Reinforcement Learning for Dynamic Dimensioning of Cloud Caches: A Restless Bandit Approach

Guojun Xiong, Shufan Wang, Gang Yan, Jian Li
SUNY-Binghamton University
{gxiong1,swang214,gyan2,lij}@binghamton.edu

***Abstract*—We study the dynamic cache dimensioning problem, where the objective is to decide how much storage to place in the cache to minimize the total costs with respect to the storage and content delivery latency. We formulate this problem as a Markov decision process, which turns out to be a restless multi-armed bandit problem and is provably hard to solve. For given dimensioning decisions, it is possible to develop solutions based on the celebrated Whittle index policy. However, Whittle index policy has not been studied for dynamic cache dimensioning, mainly because cache dimensioning needs to be repeatedly solved and jointly optimized with content caching. To overcome this difficulty, we propose a low-complexity fluid Whittle index policy, which jointly determines dimensioning and content caching. We show that this policy is asymptotically optimal. We further develop a lightweight reinforcement learning augmented algorithm dubbed fW-UCB when the content request and delivery rates are unavailable. fW-UCB is shown to achieve a sub-linear regret as it fully exploits the structure of the near-optimal fluid Whittle index policy and hence can be easily implemented. Extensive simulations using real traces support our theoretical results.**

I. INTRODUCTION

Content delivery networks (CDNs) carry more than 50% of the Internet traffic today, and this number is predicted to increase over the coming years [1]. However, the entry cost of traditional CDNs is high, especially for small content providers (CPs) since the lease is on a long-term basis with fixed prices. This motivated the popular "cloud CDNs", which provide managed platforms with a pay-as-you-go model for CPs. For example, Amazon AWS [2] provides both *cache dimensioning* and caching implementation software services to CPs. A CP can lease storage from AWS and implement caching policy using open software such as Amazon CloudFront [3], which allows dynamic storage scaling. Caching dimensioning in cloud CDNs provides large economic yields to CPs since CPs can shut down storage in clouds when the traffic is low.

At the same time, this new model brings significant challenges. First, the CPs not only have to decide how much storage to lease to meet user content requests, but also what content to store in cache. This challenge is further exacerbated in cloud CDNs, where the storage is leased on-the-fly using cloud resources, and dimensioning needs to be solved regularly, often on a per-day or per-hour basis, to accommodate the time-varying nature of content requests. Finally, leasing more storage improves caching performance, e.g., reduced user content service latency, but also incurs additional expenditure. With a given operational expenditure budget, all cache dimensioning decisions are strongly coupled over time. Indeed, finding the optimal tradeoff between maximizing caching performance and minimizing cache dimensioning costs naturally leads to a new set of algorithmic challenges as the caching performance also depends on the variability of content popularity.

In this paper, we are interested in jointly optimizing cache dimensioning and content caching when a CP regularly leases storage from cloud CDNs. We note that there is a natural timescale separation between cache dimensioning and content caching, where the former is a much slower operation than the latter. Using this observation, we formulate the problem of dynamic dimensioning of cloud caches as a two-timescale Markov decision process (MDP) [4] in Section III, where the goal is to propose provably optimal algorithms to minimize the total expected costs due to cache dimensioning and content delivery latency. This MDP turns out to be a restless multi-armed bandit (RMAB) problem [5]. Though in theory it can be solved by value iteration [4], this approach suffers from the curse of dimensionality. Therefore, it is highly desirable to design provably optimal and low-complexity solutions. A celebrated heuristic is the Whittle index policy [5], which relaxes the hard constraint, in which the number of cached contents is *exactly* the leased storage, to a time-averaged constraint, in which the number of cached contents is the leased storage *on average*. However, to the best of our knowledge, there is no such Whittle index policy for dynamic dimensioning of cloud caches. Part of the difficulty is that Whittle index policy is not feasible when the lease storage is unknown, which needs to be repeatedly solved and jointly optimized with content caching.

