovian policies that do not require past history and generalize the deterministic cases. However, in contrast to traditional replacement policies, missed contents may not be cached under WFC. Rather than only focusing on hit rate, WFC explicitly controls *occupancy rates* through assigning weights to the individual content items. Because the sum of occupancy rates over all contents equals the cache size, which is often fixed in the short term, cache owners can make various resource allocation decisions by tuning the weights of content items under WFC.

We study the class of probabilistic Markovian policies under the commonly used *Independence Reference Model* (IRM) [3], and characterize the occupancy time of contents that follows a phase-type distribution (Theorem 1), and its relationships with the hit rate and occupancy rate (Theorem 2). Through these general results, we further analyze the properties of WFC and derive following theoretical results.

- We derive the closed-form of steady-state distribution of *Continuous-Time Markov Chain* (CTMC), the mean occupancy time and occupancy rate of content as functions of weights under WFC (Theorem 3).
- We derive the monotonicity and scaling properties (Theorems 4 and 5), and show that the occupancy rate is determined by the normalized weight under WFC (Theorem 6).
- We show that class-based service differentiation can be achieved under WFC (Theorem 7).

We further evaluate the performance of WFC using real data trace from a major video content provider. We find that

- WFC achieves fairer resource allocations among contents compared to traditional cache policies, while sacrificing an acceptable amount of hit rates for fairness.
- WFC provides effective service differentiation among different types of contents.

We believe that WFC provides a novel framework for cache providers to better manage resources and achieve new goals.

2. Related work

System resources can be categorized [4] into *time-shared* resources such as CPU and network bandwidth, and *space-shared* resources such as cache and main memory. In this section, we discuss related resource management schemes for these two types of resources separately.

2.1. Indivisible time-shared resources

Most research of the scheduling of indivisible resources focused on work-conserving mechanisms, under which resource will not be idle with the presence of unfinished workload. Parekh and Gallager [1] studied Generalized Processor Sharing (GPS) in the scope of network bandwidth allocation, while Demers et al. [2] proposed the packet-based version under the name of Weighted Fair Queueing (WFQ). These algorithms achieve advantages over First Come First Serve (FCFS) such as fair bandwidth allocation and shorter delay. Bennett and Zhang [5] further proposed Worst-case Weighted Fair Queueing (WF^2Q) to address the possible discrepancies between service provided under GPS and WFQ. Goyal et al. [6] proposed another fair queueing mechanism called Start-time Fair Queueing (SFQ), which is believed to be better suited for integrated services networks than WFQ. As for CPU scheduling, Stoica et al. [7] proposed a proportional share resource allocation algorithm, which assigns a weight to processes that determines the share of resource to receive. Waldspurger and Weihl proposed lottery [8] and stride [9] scheduling mechanisms. The former is probabilistically fair for *time-shared* resources such as processor time and the latter provides a deterministic alternative. Although plenty of mechanisms exist for explicit resource allocation for indivisible time-shared resources, this is not the case for space-shared ones.

2.2. Divisible space-shared resources

Most of the management schemes for space-shared resources focused on efficiency in terms of hit rate, e.g., LRU, and simplicity in implementation, e.g., FIFO [10], or randomized policies [11]. However, exact analysis for such policies is difficult due to the coupling among contents. Che et al. [12] proposed an approximation that identifies a characteristic time for each cache. Furthermore, *time-driven* policies such as TTL-based caches have been studied. As mentioned in Dehghan et al. [13], TTL-based caching treats contents in a decoupled manner, reducing the theoretical complexity and making exact analysis possible. Fofack et al. [14] and Berger et al. [15] carried out performance evaluation and exact analysis of TTL-based network of caches. Ma and Towsley [16] designed a pricing mechanism to maximize the utility of a single cache. Ferragut et al. [17] also considered optimization of TTL-based cache, while under heavy-tailed demands.

Nevertheless, *time-driven* policies induce a limitation that occupancy time requirements might not always be fulfilled due to the variable demands over a fixed number of cache slots. Also, the cache may not be fully utilized, which bears a resemblance to the non-work-conserving scheduling of *time-shared* resources. In contrast, similar to the work-conserving scheduling, there exist caching policies that evict contents only upon cache misses when the cache is full, which we refer to as *capacity-driven* policies. Deterministic policies such as LRU, LFU and FIFO belong to this class of policies, but only approximate analyses [10,12] are available. On the other hand, Quinones et al. [18] indicated that randomized cache replacement eliminates dependencies on access history, and probabilistic analysis on cache behavior is possible. Moreover, Psounis and Prabhakar [19] showed that LRU policy might not be suitable for web caches, while randomized algorithm can provide approximation without the need to maintain complicated data structures. Waldspurger and Weihl [8] proposed a weighted random replacement policy based on an inverse lottery design. WFC is a capacity-driven, probabilistic and weighted mechanism. We will provide an analysis of both WFC and the seemingly similar *Weighted Random Replacement* (WRR) mechanism and compare their differences in Section 3.

Not many previous works were concerned about the fair sharing of cache. Wang et al. [20] developed a heuristic caching solution that considers both the performance and fairness in the context of information centric networking (ICN). Pu et al. [21] proposed FairRide, a near-optimal solution for fair allocation of memory cache among multiple users with shared files. While these works focused on fairness achieved through probabilistic blocking or expected delay, WFC can be regarded as a general resource allocation framework, under which both fair-sharing and service differentiations can be achieved. Hargreaves et al. [22] built a News Feed model of online social networks inspired by TTL-based cache and studied occupancies of publishers under α-fairness. WFC, on the other side, is a policy designed for general caches. Although the occupancy is also treated as one important metric, WFC achieves different allocations by tuning the probabilities of replacement, rather than deriving the optimal TTLs. The news feed also differs from the actual cache in several aspects, e.g., it is triggered by new posts from publishers, instead of user requests, and it may hold multiple copies from one publisher, instead of at most one.

3. Weighted fair caching mechanism

Without loss of generality, we consider a scenario where there are N available cache slots with equal size and a set $\mathcal{M}$ of content items, each of which can be stored in a cache slot. In reality, contents such as videos may not have the same size; however, they are often divided into data chunks of similar sizes in transit and when being stored. We denote each content item by $i = 1, 2, \ldots, M$. When $M > N$, contents need to compete for the limited cache resources, and we will study the caching policies designed for such scenarios. In particular, we focus on *capacity-driven* policies, under which contents are not evicted until the capacity of the cache is fully utilized and new requests generate cache misses.

3.1. Probabilistic Markovian caching policies

We denote the state of the cache at any time t by $S(t) = s$, which describes the set $s \in \mathcal{M}$ of cached content items. Meanwhile, content items not cached are denoted by $s^c = \mathcal{M} \setminus s$. All the possible states are denoted by $\mathcal{S}$, and all the states that contain content item i are denoted by $\mathcal{S}_i$, i.e., $\mathcal{S} := \{s : s \subset \mathcal{M}, |s| = N\}$ and $\mathcal{S}_i := \{s : i \in s \in \mathcal{S}\} \subset \mathcal{S}$, and $\mathcal{S}_i^c = \mathcal{S} \setminus \mathcal{S}_i$. For any capacity-driven policy, it needs to determine the state of the cache only when a new request of i arrived at time t generates a cache miss, i.e., $i \notin S(t)$. We consider the class of probabilistic Markovian policies defined as follows.

Definition 1. A probabilistic Markovian policy for capacity-driven cache is defined for any cache state s and content $i \notin s$, the probability $P_k(s, i)$ of evicting any content k for all $k \in s$.

The above definition of a caching policy is Markovian because possible cache evictions only depend on the current cache state $S(t) = s$ and the requested content i, but do not rely on any history $S(t')$ for any $t' < t$. This greatly simplifies the implementation of cache policies in practice. Furthermore, probabilistic policies also generalize the deterministic policies, under which one of the probabilities $P_k(s, i)$ is set to be 1. Notice that $\sum_{k \in s} P_k(s, i)$ might be strictly less than 1, which implies that with probability $1 - \sum_{k \in s} P_k(s, i)$, no existing content is evicted and the newly requested content i will not be cached; and therefore, the cache state remains the same.

Content providers often care about the cache performance, which indicates whether it is worth purchasing the caching service. We denote the hit rate of content i by h_i, defined as the percentage of requests of i that result in cache hits. Most prior work on caching focused on the hit rate, while not much has referred to the *occupancy rate*, defined as follows.

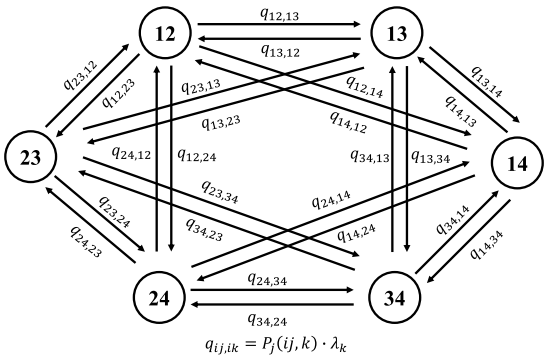


Fig. 1. State transition diagram of the CTMC when $(M, N) = (4, 2)$.

Definition 2. The occupancy rate o_i of any content item i is the percentage of time that i has been cached, defined as

$$o_i = \lim_{T\to+\infty} \frac{\int_0^T 1_{\{i\in S(t)\}} dt}{T}. \tag{1}$$

We want to emphasize that occupancy rate is an equally important performance metric that characterizes the cost of caching from the cache provider's perspective, and can be regarded as the resource allocated to the competing content items. We will evaluate the cache performance in terms of both the hit and occupancy rates of the content items.

We consider the commonly used *Independence Reference Model* (IRM) [3], under which we assume requests for i follow a *Poisson process* with arrival rate λ_i. Later, we will also evaluate results from trace-driven simulations where the arrival process does not strictly follow the IRM. Owing to the memoryless property, we can treat each state of the cache as a state of a *Continuous-Time Markov Chain (CTMC)*, and thus the size of the state space is $|\mathcal{S}| = \binom{M}{N}$. For a CTMC, one state can transit into another with certain transition rate; the transition rate is 0 if two states do not communicate directly. When there is only one different content item between states s and s', for example, $s' \setminus s = \{i\}$ and $s \setminus s' = \{j\}$, then there exists a valid state transition from s to s' if the arrival of i replaces j in the cache. The corresponding state transition rate follows $q_{s,s'} = P_j(s, i)\lambda_i$ such that s' is the next state according to the caching policy determined by $P_j(s, i)$. Fig. 1 shows the state transition diagram of the CTMC of an example of $(M, N) = (4, 2)$. The state transition rates can be written as a transition rate matrix $Q = (q_{s,s'})_{l\times l}$, $l = \binom{M}{N}$, based on which we can derive the *steady-state distribution* $\boldsymbol{\pi} = (\cdots, \pi_s, \ldots)$ that satisfies $\boldsymbol{\pi}Q = \mathbf{0}$ and $\boldsymbol{\pi}\mathbf{1} = 1$. π_s denotes the probability that the cache is in state s in a steady-state after a long-run, which can also be interpreted as the percentage of time that the cache is in state s. To derive the occupancy rate o_i, we are also concerned about the behavior of an individual content item i each time it gets cached, especially the *occupancy time* T_i defined as the random amount of time that i is cached until it gets evicted upon a cache replacement.