In this paper, we design a new Whittle-like index policy for dynamic dimensioning of cloud caches. Similar to Whittle [5], we first relax the MDP, and then overcome the above difficulties of directly designing Whittle policy by studying the corresponding fluid dynamics, the optimal solution to which is described by a linear programming (LP). The optimal values of the LP (i.e., the optimal dimensioning decisions) provide a lower bound on the cost of the original MDP [6], [7], and are then used to design Whittle index policy for the original MDP. We show that our original MDP is indexable, and derive explicitly the Whittle indices of each content. Based on this result, we propose an associated policy by caching contents with the largest Whittle indices whose number is constrained by the optimal dimensioning decisions. Finally, we prove

This work was supported in part by NSF grant CRII-CNS-2104880 and the U.S. Department of Energy grant DE-EE0009341. The authors have provided public access to their code at https://git.io/JyShk.

that our proposed index policy is asymptotically optimal. Our contribution in Section IV is non-trivial since establishing the Whittle indexablity of RMAB problems is typically intractable [8] and the Whittle indices of many practical problems remain unknown except for a few special cases. Exacerbating this problem is the fact that the cache dimensioning and content caching decisions are coupled in our problem.

Due to the time-varying nature of cloud CDNs, the system parameters such as content request and delivery rates are typically unknown. In Section V, we further propose a reinforcement learning (RL) augmented algorithm to address this issue. However, simply applying off-the-shelf methods such as UCRL2 [9] or Thompson Sampling [10] is tricky due to the curse of dimensionality, since these generic RL approaches ignore the rich underlying structure of our problem and hence are inefficient of learning in our settings. To this end, we propose *fW-UCB* that not only leverages the approach of *optimism-in-the-face-of-uncertainty* [9], [11] to balance exploration and exploitation, but more importantly, it learns to leverage the near-optimal index policy for making decisions. We show that *fW-UCB* achieves an optimal sub-linear regret with a low-complexity, and hence can be easily implemented in real systems. To the best of our knowledge, our work is the first in the literature to design an index based reinforcement learning algorithm for dynamic dimensioning of cloud caches.

We support our analytical results with extensive simulations in Section VI using both synthetic and real-world traces, which demonstrate the superior performance of our proposed policies over state of the arts that are employed in today's major CDNs. Furthermore, our simulation results are in close agreement with theoretical models and confirm a desired tradeoff between cache dimensioning and content delivery latency costs, which can be achieved by tuning a system-wide parameter.

II. Related Work

In this section, we mainly overview two main areas that are closely related to our work: cache dimensioning and restless multi-armed bandit, and further provide a brief discussion of our design methodology in the context of prior work.

Cache Dimensioning. Though offline cache dimensioning with known content requests has been studied for designing CDN systems prior to their deployment [12], [13], it is not appropriate for cloud CDNs with unknown and time-varying content requests. Online cache dimensioning has received little attention to date. A TTL-based approximation approach was proposed for elastic cloud provisioning to minimize the storage and miss costs [14]. We instead formulate the problem as a MDP and study the storage and content delivery latency costs, which are critical for many emerging delay-sensitive applications. Cache dimensioning in wireless CDNs studied in [15] requires strong assumptions on request process, while our proposed research provides provable performance for any request process. Similar problem was considered in memory management systems [16] for the assignment of memory to various applications with a computationally intensive policy.

Restless Multi-Armed Bandit. The RMAB is a general model for sequential decision making problems ranging from wireless communication [17]–[19], cloud computing [20], queueing systems [21], etc. However, the RMAB is PSPACE hard [22]. One celebrated heuristic is the Whittle index policy [5]. Since then, many studies focused on finding index policies for RMABs, e.g., [7], [23]–[26]. These works assume that system parameters are known. Since the true parameters are typically unavailable, it is important to examine RMAB from a learning perspective, e.g., [18], [27]–[34]. However, these methods contend directly with an extremely high dimensional state-action space yielding the algorithms to be too slow to be of any practical use. Reinforcement learning based algorithms have been developed in recent works [35]–[38] to explore the problem structure through index policy. However, they either are designed for one class of decision variables (e.g., only caching), unlike us with two coupled variables (i.e., dimensioning and caching); or lack finite-time performance analysis and multi time-scale stochastic approximation algorithms usually suffer from the slow convergence.