Theorem 1 (*Occupancy Time*)**.** *T_i follows a phase-type distribution $PH(\boldsymbol{\alpha}_i, Q_i)$, where Q_i is the sub-matrix of Q that only includes the transition rates between the states in $\mathcal{S}_i$ and $\boldsymbol{\alpha}_i = (\ldots, \alpha_s, \ldots)$ is the initial distribution vector that follows*

$$\alpha_s = \frac{\sum_{r\in\mathcal{S}_i^c} \pi_r P_{r\setminus s}(r, i)}{\sum_{r'\in\mathcal{S}_i^c} \pi_{r'}}, \quad \forall s \in \mathcal{S}_i. \tag{2}$$

The average occupancy time satisfies

$$\mathbb{E}[T_i] = \frac{1}{\lambda_i} \frac{\sum_{s\in\mathcal{S}_i} \pi_s}{\sum_{r\in\mathcal{S}_i^c} \pi_r}. \tag{3}$$

Theorem 1[2] shows that the occupancy time can be characterized by the *phase-type* distribution [23]. Given the initial distribution vector $\boldsymbol{\alpha}_i$ where α_s stands for the probability that the initial state is $s \in \mathcal{S}_i$ when i gets cached, and Q_i is the state transition matrix between the states in $\mathcal{S}_i$, the *phase-type* distribution $PH(\boldsymbol{\alpha}_i, Q_i)$ describes the distribution of time until entering the absorbing state, which in our case refers to when i gets replaced from the cache and the CTMC transits to some state $r \in \mathcal{S}_i^c$. Since there should be only one absorbing state, for content item i, if we treat all $r \in \mathcal{S}_i^c$ as an aggregated state S_0, such that any state transition to any state r is considered as a state transition towards S_0, then S_0 is the absorbing state. Therefore, when i gets cached, the process enters the initial state; and when i gets evicted, the process enters the absorption state. By using the average occupancy time of Theorem 1, we can further connect the occupancy rate, hit rate and steady-state distribution as follows.

[2] Proofs of theorems can be found in Appendix A.

Theorem 2 (*Occupancy Rate and Hit Rate*)**.** *Under IRM, the hit rate h_i and occupancy o_i of any content item i satisfy*

$$h_i = \frac{\lambda_i \mathbb{E}[T_i]}{\lambda_i \mathbb{E}[T_i] + 1} = o_i = \sum_{s \in \mathcal{S}_i} \pi_s, \quad \forall i \in \mathcal{M}. \tag{4}$$

Theorem 2 shows that the hit rate and occupancy rate of any content item are the same under the IRM, which is due to the fundamental property of Poisson Arrivals Sees Time Averages (PASTA). Although we will primarily focus on the design of caching policies that allocate resource among contents in terms of their occupancy rates, the hit rates are positively correlated with the occupancy rates in general.

3.2. Weighted caching policies

An intuitive subclass of the probabilistic Markovian policies that allows us to explicitly differentiate contents is *weighted policies*, under which each content i is assigned a non-negative weight w_i and the probabilities $P_k(s, i)$ become a function of the weights w_i and w_ks for all $k \in s$. If a weighted policy always guarantees that missed contents are cached, it leads to a *replacement policy* with the simplest form of a *Weighted Random Replacement* (WRR) [8] policy defined as

$$P_k(s, i) = \frac{w_k}{W_s}, \quad \forall\, k \in s, \text{ where } W_s = \sum_{j \in s} w_j. \tag{5}$$

In this work, we propose a new policy *Weighted Fair Caching* (WFC) that does not always cache missed contents, defined as

$$P_k(s, i) = \frac{w_k}{W_s + w_i}, \ \forall k \in s. \tag{6}$$

The difference of the above two lies in that under WFC, the requested content item i is also involved in the decision of cache replacement. In particular, content i might not be cached with probability $w_i/(W_s + w_i)$ under WFC, proportional to all the eviction probabilities of content items existing in the cache. Regardless of whether the arrival process strictly follows the IRM assumption, WFC is implemented by simply associating content items with weights which determine the probability of being replaced or not cached. Nonetheless, the closed-form results derived from Markovian analysis may not still follow without the IRM assumption.

Theorem 3 (*Occupancy Metrics under Weighted Policies*)**.** *The steady-state distributions of the CTMC under WRR and WFC, respectively, satisfy*

$$\pi_s^{RR} = \frac{W_s \phi_s}{\sum_{r \in \mathcal{S}} W_r \phi_r} \quad \text{and} \quad \pi_s^{FC} = \frac{\phi_s}{\sum_{r \in \mathcal{S}} \phi_r},$$

where $W_s = \sum_{j \in s} w_j$ and $\phi_s = \left(\prod_{j \in s} \lambda_j\right)\left(\prod_{j \in s^c} w_j\right)$. The expected occupancy times under WRR and WFC satisfy

$$\mathbb{E}[T_i^{RR}] = \frac{1}{\lambda_i} \frac{\sum_{s \in \mathcal{S}_i} W_s \phi_s}{\sum_{r \in \mathcal{S}_i^c} W_r \phi_r}, \quad \mathbb{E}[T_i^{FC}] = \frac{1}{\lambda_i} \frac{\sum_{s \in \mathcal{S}_i} \phi_s}{\sum_{r \in \mathcal{S}_i^c} \phi_r}.$$

The occupancy rates under WRR and WFC satisfy

$$o_i^{RR} = \frac{\sum_{s \in \mathcal{S}_i} W_s \phi_s}{\sum_{r \in \mathcal{S}} W_r \phi_r} \quad \text{and} \quad o_i^{FC} = \frac{\sum_{s \in \mathcal{S}_i} \phi_s}{\sum_{r \in \mathcal{S}} \phi_r}. \tag{7}$$

Theorem 3 derives the closed-form of occupancy time and occupancy rate of any content item $i \in \mathcal{M}$ as a function of the arrival rates and weights of all contents. Notice that if all the weights are strictly positive, i.e., $w_i > 0$ for all $i \in \mathcal{M}$, the above results can be rescaled by dividing both the numerators and denominators by $\tilde{W} = \prod_{i=1}^{M} w_i$, and therefore, ϕ_s could be replaced by the more intuitive notation φ_s defined as

$$\varphi_s = \phi_s / \tilde{W} = \prod_{i \in s} \lambda_i / w_i,$$

which only depends on the arrival rates and weights of contents that are cached under state s. The above results also generalize the case of any content i permanently cached when w_i is set to 0 under WFC, because we will have $\phi_s = 0$ for all $s \notin \mathcal{S}_i$, and therefore, its occupancy rate satisfies $o_i^{FC} = 1$. To compare WRR and WFC, we consider the case of $N = 1$ as follows.

Corollary 1. *Consider the case of $N = 1$. According to Theorem 3, the occupancy rates under WRR and WFC are*

$$o_i^{RR} = \frac{\lambda_i}{\sum_{j=1}^{M} \lambda_j} \quad \text{and} \quad o_i^{FC} = \frac{\lambda_i \prod_{k \neq i} w_k}{\sum_{j=1}^{M} \lambda_j \prod_{k \neq j} w_k}.$$

Corollary 1 shows that the occupancy rate o_i^{RR} under WRR only depends on the arrival rates λ_js, but not the weights w_js. This implies that weights cannot be used to control the occupancy rates across different contents. The fundamental reason is that any missed item is cached for sure, and therefore, a content item's arrival rate naturally provides a lower bound of its occupancy rate, regardless of the weight associated with it. If we focus on the non-trivial cases where $w_i > 0$ for all $i \in \mathcal{M}$ under WFC, the occupancy rate can be simplified as

$$o_i^{FC} = \frac{\varphi_i}{\sum_{j=1}^{M} \varphi_j} = \frac{\lambda_i / w_i}{\sum_{j=1}^{M} \lambda_j / w_j}.$$

From the above equation, we observe that the weights could be used to achieve any distribution of occupancy rates. In particular, any occupancy rate vector $\mathbf{o} = (o_1, \ldots, o_M)$ that satisfies $\sum_{i=1}^{M} o_i = 1$ and $o_i > 0$ for all $i \in \mathcal{M}$ can be achieved if each weight w_i is set to be proportional to λ_i / o_i. Through this comparison, we observe that WRR is somehow limited if one wants to use weights to control the resource allocation among the contents in terms of occupancy rate, while WFC provides more flexibility for that purpose. Thus, in the following sections, we will focus on WFC (and omit this superscript in the notation) and study how the weights can be used to allocate cache resources in terms of the occupancy rates for content items.

4. Resource allocation under WFC

In the previous section, we derived the occupancy rates of contents under WFC via an analysis of the underlying CTMC and showed that weights w_is can be used to affect the cache performance for contents in terms of their occupancy rates o_is, which equal the corresponding hit rates h_is under IRM. Because *capacity-driven* policies always keep all N cache slots occupied at any time, by Definition 2, the occupancy rates satisfy $\sum_{i \in \mathcal{M}} o_i = N$, which holds even without the IRM assumption. Consequently, we can regard the result of any WFC policy as a resource allocation solution $\mathbf{o} \in \mathcal{O}$, where the domain of resource allocation is defined as

$$\mathcal{O} = \{\mathbf{o} : \sum_{i \in \mathcal{M}} o_i = N \quad \text{and} \quad o_i > 0 \; \forall i \in \mathcal{M}\}.$$

From a cache provider or CDN's point of view, limited cache space is valuable resources that can be monetized, and the design of caching policies should be driven by business models. Since the occupancy rate of a content item can be regarded as its incurred cost to the cache provider, and suppose the provider of content i is charged an amount $R_i(o_i)$ based on the cost o_i, the cache provider generally wants to find an optimal allocation $\mathbf{o}^* \in \mathcal{O}$ that maximizes the aggregate revenue $\sum_{i \in \mathcal{M}} R_i(o_i)$. In this regard, WFC provides a useful mechanism for cache providers to realize various resource allocations by tuning the weights of the contents, shown as the following results.

Theorem 4 (*Monotonicity of WFC*)**.** *For any content $i \in \mathcal{M}$,*

$$\frac{\partial o_i}{\partial w_i} < 0 \quad \text{and} \quad \frac{\partial o_i}{\partial w_j} > 0, \quad \forall j \neq i. \tag{8}$$

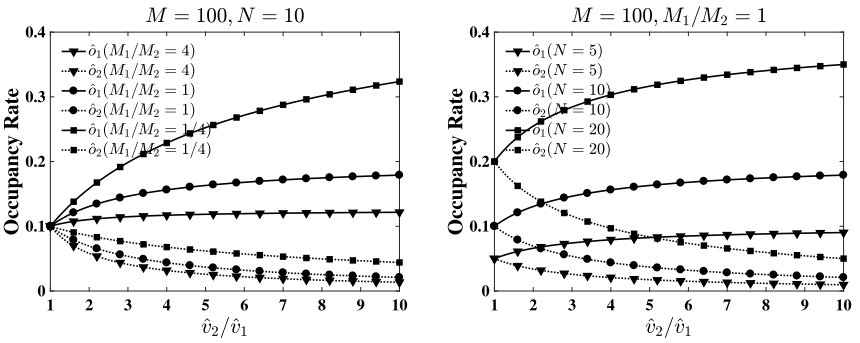
Theorem 4 intuitively states that when the weight of any content i increases unilaterally, the occupancy rate of i decreases, while that of all other contents increases. This implies that by adjusting the weights, one can steer the distribution of occupancy rates from some contents to others.

As an important desirable property of resource allocation, fairness is often a concern when resources are shared among competing customers that pay the same, and achieving equal occupancy rates for them are required. This goal cannot be fulfilled by traditional replacement policies such as LRU and LFU that focus on the aggregate hit rate over all contents. The following result shows that weights can be set easily to achieve a fair sharing of cache among all contents under WFC.

Theorem 5 (*Scaling of WFC*)**.** *Suppose $\mathbf{o} \in \mathcal{O}$ is achieved for arrival rates $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_M)$ by using the weights $\mathbf{w} = (w_1, \ldots, w_M)$ under WFC. Under any new arrival rates $\boldsymbol{\lambda}' = (\lambda_1', \ldots, \lambda_M')$, the same occupancy rates of the contents can be achieved if the new weights $\mathbf{w}' = (w_1', \ldots, w_M')$ satisfy*

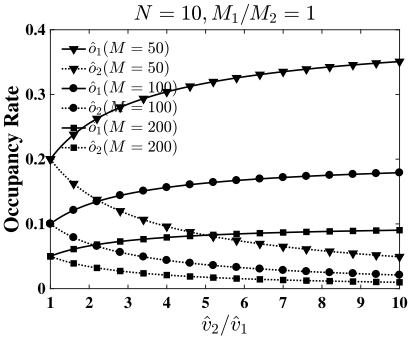
$$w_i' = \frac{\lambda_i'}{\lambda_i} w_i, \quad \forall i \in \mathcal{M}. \tag{9}$$

Theorem 5 shows that in order to maintain the same allocation of occupancy rates for contents when their arrival rates change, one only needs to rescale each content i's weight such that the ratio of λ_i and w_i is kept a constant. Because with equal weights, contents will obtain the same occupancy rate when their arrival rates are the same, Theorem 5 further implies that a fair allocation among contents can be achieved by setting the weights proportional to the corresponding arrival rates. Consequently, we can decouple the impact of arrival rates on the resulting occupancy rates by focusing on the normalized weight $v_i = w_i / \lambda_i$ without loss of generality. Notice that the fair allocation of equal occupancy rates is achieved when the normalized weights of contents are equal.



(a) When M and N are fixed and M_1/M_2 is varying.

(b) When M and M_1/M_2 are fixed and N is varying.



(c) When N and M_1/M_2 are fixed and M is varying.

Fig. 2. Occupancy rates of contents belonging to 2 classes under different M, N or M_1/M_2, when $\hat{v}_2/\hat{v}_1$ ratio is varying.

Theorem 6 (*Impact of Normalized Weight on Occupancy*)**.** *For any content $i \in \mathcal{M}$, its occupancy rate o_i satisfies*

$$o_i = \frac{\bar{v}_{-i}}{v_i + \bar{v}_{-i}} \tag{10}$$

where $\bar{v}_{-i}$ is a function of the normalized weights v_js of all content items other than i, defined as

$$\bar{v}_{-i} = \frac{\sum_{s \in \mathcal{S}_i} \prod_{j \in s \setminus \{i\}} v_j^{-1}}{\sum_{r \in \mathcal{S}_i^c} \prod_{k \in r} v_k^{-1}} = \frac{\sum_{s \in \mathcal{S}_i} \prod_{j \in s^c \cup \{i\}} v_j}{\sum_{r \in \mathcal{S}_i^c} \prod_{k \in r^c} v_k}. \tag{11}$$

Theorem 6 establishes a relationship between any content i's occupancy rate o_i and its normalized weight v_i, through an aggregate information $\bar{v}_{-i}$ of all other contents. In general, $\bar{v}_{-i}$ can be regarded as a form of weighted average of the normalized weights v_js of the competing contents. As an illustration of $(M, N) = (4, 2)$, we have $\bar{v}_{-1} = (v_2v_3 + v_2v_4 + v_3v_4)/(v_2 + v_3 + v_4)$. To achieve any target occupancy rate o_i, Theorem 6 implies that one can simply set the weight to be $v_i = \bar{v}_{-i}(1 - o_i)/o_i$.

By far, we know that equal normalized weights induce a fair resource allocation, while varying them leads to fine-grained distributions of occupancy rates among contents. In practice, contents are often classified into groups and served differently. This could be due to the nature of contents, e.g., inelastic contents that are sensitive to delays vs. elastic contents that are less sensitive to delay, or different types of content providers that have different willingness to pay for caching services. It is desirable to be able to prioritize certain contents with higher occupancy rates so as to reduce the retrieval time for them. By utilizing WFC, we can create differentiated service classes such that the occupancy rates of contents within any class are the same, while those across classes can be differentiated. In general, we consider K service classes. We denote the number of content items served under any class k by M_k and define $M = \sum_{k=1}^{K} M_k$. To achieve the same occupancy rate within any class k, we ensure that the normalized weights of contents equal a constant, denoted by $\hat{v}_k$. In other words, the weight w_i is set to be $\lambda_i \hat{v}_k$ for any content i served under class k. We denoted the occupancy rate of contents under class k by $\hat{o}_k$ and derive it as a function of the weights $\hat{v}_k$s as follows.

Theorem 7 (*Class-Based Allocation under WFC*). *Let* $\mathbf{n} = (n_1, \ldots, n_K)$ *denote a partition of cache space where each* n_k *denotes the number of cache slots occupied by content items from class k, and define two domains of such partitions as* $\mathcal{N} = \{\mathbf{n} : \sum_{k=1}^{K} n_k = N,\ 0 \le n_k \le M_k,\ \forall k = 1, \ldots, K\}$ *and* $\mathcal{N}_k = \{\mathbf{n} \in \mathcal{N} : n_k \neq 0\}$. *The occupancy rate* $\hat{o}_k$ *for contents in class k satisfies* $\hat{o}_k = O_k/O$, *where*

$$O_k = \frac{1}{M_k} \sum_{\mathbf{n} \in \mathcal{N}_k} n_k \frac{\prod_{l=1}^{K} \binom{M_l}{n_l}}{\prod_{l=1}^{K} \hat{v}_l^{n_l}} \quad \text{and} \quad O = \sum_{\mathbf{n} \in \mathcal{N}} \frac{\prod_{l=1}^{K} \binom{M_l}{n_l}}{\prod_{l=1}^{K} \hat{v}_l^{n_l}}.$$

Theorem 7 shows that $\hat{v}_k$s play the role of weights in the denominators for each combinatorial term $\prod_{l=1}^{K} \binom{M_l}{n_l}$ that corresponds to a partition $\mathbf{n}$ of cache for the service classes. To further illustrate the impact of various parameters such as the cache size N, aggregate content size M, the class size M_k and the normalized weight of class $\hat{v}_k$ on the class-based resource allocation, we consider a two-class scenario, i.e., $K = 2$, and plot the occupancy rates $\hat{o}_1$ and $\hat{o}_2$ under different settings in Fig. 2. We assume that class 1 is a prioritized class and therefore, we increase the ratio $\hat{v}_2/\hat{v}_1$ from 1 along the x-axis. In all three subfigures, we observe that fair allocations, i.e., $\hat{o}_1 = \hat{o}_2 = N/M$, are achieved between the two service classes when $\hat{v}_2/\hat{v}_1 = 1$, and $\hat{o}_1$ increases with the ratio $\hat{v}_2/\hat{v}_1$ while $\hat{o}_2$ decreases, which can be inferred from Theorem 4. In Fig. 2(a), we vary the ratio of M_1/M_2 under a fixed total size M and observe that both $\hat{o}_1$ and $\hat{o}_2$ decrease with the ratio M_1/M_2. Intuitively, when $\hat{v}_2 > \hat{v}_1$, $\hat{o}_2 < \hat{o}_1$ and decreasing the ratio of M_1/M_2 must increase $\hat{o}_1$, because pushing more contents to the lower-quality class will improve the occupancy rate of the higher class. Furthermore, as $M_1\hat{o}_1 + M_2\hat{o}_2 = N$ always holds, the corresponding occupancy rate $\hat{o}_2$ of the lower class also increases. Under a fixed ratio of $M_1/M_2 = 1$, we vary the cache size N and content size M in Figs. 2(b) and 2(c), respectively. We observe that both $\hat{o}_1$ and $\hat{o}_2$ increase with N but decrease with M, because a larger cache provides higher occupancy rates while a larger number of contents increases the competition and reduces the occupancy rate of any individual content. In conclusion, in order to achieve a higher occupancy rate $\hat{o}_k$ for a service class k, it is possible to reduce the ratio of the normalized weight, i.e., $\hat{v}_k/\sum_{l=1}^{K} \hat{v}_l$, since the chance of contents from class k being replaced is reduced, or decrease the class size M_k if it is a high-priority class. Although content size M and cache size N are usually fixed, their impacts on the occupancy rates are also understood intuitively.

5. Trace-Driven evaluation

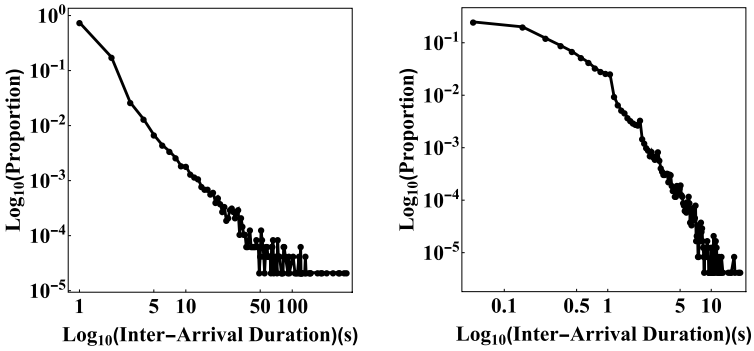
In the previous sections, we provide theoretical properties of WFC under the IRM assumption. In this section, we further evaluate the performance of WFC using trace data from two different major video content providers. We use the word *video* and *content* interchangeably in this section. In particular, we first explain the raw datasets and the data cleaning process, and show that the characteristics of the dataset deviates from the IRM assumption substantially. Our implementation of WFC depends solely on the different weights assigned to content items, regardless of whether the IRM assumption is strictly satisfied. We then proceed to (1) evaluate the fairness of WFC by comparing it with traditional capacity-drive policies such as LRU and FIFO, and (2) demonstrate the class-based service differentiation enabled by WFC.

5.1. Dataset

Our datasets came from two major Chinese video streaming service providers. Dataset A was collected throughout three entire days, i.e., Dec. 21st to 23rd 2017, on three servers, which means there are nine trace files in total. Each of these trace files is around 1GB in size, and each record in the data trace consists of five fields, namely the URI of the resource, the timestamp of the request, the ISP, the requested resource size and where the requested resource was found. We take the trace collected from server 1 on Dec. 21st for evaluation, as these trace files reflect similar request patterns. Since each request for any resource on the server has been recorded, we first remove those not for video resources, such as requests for .xml and .apk files. We then filter out invalid records, such as those with an empty URI, and irrelevant fields such as the ISP and where the resource was found.