Our Design Philosophy. This paper contributes to both areas. First, we pose the cache dimensioning problem as a RMAB to minimize the overall costs, including not only dimensioning costs, bust also content delivery latency costs. This enables us to characterize tradeoffs between those two interrelated costs. Second, we depart from existing assumptions on content request processes and consider this RMAB from an online perspective. A key differentiator between our approach and existing ones stems from two perspectives: (i) we focus on designing index policies for dynamic cache dimensioning, which operate on a much smaller dimensional subspace by exploiting the inherent structure of our problem; and (ii) our index-policy approach naturally lends itself to a lightweight RL based framework that can fully exploit the structure of our index policy so as to reduce the high computational complexity and achieve an optimal sub-linear regret.

III. Model and Problem Formulation

In this section, we present the system model and formulate the dynamic cache dimensioning problem.

A. System Model

Consider a system where a CP dynamically leases storage from cloud CDNs to provide services to users. Let $\mathcal{N} = \{1, \cdots, N\}$ be the *content catalog*[1] with each integer representing a different content of equal size. We note that there is a natural timescale separation between cache dimensioning and content caching where the former is a much slower operation than the latter. To this end, we consider a two-timescale dynamic system. Specifically, the cache dimensioning is performed in a slower timescale indexed by $k \in \mathcal{K} = \{1, \cdots, K\}$

[1]Our formulation allows the content catalog to be dynamic. Following our design philosophy, for the index policy design in Section IV, we assume that the content catalog $\mathcal{N}$ remains static throughout all K frames, but the number of requests to each content is bounded and varies across frames. A content is also possible to be never requested within one frame. This assumption is fully relaxed in our RL solutions in Section V.

and $K < \infty$. We call each k as one *frame* with a fixed duration, e.g., 1 hour or 1 day in some real-world systems.

Requests. Requests for content $n \in \mathcal{N}$ follow a Poisson process[2] with arrival rate $\lambda_{n,k}$ in frame k. The time taken to deliver content n to users in frame k is a random variable that is exponentially distributed with mean $1/\mu_{n,k}$, which is independently across different contents. Note that the content arrival rates and delivery rates often vary across different frames due to the time-varying nature of CDN systems which necessitates the need of dynamic cache dimensioning for CPs.

Cache Dimensioning. At the beginning of each frame, the CP determines how much storage to lease from the cloud CDN. We denote the leased storage in frame k as a random variable B_k, which is measured in the number of content size units. Note that B_k must be non-negative as it is impossible to sell storage, and it is meaningless to lease more than N content size units since that would be enough to fit the entire content catalog. Hence we have $B_k \in [0, N]$, $\forall k$.

Content Caching. We use a binary variable A_n to indicate caching decisions on content n, where $A_n = 1$ means content n is cached and $A_n = 0$, otherwise. Since the CP leases B_k storage in frame k, the set of feasible content caching decisions in frame k is $\mathcal{A}_k = \{A_n \in \{0,1\}^N : \sum_{n=1}^N A_n \leq B_k\}$.

B. System Dynamics

Now we formulate the dynamic cache dimensioning problem for the above model as an MDP.

States. We denote the number of outstanding requests for content n at time t in frame k as $S_n(k,t) \leq S_{\max}$, where $S_{\max}$ is the maximum number of requests expected in each frame for any content, and can be arbitrarily large but bounded. Denote $\mathbf{S}(k,t) = (S_1(k,t), \cdots, S_N(k,t))$. For the ease of readability, we denote $\mathcal{S}$ as the finite state space in our system.

Actions. The action $A_n(k,t)$ for content n at time t in frame k is defined as whether to cache content n or not. Hence $A_n(k,t) \in \mathcal{A}_k$, $\forall t$. Since the cache dimensioning decision B_k is fixed in frame k and for the abuse of notation, denote $\mathbf{A}(k,t) := (A_1(k,t), \cdots, A_N(k,t), B_k)$, and $\mathbf{A} = \{(\mathbf{A}(1,t), \cdots, \mathbf{A}(K,t)), \forall t\}$. Decisions are made only at those time instants when either a new request arrives, or a content delivery occurs. As a result, $\mathbf{A}(k,t)$ stays unchanged in between these time instants.

A cache dimensioning policy π maps the state of the system $\mathbf{S}(k,t)$ to the action $\mathbf{A}(k,t)$, i.e., $\mathbf{A}(k,t) = \pi(\mathbf{S}(k,t))$.

Controlled Transition Kernel. In frame k, the state of content n can change from S_n to either $S_n + 1$ or $(S_n - 1)_+$ with the transition rates satisfy

$$\mathbf{S} \to \begin{cases} \mathbf{S} + \mathbf{e}_n, & \text{with transition rate } b_{n,k}(S_n, A_n), \\ \mathbf{S} - \mathbf{e}_n, & \text{with transition rate } d_{n,k}(S_n, A_n), \end{cases} \tag{1}$$

where $\mathbf{e}_n$ is a N-dimensional vector with the n-th entry being 1 and all other elements being zero, and $b_{n,k}(S_n, A_n) = \lambda_{n,k}$, $d_{n,k}(S_n, A_n) = \mu_{n,k}(S_n)A_n$, with $\mu_{n,k}(0) = 0$. We allow a state-dependent content delivery rate, which enables us to model realistic settings [45], [46]. In this paper, we consider the classic $M/M/k$ queue, i.e., $d_{n,k}(S_n, A_n) = \mu_{n,k} S_n A_n$.

C. Problem Formulation

Dynamic Cache Dimensioning Problem. A CP may have different requirements on content delivery latency and dimensioning costs. A CP for applications that are more delay-sensitive may place a greater penalty for its content delivery latency, while a CP with a smaller dimensioning expenditure may be more sensitive to the dimensioning costs. To posit a tradeoff between content delivery latency and dimensioning cost, we incorporate unit costs for content delivery latency, c_d and for cache dimensioning, c_b.

Our goal is to derive a policy π to minimize the total expected costs incurred by cache dimensioning and content delivery latency as defined below:

$$C_\pi(\mathbf{A}) = \kappa C_{\text{latency}} + (1-\kappa) C_{\text{dim}}, \tag{2}$$

where C_{latency} is the total content delivery latency cost, C_{dim} is the total cache dimensioning cost and $\kappa \in [0,1]$ is a system-wide weighting factor that determines how the two costs are weighted against each other. By Little's Law, the total expected content delivery latency cost under policy π is given by

$$C_{\text{latency}} = \mathbb{E}_\pi \left(c_d \sum_{k=1}^K \limsup_{T_k \to \infty} \sum_{n=1}^N \frac{1}{T_k} \int_{t=1}^{T_k} S_n(k,t) dt \right), \tag{3}$$

where the subscript denotes the fact that expectation is taken with respect to the measure induced by the policy π. We focus on Markovian policies which base their decisions on the current state and time. Similarly, we have $C_{\text{dim}} = \sum_{k=1}^K c_b B_k$. Then we can write the overall cost under policy π as

$$C_\pi(\mathbf{A}) = \mathbb{E}_\pi \Bigg(\kappa c_d \sum_{k=1}^K \limsup_{T_k \to \infty} \sum_{n=1}^N \frac{1}{T_k} \int_{t=1}^{T_k} S_n(k,t) dt + (1-\kappa) c_b \sum_{k=1}^K B_k \Bigg). \tag{4}$$

Therefore, our dynamic cache dimensioning problem to minimize the total costs subject to the dimensioning constraint in each frame can be formulated as the following MDP:

$$\min_{\pi \in \Pi} \; C_\pi(\mathbf{A}), \quad \text{s.t.} \sum_{n=1}^N A_n(k,t) \leq B_k, \; \forall k, t. \tag{5}$$

We refer to (5) as the "original problem".