Dataset B provided dozens of different types of video contents such as movies, news clips and TV series. The raw data was collected throughout November 2014 with a size around 780GB. As the total number of views from users is in the order of 10^8 per day, we take the data trace of a typical day, i.e., Nov. 7th, for our evaluations. Each data record in the trace contains multiple fields of information such as user's ID, IP address, viewing duration and departure time, and video's ID and type. We first filter out invalid records that have 0 viewing duration or invalid video IDs. We then calculate the *request time* of contents from users by subtracting their departure times with their corresponding viewing durations. We also remove irrelevant fields in the original dataset, such as IP address and ISP name. The data records in the cleaned dataset used in our evaluations are composed of the request time, video ID, and video type per record, and are sorted by the request times.

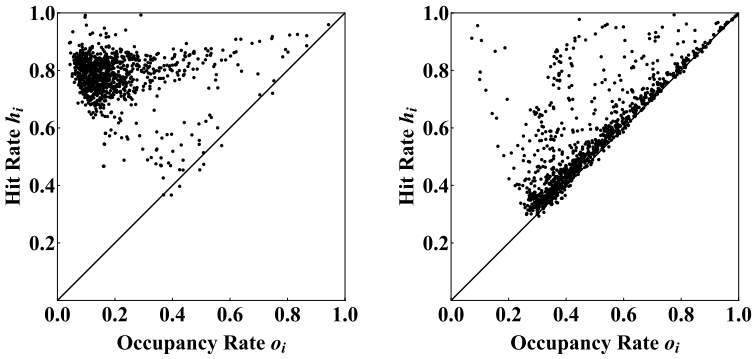
To take a quick glance at the datasets, we plot the distribution of inter-request durations for the most popular video from them in Fig. 3. Popularity is measured by the total number of times that a video is requested during a day, and we sort the videos according to their popularity. Since the x-axis and y-axis are both shown in a log-scale, we observe that the distribution deviates largely from an exponential distribution, for either one of the datasets, which should be expected if the IRM assumption holds for the data. Furthermore, requests for the most popular video from Dataset B arrive more frequently, as a larger portion of inter-arrival duration appear in a smaller interval. We can also gain an insight into the distribution of popularity of the videos. For Dataset A, the popularity distribution of the videos can be fitted by a Zipf distribution $Zipf(s, N)$



(a) The most popular video from Dataset A.

(b) The most popular video from Dataset B.

Fig. 3. Proportion of inter-arrival durations over different intervals.



(a) The most popular 1000 videos from Dataset A.

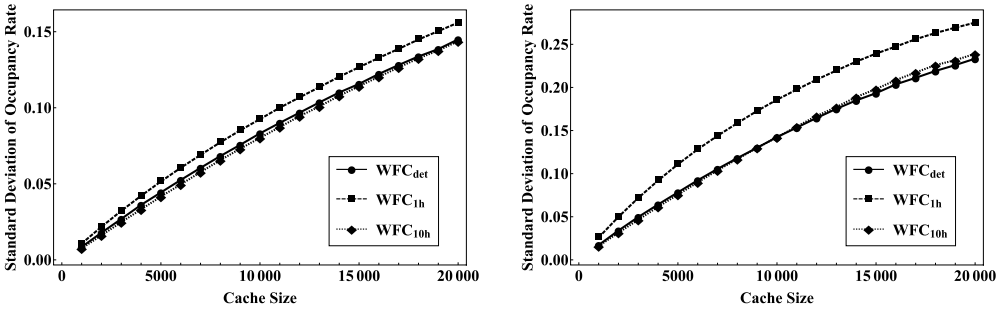
(b) The most popular 1000 videos from Dataset B.

Fig. 4. Hit rates vs. occupancy rates for the 1000 most popular videos under WFC policy. Cache size is N = 1000.

such that out of N videos, the frequency of the video of popularity rank k is given by $f(k; s, N) = k^{-s}/\sum_{n=1}^{N} n^{-s}$, where $N = 47010$, $s = 0.336925$. For Dataset B, we can similarly model the popularity distribution by *Zipf*(0.619837, 234392).

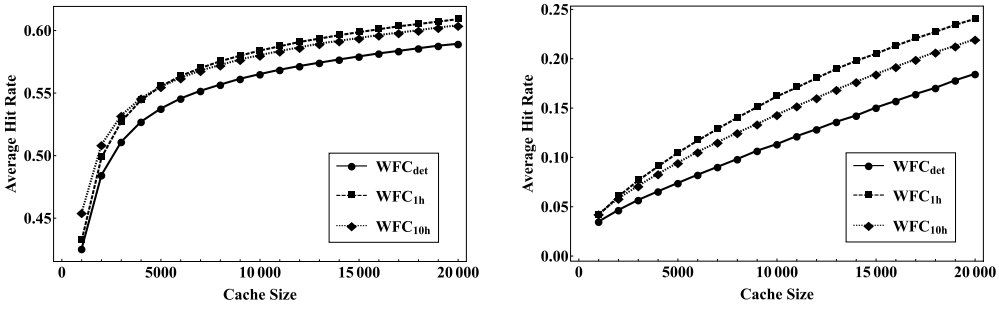
We also implemented WFC with $w_i = 1$ for all contents with a cache of size $N = 1000$ and plot the occupancy rates vs. the corresponding hit rates for the 1000 most popular content items in the dataset in Fig. 4. The diagonal line represents the points of $h_i = o_i$. According to Theorem 2, the hit rates should be equal to the occupancy rates if the requests follow the IRM assumption. However, we observe that for both of the datasets, the hit rates are consistently greater than the occupancy rates, which again indicates that the IRM assumption does not hold for our datasets.

Despite the deviation from our theoretical assumption, we can still evaluate the WFC policy with regard to fairness and class-based resource allocation as studied in the previous section. We use both of the datasets in the evaluation of fairness. They may differ in regard to the actual request arrival distribution, and certain format of the records, but we apply the same cache simulation program to the trace files from them. The cleaned trace file from Dataset A contains 7291823 records of request arrivals, with 177 621 different videos involved in total. Dataset B is also used in the evaluation of class-based resource allocation, since it explicitly provides information on the types of videos which Dataset A does not. In particular, we choose two most representative types of videos, i.e., movies and news clips, for the rest evaluations. Other video types are either similar to the two chosen types, or are not popular enough to provide sufficient data for separate evaluations. The characteristics of and demands for movies and new clips are quite different, which often need to be treated in different service classes if service differentiation is possible. For example, a movie is usually hours long and can remain continuously popular for days or even weeks, while a news clip only lasts for several minutes and is quite time sensitive that can only gain attraction from the majority for a few hours. On a daily basis, users request for these types of videos for millions of times, and the number of distinct content items are in the order of $10^4 \sim 10^5$ for each type. In total, we have M = 60 506 videos of movie and news clips viewed by users for 3944832 times. In the case of class-based service differentiation, we label news clip as class 1 content and movie as class 2 content, whose sizes are $M_1 = 42587$ and $M_2 = 17919$.



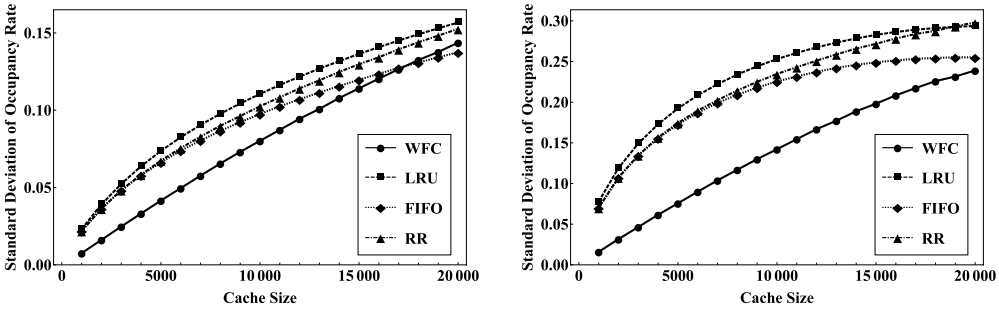
(a) Dataset A. (b) Dataset B.

Fig. 5. Standard deviation of occupancy rates of videos under various cache sizes, using WFC with pre-determined weights or weights estimated by a sliding window.



(a) Dataset A. (b) Dataset B.

Fig. 6. Average hit rates of videos under various cache sizes, using WFC with pre-determined weights or weights approximated by a sliding window.



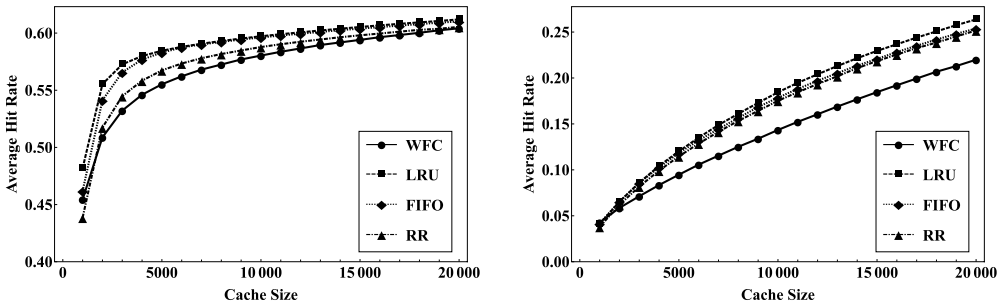
(a) Dataset A. (b) Dataset B.

Fig. 7. Standard deviation of occupancy rates of videos under various cache sizes and different caching policies.

5.2. Evaluation of fairness

We first evaluate the fairness property of WFC. As we know from Theorem 5, equal occupancy rates can be guaranteed if the weights w_i are set proportional to the request rates λ_i, or equivalently equalizing the normalized weights v_i for all videos. In the previous steps, we have already pre-processed the data trace and have obtained the popularity of each video. Hence, we can set a determined weight for each video, as the average arrival rate of request for a video is implied by its popularity. We denote WFC implemented under the pre-determined weights by WFC$_{\text{det}}$. However, λ_i is not known in advance in practice. In our implementation of WFC, we can dynamically update and keep track of λ_i by counting the number of arrivals within a sliding time window. Since a cache user could require a trade-off between the accuracy of estimation and the cost of maintaining the sliding window, we test two different lengths of the sliding window, 1 h and 10 h. We denote these two versions of WFC by WFC$_{1\text{h}}$ and WFC$_{10\text{h}}$ respectively.

We first compare these three different versions of WFC implementations, to see the influence of estimating λ_i by a sliding window on the performance of WFC. Because capacity-driven policies always utilize all cache slots, the average occupancy



(a) Dataset A. (b) Dataset B.

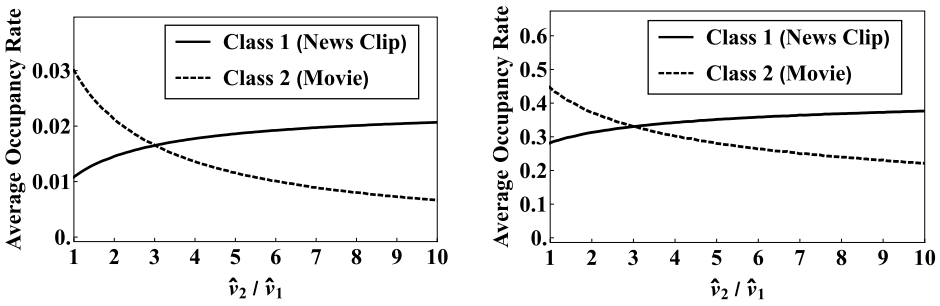
Fig. 8. Average hit rates of videos under various cache sizes and different caching policies.

rate across all contents always equals N/M. Thus, to compare the fairness across different situations, we plot the standard deviation of the occupancy rates over various cache sizes, i.e., from $N = 1000$ to 20 000, with a step size of 1000 in Fig. 5, which provides hints on how fair a group of values are with only one value. We can notice that $\text{WFC}_{1\text{h}}$ has the highest value, either for Dataset A or Dataset B, indicating that λ_i estimated by such a short sliding window may not be accurate enough. On the other hand, the value of $\text{WFC}_{10\text{h}}$ is even slightly lower than WFC_{det}, for Dataset A under all measured cache sizes and for Dataset B when $N \leq 10000$, probably because the instantaneous request arrival rates approximated by the former are more accurate than the long-term average arrival rates used in the latter for some moments if the actual request arrival process is not identical throughout the day.