Remark 1. *We cannot directly apply the conventional value iteration method [4] to solve (5) since the cache dimensioning and content caching decisions are strongly coupled and need to be jointly optimized in (5). Even when the cache dimensioning decisions, i.e., B_k, $\forall k$ are given, it is well known that solving the above MDP using value iteration suffers from the curve of dimensionality and lacks of insights. As a result, many efforts have been focusing on designing low-complexity*

[2]Poisson arrivals are widely used in the literature, e.g., [39]–[43]. However, our model holds for general stationary process [44] and our RL solutions in Section V hold for any request process. See Section V for details.

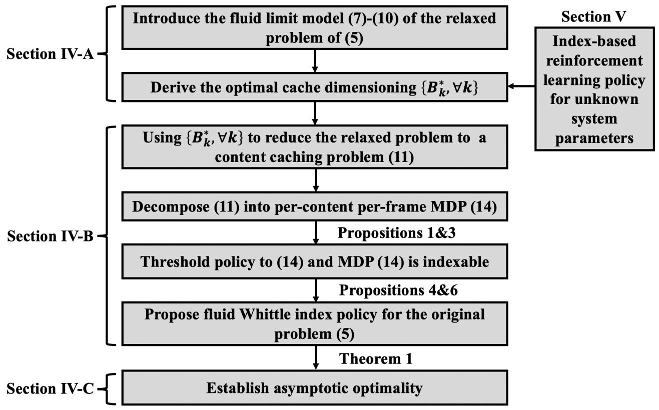
Section IV-A
Introduce the fluid limit model (7)-(10) of the relaxed problem of (5)
Derive the optimal cache dimensioning $\{B_k^*, \forall k\}$
Section V
Index-based reinforcement learning policy for unknown system parameters
Section IV-B
Using $\{B_k^*, \forall k\}$ to reduce the relaxed problem to a content caching problem (11)
Decompose (11) into per-content per-frame MDP (14)
Propositions 1&3
Threshold policy to (14) and MDP (14) is indexable
Propositions 4&6
Propose fluid Whittle index policy for the original problem (5)
Theorem 1
Section IV-C
Establish asymptotic optimality

Fig. 1: The framework of the proposed solution.

solutions. One celebrated heuristic is the Whittle index policy [5]. However, Whittle index policy is infeasible if the resource constraint is not given or unknown, which in our case is the cache dimensioning B_k in each frame. This is because in the RMAB literature, whether to activate an arm (resp. whether to store a content in our model) is not only determined by its Whittle index value, but also depends on constrained resource (resp. B_k in our model). To overcome this challenge, we next introduce a new notion of fluid Whittle index policy.

IV. Fluid Whittle Index Policy

In this section, we propose asymptotically optimal index policies when cache dimensioning and content caching decisions are coupled. Specifically, we introduce the notation of *fluid Whittle index policy*, which generalizes the classic Whittle index policy to the dynamic cache dimensioning setting.

Our proposed policy is based on a relaxed formulation of (5) and consists of two interdependent steps, as illustrated in Figure 1. First, we show that the optimal cache dimensioning decisions in each frame can be solved using a fluid version of the relaxed problem. Second, given the optimal cache dimensioning decisions, we show that our problem is Whittle indexable and explicitly derive the Whittle indices. These two steps enable us to design the *fluid Whittle index policy*, which we show is asymptotically optimal. Note that the our relaxed formulation and hence the fluid problem is very general and can be applied to other MDPs. Hence our methodologies not only apply to the dynamic cache dimensioning problem in this paper but also to other large MDP problems with coupled decision-makings on resource allocation and scheduling.

A. Optimal Fluid Control

In this subsection, we first introduce the relaxed formulation. We then solve the fluid version of this relaxed formulation, i.e., we only take into account the average behavior of the system. The optimal fluid solution provides a lower bound on the optimal cache dimensioning in the original model [6].