As we observed from Fig. 4 that hit rates do not equal occupancy rates in general when the IRM assumption is not satisfied, we plot the average hit rates of videos in Fig. 6. $\text{WFC}_{1\text{h}}$ shows the highest values, followed by $\text{WFC}_{10\text{h}}$, for Dataset A when $N \geq 5000$ and for Dataset B when $N \geq 2000$. WFC_{det} has noticeably lower average hit rates than the other two when the cache size N becomes larger. By the initial design of our WFC policy, we understand it may sacrifice a bit of the hit rate compared to some other policies that focus on the overall hit rate, to achieve fairness. Hence, as $\text{WFC}_{1\text{h}}$ does not perform as well as $\text{WFC}_{10\text{h}}$ and WFC_{det} in terms of fairness, it on the other side outperforms them in hit rates. In the rest of this section, we take $\text{WFC}_{10\text{h}}$ for further comparison of the performance, and refer to it as WFC for simplicity.

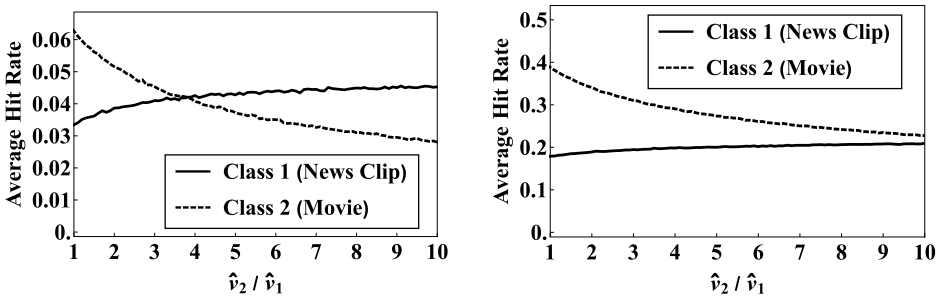
We then compare WFC with other well-known caching policies, such as LRU, FIFO or random replacement (RR), to see if WFC performs the best on fairness. Therefore, we have also implemented another three caches based on LRU, FIFO and RR, and compare them also on the standard deviation of occupancy rates and average hit rates. Similar to Fig. 5, we observe from Fig. 7 that the standard deviation of occupancy rates is increasing over N under all the four caching policies. LRU generally shows the largest deviation in occupancy rates, as it aims to improve the chance of hit for popular contents, thus more popular contents tend to achieve higher hit rates and occupancy rates; FIFO and RR share almost the same values when the cache size is small, e.g., $N \leq 5000$, appearing slightly smaller than those under LRU, as they both set some rule equally for each content to stay in cache, instead of preference for more popular ones. When the cache becomes larger, FIFO performs better than RR, as FIFO allows any content to stay in the cache for a same amount of time once getting cached, which is less influenced by the arrival rates of different contents. RR, on the other side, even shows the highest standard deviation for Dataset B when $N = 20\,000$. Although WFC always has the lowest value of all when the cache size N is relatively smaller, for Dataset A, it is overtaken by other policies such as FIFO when N increases, e.g., $N > 17\,000$. For Dataset B, WFC always shows the smallest value, but the difference between WFC and FIFO also shrinks when the cache is large. Not to mention that in reality, cache size N usually does not exceed $1/10$ of the content size M which is 177 621 for Dataset A and 60 506 for Dataset B, considering the expensive price of cache, but we can also understand the occupancy rates from another perspective. Instead of focusing solely on a single value for each execution of the cache simulation, we also plot groups of individual occupancy rates in Appendix B. We observe from Figs. B.11 and B.12 that WFC is the only one of the compared policies that attempts to achieve fairness, even when the standard deviation of occupancy rates is not as low as that of other policies.

Finally, we also plot the average hit rates of contents under the four caching policies in Fig. 8. Similar to our observation from Fig. 4, the average hit rates under all caching policies do not match the theoretical value under the IRM assumption that equals the average occupancy rate N/M. It is no surprise that LRU performs the best in terms of average hit rate, as it aims to generate more cache hits. FIFO performs slightly worse than LRU, especially under a larger cache size. Our WFC policy, on the other side, sacrifices a bit in hit rate, in order to achieve fairness in terms of occupancy rate which is more concentrated on. Nonetheless, WFC usually shows no more than 0.05 in difference compared to the average hit rate under LRU, and even outperforms LRU slightly for Dataset B when $N = 1000$, but it is the only policy that brings fairness into the allocation of cache.



(a) Cache size $N = 1000$. (b) Cache size $N = 20000$.

Fig. 9. Average occupancy rate of contents belonging to 2 classes when $\hat{v}_2/\hat{v}_1$ ratio is varying for cache size $N = 1000$ and 20 000.



(a) Cache size $N = 1000$. (b) Cache size $N = 20000$.

Fig. 10. Average hit rate of contents belonging to 2 classes when $\hat{v}_2/\hat{v}_1$ ratio is varying for cache size $N = 1000$ and 20 000.

5.3. Evaluation of class-based resource allocation

After evaluating the fairness property of WFC, we evaluate the use of WFC for class-based service differentiation. Based on Theorem 7, we can set a normalized weight $\hat{v}_j$ for each class j and adjust the ratio $\hat{v}_2/\hat{v}_1$ to influence the class-based resource allocation $\hat{o}_1$ and $\hat{o}_2$.

We vary the ratio $\hat{v}_2/\hat{v}_1$ along x-axis and plot the average occupancy rates of the two classes of videos, i.e., $\bar{o}_1$ for news clips and $\bar{o}_2$ for movies, in Fig. 9. We observe that the average occupancy rates show similar trends as $\hat{o}_1$ and $\hat{o}_2$ observed from Fig. 2: (1) when $\hat{v}_2/\hat{v}_1$ increases, $\hat{o}_2$ drops while $\hat{o}_1$ increases, and (2) both $\hat{o}_1$ and $\hat{o}_2$ increase with the cache size N, which matches our intuition and previous analysis. Unlike the theoretical result from Theorem 7 that the fair allocation is achieved at $\hat{v}_2/\hat{v}_1 = 1$ and $\hat{o}_1 = \hat{o}_2 = N/M$, we observe that the fair allocation is achieved when $\hat{v}_2/\hat{v}_1$ is around 3, due to the discrepancies between actual arrival process of the requests and the IRM assumption.

In addition, we plot the corresponding average hit rates in Fig. 10. Similar trends can be observed for the average hit rates, although the curves are not that smooth compared to those of the occupancy rates in Fig. 9. We believe that the average hit rates fluctuate possibly due to the randomness brought by the probabilistic replacement decisions, while the average occupancy rates are more strongly correlated with the ratio $\hat{v}_2/\hat{v}_1$, and are always constrained by $M_1\hat{o}_1 + M_2\hat{o}_2 = N$. Moreover, the average hit rates also differ from the occupancy rates. When the cache size is set to be a larger value, i.e., 20 000, the average hit rates even become smaller than the average occupancy rates, and the point where $\hat{h}_2 = \hat{h}_1$ seems to be somewhere $\hat{v}_2/\hat{v}_1 > 10$.

6. Conclusion

We have proposed WFC, a capacity-driven caching policy that allows tunable resource allocation through weights assigned to contents. WFC provides an occupancy-centric framework that allows cache owners to make flexible resource allocations decision to achieve various objectives. We derived multiple properties of WFC as well as closed-form results, and showed that fairness and class-based differentiation could be achieved under WFC via trace-driven evaluation, where the IRM assumption does not hold. Meanwhile, WFC is able to trade off an acceptable amount of performance, in terms of hit rate, to realize fairness.

Appendix A. Proof of theorems

A.1. Proof of Theorem 1

Proof. Each element α_s of $\boldsymbol{\alpha}_i$ represents the probability that when i gets cached, state of the CTMC transits from any $r \in \mathcal{S}_i^c$ to $s \in \mathcal{S}_i$. The probability that the previous state is any one of such r is $\pi_r / \sum_{r' \in \mathcal{S}_i^c} \pi_r'$ and the probability of state transition from r to s is $P_{r \setminus s}(r, i)$ if $s \setminus r = i$. Therefore, α_s is the aggregate probability that state transition occurs from any state in $\mathcal{S}_i^c$ to s, that is,

$$\alpha_s = \frac{\sum_{r \in \mathcal{S}_i^c} \pi_r P_{r \setminus s}(r, i)}{\sum_{r' \in \mathcal{S}_i^c} \pi_{r'}}, \ \forall s \in \mathcal{S}_i.$$

Assume $s_j \in \mathcal{S}_i, j = 1, \ldots, k$ and $s_j \in \mathcal{S}_i^c, j = k+1, \ldots, l$. Denote $\boldsymbol{\pi}_{\mathcal{S}_i} = (\pi_{s_1}, \ldots, \pi_{s_k})$, $\boldsymbol{\pi}_{\mathcal{S}_i^c} = (\pi_{s_{k+1}}, \ldots, \pi_{s_l})$. Assume Q is also sorted in the same manner and denote sub-matrices $Q_i = Q[1, \ldots, k; 1, \ldots, k]$, $R_i = Q[k+1, \ldots, l; 1, \ldots, k]$. Q_i is the transition rate matrix between states in $\mathcal{S}_i$ that is applied to calculate the absorbing time of *phase-type* distribution. R_i represents the transition rates from some $r \in \mathcal{S}_i^c$ to some $s \in \mathcal{S}_i$. Since $\boldsymbol{\pi} Q = \mathbf{0}$, we have $\boldsymbol{\pi}_{\mathcal{S}_i} Q_i + \boldsymbol{\pi}_{\mathcal{S}_i^c} R_i = 0$. Thus,

$$(\ldots, \pi_s, \ldots)_{s \in \mathcal{S}_i} I = -(\ldots, \pi_r, \ldots)_{r \in \mathcal{S}_i^c} R_i (Q_i)^{-1} = -(\ldots, \sum_{r \in \mathcal{S}_i^c} \pi_r q_{r,s}, \ldots)_{s \in \mathcal{S}_i} (Q_i)^{-1}.$$

Therefore,

$$\sum_{s \in \mathcal{S}_i} \pi_s = -(\ldots, \sum_{r \in \mathcal{S}_i^c} \pi_r q_{r,s}, \ldots)_{s \in \mathcal{S}_i} (Q_i)^{-1} \mathbf{1} = -\lambda_i \sum_{r \in \mathcal{S}_i^c} \pi_r \boldsymbol{\alpha}_i (Q_i)^{-1} \mathbf{1}.$$