Following Whittle [5], we study the relaxed problem in which the cache dimensioning constraint at each time in a frame in (5) is satisfied on average, i.e.,

$$\min_{\pi\in\Pi} \quad C_\pi(\mathbf{A}),$$

$$\text{s.t.} \quad \limsup_{T_k\to\infty} \frac{1}{T_k}\mathbb{E}_\pi \int_{t=1}^{T_k} \sum_{n=1}^{N} A_n(k,t)dt \leq B_k,\ \forall k. \quad (6)$$

We then define the fluid limit model of the relaxed problem under a stationary Markovian policy π, which is independent of the initial state of the MDP [6], [47], [48]. Let $x_{n,k,s}^a$ be the fraction of content n in state s under action a at frame k and let $x_{n,k,s} = x_{n,k,s}^0 + x_{n,k,s}^1$ be the fraction of content n in state s. Similar to [7], we focus on finding an optimal equilibrium point of the fluid dynamics that minimize the total costs. More precisely, we formulate the following LP problem:

$$\min_{\{x_{n,k,s}\},\{B_k\}} \kappa c_d \sum_{k=1}^{K}\sum_{n=1}^{N}\sum_{(s,a)} s x_{n,k,s}^a + (1-\kappa)c_b \sum_{k=1}^{K} B_k \quad (7)$$

$$\text{s.t.} \sum_{n=1}^{N}\sum_{s\in\mathcal{S}} x_{n,k,s}^1 \leq B_k,\ \forall k, \quad (8)$$

$$\sum_a\sum_{s'} x_{n,k,s}^a P_{n,k}(s'|s,a) = \sum_a\sum_{s'} x_{n,k,s'}^a P_{n,k}(s|s',a), \quad (9)$$

$$\sum_{s,a} x_{n,k,s}^a = 1, x_{n,k,s}^a \geq 0, 0 \leq B_k \leq N,\ \forall n,s,a,k, \quad (10)$$

where (8) is equivalent with the constraint in (6), constraint (9) represents the equilibrium condition of the fluid system, i.e., the fluid flows in state s equals to the fluid flows out state s, and (10) holds due to the definition of $x_{n,k,s}^a$.

Denote the optimal cache dimensioning solutions to this LP as $\{B_k^\star, \forall k\}$. It is well-known that this LP is equivalent to the relaxed problem (6) [6] and there exists a stationary policy for this LP [7]. Furthermore, the fluid analysis achieves a lower bound of the original problem [5], [7], which is leveraged for the asymptotic optimality analysis of index policy.

B. Fluid Whittle Index Policy

Since there exists a stationary Markovian policy π_k for each frame $k \in \mathcal{K}$ that is independent of the initial state, the fluid control in (7)-(10), or equivalently the relaxed problem (6) can be decomposed into each frame k. To this end, we decompose the relaxed problem (6) to obtain the following "per-frame MDPs" given the optimal caching dimensioning $B_k^\star$, which only need to make content caching decisions in each frame k:

$$\min_{\pi_k\in\Pi} \quad \mathbb{E}_{\pi_k} \limsup_{T_k\to\infty} \frac{1}{T_k} \sum_{n=1}^{N} \int_{t=1}^{T_k} S_n(k,t)dt$$

$$\text{s.t.} \quad \limsup_{T_k\to\infty} \frac{1}{T_k}\mathbb{E}_{\pi_k} \int_{t=1}^{T_k} \sum_{n=1}^{N} A_n(k,t)dt \leq B_k^\star, \quad (11)$$

where we drop the constant κc_d for simplicity but we consider it when compute the total costs. Next we consider the following Lagrangian associated with (11),

$$L_{\pi_k}(W_k) = \limsup_{T_k\to\infty} \frac{1}{T_k}\mathbb{E}_{\pi_k} \int_{t=1}^{T_k} \Bigg[\sum_{n=1}^{N} S_n(k,t) - W_k\Big(B_k^\star - \sum_{n=1}^{N} A_n(k,t)\Big)\Bigg]dt, \quad (12)$$

where W_k is the Lagrangian multiplier for the k-th frame. Then dual function is then defined as

$$D(W_k) := \min_{\pi_k} L_{\pi_k}(W_k). \tag{13}$$

A key observation made by Whittle is that (13) can be decomposed into N subproblems, one for each content n. As a result, we obtain the following "per-content per-frame MDP":

$$\min_{\pi_k} \limsup_{T_k\to\infty} \frac{1}{T_k}\mathbb{E}_{\pi_k}\int_{t=1}^{T_k}\Big(S_n(k,t)-W_k(1-A_n(k,t))\Big)dt. \tag{14}$$