Consequently, $\mathbb{E}[T_i] = -\boldsymbol{\alpha}_i (Q_i)^{-1} \mathbf{1} = \frac{1}{\lambda_i} \frac{\sum_{s \in \mathcal{S}_i} \pi_s}{\sum_{r \in \mathcal{S}_i^c} \pi_r}$. □

A.2. Proof of Theorem 2

Proof. From renewal theory [24], if we consider time epochs when content i gets cached, then these time epochs happen to be the regeneration times of a *regenerative process*. Hence, they constitute the event times of a *renewal process*, and we can focus only on a single renewal cycle within two successive renewal epochs to study the behavior of the entire process. Denote H as the number of cache hits within such a renewal cycle. Obviously there is only one cache miss within each cycle, thus the average hit rate can be written as $h_i = E[H]/(E[H]+1)$ [25]. Moreover, $E[T_i]$ is just the average duration that i could stay in cache during each cycle. From *Wald's Equation*, we can derive that $E[H] = \lambda_i E[T_i]$. Finally, there is

$$h_i = \frac{\lambda_i E[T_i]}{\lambda_i E[T_i] + 1} = \frac{\sum_{s \in \mathcal{S}_i} \pi_s}{\sum_{s \in \mathcal{S}_i} \pi_s + \sum_{r \in \mathcal{S}_i^c} \pi_r} = \sum_{s \in \mathcal{S}_i} \pi_s = o_i. \quad \square \tag{A.1}$$

A.3. Proof of Theorem 3

Proof. Obviously there are $\boldsymbol{\pi}^{RR} \mathbf{1} = 1$ and $\boldsymbol{\pi}^{FC} \mathbf{1} = 1$. Without loss of generality, we only show that $\forall r \in \mathcal{S}$, the corresponding column of Q, $\mathbf{q}_r = (\ldots, q_{s,r}, \ldots)$ satisfies $\boldsymbol{\pi} \mathbf{q}_r = \sum_{s \in \mathcal{S}} \pi_s q_{s,r} = 0$, which leads to $\boldsymbol{\pi} Q = 0$. Denote $\mathcal{N}(r) = \{s : s \in \mathcal{S}, |s \cap r| = N - 1\}$, and suppose $s \setminus r = \{i\}$, $r \setminus s = \{j\}$. Since each row of the transition rate matrix Q of a CTMC satisfies $\sum_{s \in \mathcal{S}} q_{s,r} = 0$, $q_{s,s}$ can be derived from the previous definition of Q. Thus,

$$\begin{aligned}
&\sum_{s \in \mathcal{S}} \pi_s^{RR} q_{s,r}^{RR} = \frac{1}{\Sigma} \left(\sum_{s \in \mathcal{N}(r)} W_s \phi_s q_{s,r} + W_r \phi_r q_{r,r} \right) \\
&= \frac{1}{\Sigma} \left(\sum_{s \in \mathcal{N}(r)} W_s \phi_s \frac{w_i}{W_s} \lambda_j + W_r \phi_r \left(- \sum_{k \in r^c} \lambda_k \right) \right) = \frac{1}{\Sigma} \left(\sum_{s \in \mathcal{N}(r)} \phi_r w_j \lambda_i - W_r \phi_r \sum_{k \in r^c} \lambda_k \right),
\end{aligned}$$

where $\Sigma = \sum_{s \in \mathcal{S}} W_s \phi_s$. For any r, $|\mathcal{N}(r)| = N(M - N)$, as for each $j \in r$, there can be $N - M$ possible corresponding i from r^c. Hence, $\sum_{s \in \mathcal{N}(r)} \phi_r w_j \lambda_i = \phi_r \sum_{j \in r} w_j \sum_{i \in r^c} \lambda_i = \phi_r W_r \sum_{i \in r^c} \lambda_i$, which illustrates that $\sum_{s \in \mathcal{S}} \pi_s^{RR} q_{s,r}^{RR} = 0$. Similarly,

$$\sum_{s\in\mathcal{S}}\pi_s^{FC}q_{s,r}^{FC}=\frac{1}{\Sigma}\Big(\sum_{s\in\mathcal{N}(r)}\phi_s q_{s,r}+\phi_r q_{r,r}\Big)$$
$$=\frac{1}{\Sigma}\Big(\sum_{s\in\mathcal{N}(r)}\phi_s\frac{w_i}{W_s+w_j}\lambda_j+\phi_r W_r\Big(-\sum_{k\in r^c}\frac{\lambda_k}{W_r+w_k}\Big)\Big)$$
$$=\frac{1}{\Sigma}\Big(\sum_{j\in r}\phi_r w_j\sum_{i\in r^c}\frac{\lambda_i}{W_r+w_i}-\phi_r W_r\sum_{k\in r^c}\frac{\lambda_k}{W_r+w_k}\Big),$$

where $\Sigma=\sum_{s\in\mathcal{S}}\phi_s$. Since $W_r=\sum_{j\in r}w_j$, we have also shown that $\sum_{s\in\mathcal{S}}\pi_s^{FC}q_{s,r}^{FC}=0$.

For a CTMC, there exists a corresponding discrete-time Markov chain called its Embedded Markov Chain (EMC). Each element of its one-step transition probability matrix P represents the conditional probability of transition between states of the CTMC. P can be derived from $P=I-(diag(Q))^{-1}Q$, where I is the identity matrix and $diag(Q)$ is the diagonal matrix formed by selecting the main diagonal from Q and setting all other elements to 0. Thus, $p_{s,s'}=q_{s,s'}/(-q_{s,s})$ if $s\neq s'$; otherwise it is 0. The stationary probability distribution vector of the EMC, $\boldsymbol{\nu}$, satisfies that $\boldsymbol{\nu}P=\boldsymbol{\nu}$ and $\boldsymbol{\nu}\mathbf{1}=1$. Moreover, $\boldsymbol{\pi}$ can be represented as $\boldsymbol{\pi}=-\boldsymbol{\nu}(diag(Q))^{-1}/\|-\boldsymbol{\nu}(diag(Q))^{-1}\|_1$, that is, π_s is proportional to $\nu_s(-1/q_{s,s})$. From [26], if we consider the sequence of time epochs when the state transitions occur, i.e., $0=T_0<T_1<\cdots<T_n<\cdots$, we then get a semi-Markov process (SMP) defined as $\mathcal{Y}=\{Y_t:=X_n\ for\ T_n\le t\le T_{n+1}\}$. X_n denotes the state of the EMC after the nth state transition, while X_0 is the initial state. Define $m_s:=\mathbb{E}[T_1|X_0=s]$, according to the IRM assumption, the aggregate request arrival process is a Poisson process at rate $-q_{s,s}$. Thus, $m_s=1/(-q_{s,s})$, and the asymptotic probability that the SMP is in state s can be derived from $\psi_s:=\lim_{t\to\infty}Pr\{Y_t=s\}=\nu_s m_s/\sum_{r\in\mathcal{S}}\nu_r m_r=\big(\nu_s(-1/q_{s,s})\big)/\big(\nu_r(-1/q_{r,r})\big)$, which describes the proportion of time that the process is in state s when the total time t is long enough. Thus, the proportion of time that content i is in cache in the long run could be written as $\sum_{s\in\mathcal{S}_i}\psi_s$, which is defined as the occupancy rate o_i. Consequently, we have

$$o_i=\frac{\sum_{s\in\mathcal{S}_i}\nu_s(-1/q_{s,s})}{\sum_{r\in\mathcal{S}}\nu_r(-1/q_{r,r})}=\frac{\sum_{s\in\mathcal{S}_i}\pi_s}{\sum_{r\in\mathcal{S}}\pi_r}=\sum_{s\in\mathcal{S}_i}\pi_s.\quad\square$$

A.4. Proof of Theorem 4

Proof. From Theorem 3,

$$o_i=\frac{\sum_{s\in\mathcal{S}_i}\phi_s}{\sum_{r\in\mathcal{S}}\phi_r}=\frac{\sum_{s\in\mathcal{S}_i}\phi_s}{\sum_{s\in\mathcal{S}_i}\phi_s+\sum_{r\in\mathcal{S}_i^c}\phi_r},$$

where $\phi_s=(\prod_{j\in s}\lambda_j)(\prod_{k\in s^c}w_k)$. Therefore, we can denote $f_{-i}=\sum_{s\in\mathcal{S}_i}\phi_s$ and $g_{-i}=\frac{1}{w_i}\sum_{r\in\mathcal{S}_i^c}\phi_r$, such that f_{-i} and g_{-i} are irrelevant with w_i. As result,

$$o_i=\frac{f_{-i}}{f_{-i}+w_i g_{-i}},\quad\frac{\partial o_i}{\partial w_i}=\frac{-f_{-i}g_{-i}}{(f_{-i}+w_i g_{-i})^2},$$

where $\lambda_i>0$, $w_i>0$, $\forall i\in\mathcal{M}$. We assume here that w_i is always strictly positive; otherwise, $w_i=0$ indicates that i is always cached and o_i is 1, then the (M,N) case is actually reduced to $(M-1,N-1)$. Consequently, $\partial o_i/\partial w_i<0$.

Similarly, given any $j\neq i$, we could further define that $f_{-i}=h_{-j}+w_j l_{-j}$ and $g_{-i}=\tilde{h}_{-j}+w_j\tilde{l}_{-j}$, such that h_{-j}, l_{-j}, $\tilde{h}_{-j}$ and $\tilde{l}_{-j}$ are all irrelevant with j. That is to say,

$$h_{-j}=\sum_{s\in\mathcal{S}_i\cap\mathcal{S}_j}\phi_s,\ l_{-j}=\frac{1}{w_j}\sum_{s\in\mathcal{S}_i\setminus\mathcal{S}_j}\phi_s,\ \tilde{h}_{-j}=\sum_{s\in\mathcal{S}_i^c\cap\mathcal{S}_j}\phi_s,\ \tilde{l}_{-j}=\frac{1}{w_j}\sum_{s\in\mathcal{S}_i^c\setminus\mathcal{S}_j}\phi_s.$$

and

$$\frac{\partial o_i}{\partial w_j}=\frac{\tilde{h}_{-j}l_{-j}-h_{-j}\tilde{l}_{-j}}{(h_{-j}+w_j l_{-j}+\tilde{h}_{-j}+w_j\tilde{l}_{-j})^2}.$$

By definition, $\mathcal{S}_i^c\cap\mathcal{S}_j$ covers all states including j but excluding i, i.e., $s=\{j\}\cup s_1$, where $s_1\subset\mathcal{M}\setminus\{i,j\}$, $|s_1|=N-1$; $\mathcal{S}_i\setminus\mathcal{S}_j$ covers all states including i but excluding j, i.e., $s=\{i\}\cup s_2$, where $s_2\subset\mathcal{M}\setminus\{i,j\}$, $|s_2|=N-1$; $\mathcal{S}_i\cap\mathcal{S}_j$ covers all states including both i and j, i.e., $s=\{i,j\}\cup s_3$, where $s_3\subset\mathcal{M}\setminus\{i,j\}$, $|s_3|=N-2$; and $\mathcal{S}_i^c\setminus\mathcal{S}_j$ covers all states excluding both i and j, i.e., $s=s_4$, where $s_4\subset\mathcal{M}\setminus\{i,j\}$, $|s_4|=N$. Hence, $\tilde{h}_{-j}\cdot w_j l_{-j}$ consists of $\phi_s\phi_r$ from $\binom{M-2}{N-1}\binom{M-2}{N-1}$ pairs of (s,r), where $s\in\mathcal{S}_i^c\cap\mathcal{S}_j$, $r\in\mathcal{S}_i\setminus\mathcal{S}_j$; while $h_{-j}\cdot w_j\tilde{l}_{-j}$ consists of $\phi_{s'}\phi_{r'}$ from $\binom{M-2}{N-2}\binom{M-2}{N}$ pairs of (s',r'), where $s'\in\mathcal{S}_i\cap\mathcal{S}_j$, $r'\in\mathcal{S}_i^c\setminus\mathcal{S}_j$. It can be noticed that $\binom{M-2}{N-1}\binom{M-2}{N-1}>\binom{M-2}{N-2}\binom{M-2}{N}$. Moreover, given any $s'\in\mathcal{S}_i\cap\mathcal{S}_j$, $r'\in\mathcal{S}_i^c\setminus\mathcal{S}_j$, suppose $|s'\cup r'|=n$, then $N+2\le n\le 2N$. Thus, there exist $\binom{n-2}{N-2}\binom{n-2}{N}$ pairs of (s,r) that $s,r\subset(s'\cup r')$ and $s\in\mathcal{S}_i\cap\mathcal{S}_j$, $r\in\mathcal{S}_i^c\setminus\mathcal{S}_j$, such that given any (s,r), $\phi_s\phi_r=\phi_{s'}\phi_{r'}$. Meanwhile, there also exist $\binom{n-2}{N-1}\binom{n-2}{N-1}$ pairs of (s,r) that $s,r\in(s'\cup r')$ and $s\in\mathcal{S}_i\setminus\mathcal{S}_j$, $r\in\mathcal{S}_i^c\cap\mathcal{S}_j$,

such that given any (s, r), $\phi_s\phi_r = \phi_{s'}\phi_{r'}$. Similarly, we can notice that $\binom{n-2}{N-1}\binom{n-2}{N-1} > \binom{n-2}{N-2}\binom{n-2}{N}$. Consequently, we can infer that $h_{-j} \cdot w_j l_{-j}$ consists of more terms in the form of $\phi_s\phi_r$, while also completely covers the entire form of $h_{-j} \cdot w_j \tilde{l}_{-j}$, that is, $\tilde{h}_{-j} \cdot w_j l_{-j} - h_{-j} \cdot w_j \tilde{l}_{-j} > 0$, which indicates that $\partial o_i/\partial w_j > 0$. □

A.5. Proof of Theorem 5

Proof. Since the average arrival rate λ_i can always be considered as strictly positive, if we divide each ϕ_s by $\prod_{i\in\mathcal{M}} \lambda_i$, o_i can thus be represented as

$$o_i = \frac{\sum_{s\in\mathcal{S}_i} \prod_{j\in s^c} (w_j/\lambda_j)}{\sum_{r\in\mathcal{S}} \prod_{k\in r^c} (w_k/\lambda_k)}.$$

Therefore, if the ratio w_i/λ_i is kept constant, i.e.,

$$\frac{w_i}{\lambda_i} = \frac{w_i'}{\lambda_i'}, \quad w_i' = \frac{\lambda_i'}{\lambda_i} w_i, \quad \forall i \in \mathcal{M},$$

then o_i can remain the same. This indicates we can define the normalized weight $v_i = w_i/\lambda_i$ to decouple the impact of arrival rate on the resulting occupancy rates. □

A.6. Proof of Theorem 6

Proof. From Theorem 3, if all the weights are strictly positive, i.e., $w_i > 0,\ \forall i \in \mathcal{M}$, ϕ_s is defined as $\prod_{i\in s} \lambda_i/w_i$, thus,

$$o_i = \frac{\sum_{s\in\mathcal{S}_i} \varphi_s}{\sum_{r\in\mathcal{S}} \varphi_r} = \frac{\sum_{s\in\mathcal{S}_i} \prod_{j\in s} v_j^{-1}}{\sum_{r\in\mathcal{S}} \prod_{k\in r} v_k^{-1}} = \frac{\sum_{s\in\mathcal{S}_i} \prod_{j\in s^c} v_j}{\sum_{r\in\mathcal{S}} \prod_{k\in r^c} v_k}.$$

Hence, it can be noticed by definition that we can denote

$$f_{-i} = \sum_{s\in\mathcal{S}_i} \prod_{j\in r^c} v_j = \prod_{k\in\mathcal{M}} v_k \sum_{s\in\mathcal{S}_i} \prod_{j\in s} v_j^{-1},$$

and

$$g_{-i} = \frac{1}{v_i} \sum_{r\in\mathcal{S}_i^c} \prod_{j\in r^c} v_j = \frac{1}{v_i} \prod_{k\in\mathcal{M}} v_k \sum_{r\in\mathcal{S}_i^c} \prod_{j\in r} v_j^{-1},$$

such that f_{-i} and g_{-i} are irrelevant with i. Therefore,

$$o_i = \frac{f_{-i}}{f_{-i} + v_i g_{-i}}, \quad v_i = \frac{1 - o_i}{o_i} \frac{f_{-i}}{g_{-i}}.$$

Now we denote $\bar{v}_{-i} = f_{-i}/g_{-i}$, then $v_i = \bar{v}_{-1}(1 - o_i)/o_i$, which leads to $o_i = \bar{v}_{-i}/(\bar{v}_{-i} + v_i)$. □

A.7. Proof of Theorem 7

Proof. From Theorems 3 and 6, o_i can be written as

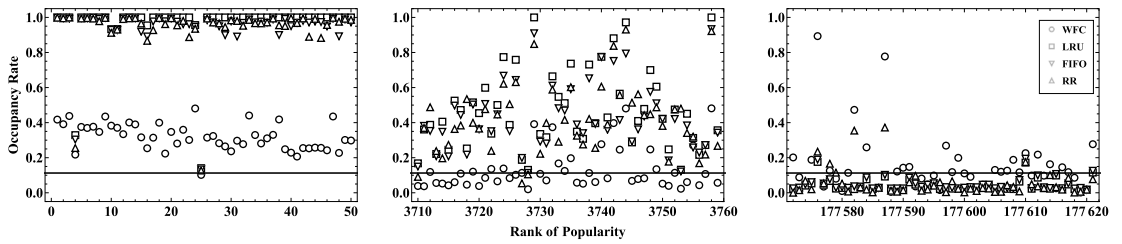
$$o_i = \frac{\sum_{s\in\mathcal{S}_i} \prod_{j\in s} v_j^{-1}}{\sum_{r\in\mathcal{S}} \prod_{k\in r} v_k^{-1}}.$$

Since each content from class l is now assigned the normalized weight of class, i.e., $\hat{v}_l$, any state s can now be interpreted as a corresponding allocation of the cache resource to each class, i.e., $\mathbf{n} = (n_1, \ldots, n_M)$. Hence, $\prod_{j\in s} v_j^{-1}$ turns into $\prod_{l=1}^{K} (\hat{v}_l^{n_l})^{-1}$ under class-based resource allocation, and all possible states of the cache correspond to all possible ways of allocation described by $\mathbf{n}$. Given a resource allocation $\mathbf{n} = (n_1, \ldots, n_M)$ that allocates n_l amount of cache resource to class l, there are $\prod_{l=1}^{K} \binom{M_l}{n_l}$ corresponding states since any n_l from the M_l content items in class l can be allocated the resource, while such possible allocation $\mathbf{n}$ is defined on the space constrained by $\sum_{l=1}^{K} n_l = N$ and $0 \le n_l \le M_l,\ \forall l = 1, \ldots, K$. As result,

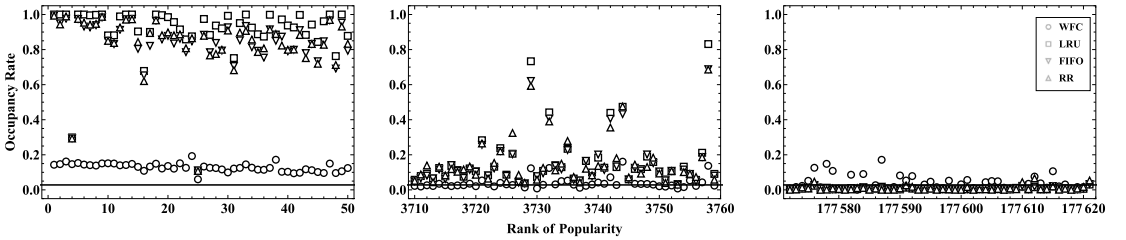
$$O \coloneqq \sum_{r\in\mathcal{S}} \prod_{k\in r} v_k^{-1} = \sum_{\mathbf{n}\in\mathcal{N}} \prod_{l=1}^{K} \binom{M_l}{n_l} \prod_{l=1}^{K} (\hat{v}_l^{n_l})^{-1}.$$

Similarly, if content i is in class k, then all possible states in $\mathcal{S}_i$ correspond to all possible ways of allocation that grant n_l amount of cache resource to any other class l, and $n_k - 1$ available cache resource to the remaining $M_k - 1$ content items in class k. That is to say,

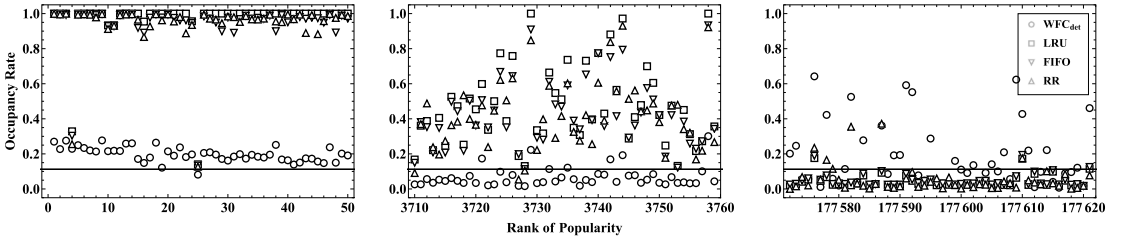
$$O_k \coloneqq \sum_{s\in\mathcal{S}_i} \prod_{j\in s} v_j^{-1} = \sum_{n\in\mathcal{N}_i} \frac{\binom{M_k-1}{n_k-1}}{\binom{M_k}{n_k}} \prod_{l=1}^{K} \binom{M_l}{n_l} \prod_{l=1}^{K} (\hat{v}_l^{n_l})^{-1} = \sum_{n\in\mathcal{N}_i} \frac{n_k}{M_k} \prod_{l=1}^{K} \binom{M_l}{n_l} \prod_{l=1}^{K} (\hat{v}_l^{n_l})^{-1},$$



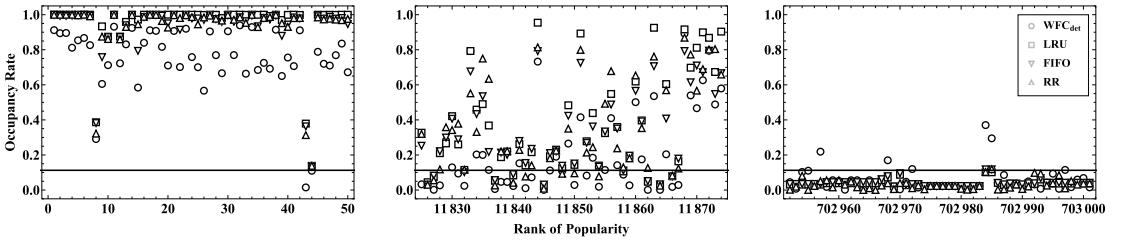
(a) Comparing WFC with LRU, FIFO and RR. Cache size $N = 20000$.



(b) Comparing WFC with LRU, FIFO and RR. Cache size $N = 5000$.



(c) Comparing $\mathrm{WFC}_{\mathrm{det}}$ with LRU, FIFO and RR. Cache size $N = 20000$.



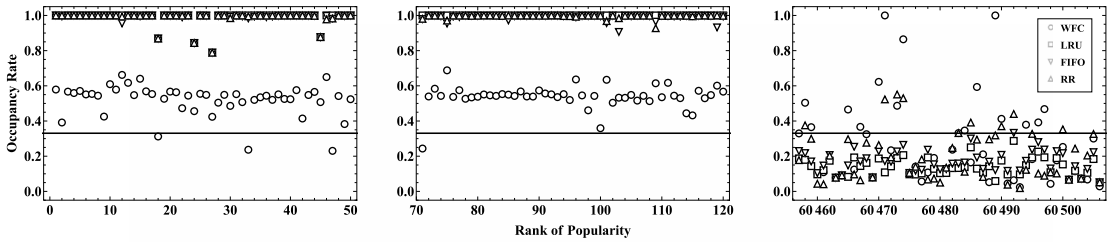
(d) Comparing $\mathrm{WFC}_{\mathrm{det}}$ with LRU, FIFO and RR. Resources are partitioned into chunks of size 10^6. Cache size $N = 79160$.

Fig. B.11. Occupancy rates of (1) the most popular 50 videos, (2) 50 videos around the video where accumulated popularity accounts for half of the total popularity and (3) the least popular 50 videos from Dataset A.

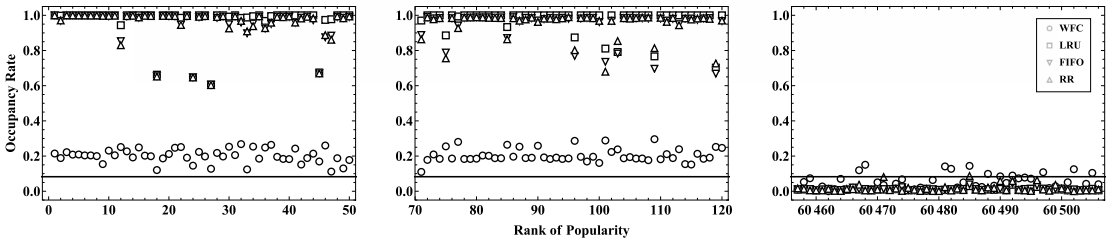
while now the possible allocation $\mathbf{n}$ is defined on the space constrained by $\sum_{l=1}^{K} n_l = N$ and $0 \leq n_k - 1 \leq M_k - 1,\ 0 \leq n_l \leq M_l,\ \forall l = 1, \ldots, K, l \neq k$. □

Appendix B. Additional experimental validation

Instead of focusing on the average value or standard deviation which converges all the values into one, we plot individual occupancy rates to inspect their distribution in Fig. B.11, as they also reflect the fairness of the caching policy. We notice that for Dataset A, the most popular 3735 videos account for 50% of the total popularity. Hence, we show the occupancy rates of (1) the most popular 50 videos (rank 1 to 50), (2) 50 videos around the 3735th most popular video (rank 3710 to 3759), (3)



(a) Comparing WFC with LRU, FIFO and RR. Cache size $N = 20000$.



(b) Comparing WFC with LRU, FIFO and RR. Cache size $N = 5000$.

Fig. B.12. Occupancy rates of (1) the most popular 50 videos, (2) 50 videos around the video where accumulated popularity accounts for half of the total popularity and (3) the least popular 50 videos from Dataset B.

the least popular 50 videos (rank 177 572 to 177 621) to check the occupancy rates of videos with different popularity. The solid horizontal line represents the average occupancy rate N/M, and theoretically all the points fall on it if IRM assumption is satisfied. We observe that in Fig. B.11(a), although WFC does not show standard deviation as low as other policies such as FIFO when $N = 20\,000$, only WFC attempts to bring all the occupancy rates around N/M that represents a fair allocation of cache, while other policies do not take this into account, especially for those popular videos that can gain occupancy rates close to 1. However, we also notice that for less popular contents, WFC sometimes causes a rather huge fluctuation in occupancy rates, probably because the effect of the randomness in cache replacement can be amplified when the request arrivals are infrequent, so some unpopular videos can stay longer than expected in the cache due to their low weights. This observation explains the slightly higher standard deviation of occupancy rates.

Compared to Fig. B.11(b) where $N = 5000$, we notice that WFC performs better in terms of fairness when the cache size is smaller, as the plot points become closer to N/M. When the cache size is smaller, a content tends to stay for a shorter period of time in the cache before it get replaced by another content, or to say cache replacements occur more often, thus the randomness brought by the probabilistic cache replacement has less influence on the distribution of the occupancy rates. Compared to Fig. B.11(c) where we use WFC_{det}, we understand that WFC_{det} performs better given same cache size $N = 20\,000$, as it has a global view on the popularity of contents, not only partially estimated from a sliding window. For Dataset B, the 96 most popular videos account for 50% of the popularity. Comparing 3735/177 621 with 96/60 506, we can also understand the difference between the popularity distribution of Dataset A and Dataset B. We observe similar results in Fig. B.12, that only WFC attempts to achieve fairness by bringing occupancy rates around N/M regardless of the content popularity, and it performs better under a smaller cache size.

Since Datasets A and B both lack information on the complete size of each video, we assume that each cache slot is able to accommodate a video content in our cache simulation. However, Dataset A provides information on how large each time a video is partially loaded upon a request in one field per record, thus we may revise WFC to see if partitioning videos into equal-sized chunks influences the results. We divide each video into chunks of size 10^6, and treat each chunk as a separate video. Hence, 177 621 videos correspond to 703 001 chunks, and we also scale the cache size such that N/M remains the same as when $N = 20\,000$ in the previous simulation. The 11850 most popular chunks take 50% of the total popularity. We observe that in Fig. B.11(d), WFC_{det} still attempts to bring down the occupancy rates of the popular videos compared to other policies, however the effect is not that good compared to Fig. B.11(c). We believe this is because such partial video size information still cannot reveal the actual size of videos, as new request may demand an uncached chunk of the video which causes the discrepancies in occupancy rates.

References

[1] A.K. Parekh, R.G. Gallager, Ageneralized processor sharing approach to flow control in integrated services networks: the single-node case, IEEE/ACM Trans. Netw. 1 (3) (1993) 344–357.

[2] A. Demers, S. Keshav, S. Shenker, Analysis and simulation of a fair queueing algorithm, in: ACM SIGCOMM Computer Communication Review, vol. 19, (4) ACM, 1989, pp. 1–12.
[3] E.G. Coffman, P.J. Denning, Operating Systems Theory, vol. 973, Prentice-Hall Englewood Cliffs, NJ, 1973.
[4] M.N. Garofalakis, Y.E. Ioannidis, Parallel query scheduling and optimization with time-and space-shared resources, SORT 1 (T2) (1997) T3.
[5] J.C. Bennett, H. Zhang, WF^2Q: worst-case fair weighted fair queueing, in: INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE ,vol. 1, IEEE, 1996, pp. 120–128.
[6] P. Goyal, H.M. Vin, H. Chen, Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks, in: ACM SIGCOMM Computer Communication Review, vol. 26, (4) ACM, 1996, pp. 157–168.
[7] I. Stoica, H. Abdel-Wahab, K. Jeffay, S.K. Baruah, J.E. Gehrke, C.G. Plaxton, A proportional share resource allocation algorithm for real-time, time-shared systems, in: Real-Time Systems Symposium, 1996., 17th IEEE, IEEE, 1996, pp. 288–299.
[8] C.A. Waldspurger, W.E. Weihl, Lottery scheduling: flexible proportional-share resource management, in: Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation, USENIX Association, 1994, p. 1.
[9] C.A. Waldspurger, W.E. Weihl, Stride Scheduling: Deterministic Proportional Share Resource Management, Massachusetts Institute of Technology. Laboratory for Computer Science, 1995.
[10] A. Dan, D. Towsley, An Approximate Analysis of the LRU and FIFO Buffer Replacement Schemes, vol. 18, (1) ACM, 1990.
[11] E. Gelenbe, A unified approach to the evaluation of a class of replacement algorithms, IEEE Trans. Comput. 100 (6) (1973) 611–618.
[12] H. Che, Y. Tung, Z. Wang, Hierarchical web caching systems: modeling, design and experimental results, IEEE J. Sel. Areas Commun. 20 (7) (2002) 1305–1314.
[13] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, Y. Tay, A utility optimization approach to network cache design, in: Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on, IEEE, 2016, pp. 1–9.
[14] N.C. Fofack, P. Nain, G. Neglia, D. Towsley, Performance evaluation of hierarchical ttl-based cache networks, Comput. Netw. 65 (2014) 212–231.
[15] D.S. Berger, P. Gland, S. Singla, F. Ciucu, Exact analysis of ttl cache networks, Perform. Eval. 79 (2014) 2–23.
[16] R.T.B. Ma, D. Towsley, Cashing in on caching: on-demand contract design with linear pricing, in: Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, 2015.
[17] A. Ferragut, I. Rodríguez, F. Paganini, Optimizing ttl caches under heavy-tailed demands, in: Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science, ACM, 2016, pp. 101–112.
[18] E. Quinones, E.D. Berger, G. Bernat, F.J. Cazorla, Using randomized caches in probabilistic real-time systems, in: Real-Time Systems, 2009. ECRTS'09. 21st Euromicro Conference on, IEEE, 2009, pp. 129–138.
[19] K. Psounis, B. Prabhakar, A randomized web-cache replacement scheme, in: INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 3, IEEE, 2001, pp. 1407–1415.
[20] L. Wang, G. Tyson, J. Kangasharju, J. Crowcroft, FairCache: introducing fairness to icn caching, in: Network Protocols (ICNP), 2016 IEEE 24th International Conference on, IEEE, 2016, pp. 1–10.
[21] Q. Pu, H. Li, M. Zaharia, A. Ghodsi, I. Stoica, Fairride: near-optimal, fair cache sharing, in: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), USENIX Association, 2016, pp. 393–406.
[22] E. Hargreaves, C. Agosti, D. Menasché, G. Neglia, A. Reiffers-Masson, E. Altman, Fairness in online social network timelines: measurements, models and mechanism design, in: IFIP Performance 2018, 2018.
[23] P. Buchholz, J. Kriege, I. Felko, Input Modeling with Phase-type Distributions and Markov Models: Theory and Applications, Springer, 2014.
[24] S.M. Ross, Stochastic Processes. 1996, Wiley, New York, 1996.
[25] J. Jung, A.W. Berger, H. Balakrishnan, Modeling ttl-based Internet caches, in: INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, vol. 1, IEEE, 2003, pp. 417–426.
[26] J. Janssen, R. Manca, Applied Semi-Markov Processes, Springer Science & Business Media, 2006.