Definition 1. *(Passive Set) Consider the per-content per-frame MDP in (14), let $\mathcal{M}_n(W_k)$ be the set of states s for which the optimal action for content n in frame k is passive, i.e., not to cache content n in state s in frame k.*

Definition 2. *(Indexability) The per-content per-frame MDP in* (14) *for content n in frame k is indexable if the passive set $\mathcal{M}_n(W_k)$ increases with W_k, i.e., if $W_k > W_k'$, then $\mathcal{M}_n(W_k) \supseteq \mathcal{M}_n(W_k')$.*

Definition 3. *(Whittle Index) If the per-content per-item MDP for content n in frame k is indexable, then the Whittle index in state s is denoted as $W_{n,k}(s)$, and is given as follows:*

$$W_{n,k}(s) := \inf_{W_k\geq 0}\{s \in \mathcal{M}_n(W_k)\}, \tag{15}$$

which is the smallest value of the W_k such that the optimal policy for content n is indifferent towards $a=0$ and $a=1$ under state s in frame k.

Threshold Policies. We show that the optimal policy of (14) is of threshold-type, and then the MDP (14) is indexable. Based on these results, we explicitly derive the Whittle indices and the fluid Whittle index policy for the original problem (5).

Proposition 1. *For a fixed $W_k \geq 0$, $\forall k$, the optimal policy for the per-content per-frame MDP in* (14) *is of threshold type depending on W_k.*

We then compute the stationary distribution under a threshold policy as a function of the threshold.

Proposition 2. *The stationary distribution of the threshold policy $\pi_{n,k}=R$ satisfies $\phi_n^R(R')=0, \forall 0\leq R' < R$, and*

$$\phi_n^R(R) = 1/\left(1+\sum_{j=1}^{\infty}\left(\frac{\lambda_{n,k}}{\mu_{n,k}}\right)^j \frac{1}{\Pi_{k=1}^j(R+k)}\right), \tag{16}$$

$$\phi_n^R(R+l) = \left(\frac{\lambda_{n,k}}{\mu_{n,k}}\right)^l \frac{1}{\Pi_{k=1}^l(R+k)}\phi_n^R(R), l=1,2,\cdots.$$

For the notation abuse, we use a superscript R to denote the stationary distribution under a particular threshold policy R.

Proposition 3. *The MDP* (14) *is indexable.*

We are now ready to present the Whittle indices for (14).

Proposition 4. *For an indexable* (14) *with $\sum_{s=0}^R \phi_n^R(s)$ strictly increasing with R, the Whittle index is given by*

$$W_k(R) = \frac{\mathbb{E}_{R+1}[S_n]-\mathbb{E}_R[S_n]}{\sum_{S_n=0}^R \phi_n^R(S_n)-\sum_{S_n=0}^{R-1}\phi_n^{R-1}(S_n)}. \tag{17}$$

Since the cost function and the stationary probabilities are known, the Whittle indices (17) can be computed. Such an approach was first used in [45], [46] for queuing systems.

Fluid Whittle Index Policy. We now describe how the solutions to the LP (7)-(10) and relaxed problem (11) are used to obtain a policy for the original problem (5). It is clear that the optimal solutions to (7)-(10) and (11) are not always feasible for (5), since in the latter at most B_k content can be cached. If B_k is known, then we can follow the Whittle policy. However, in our problem, $B_k, \forall k$ are unknown and need to be solved from the LP (7)-(10). Hence, we refer to this as the *fluid Whittle index policy* as defined below.

Definition 4. *(Fluid Whittle Index Policy) At time t in frame k, the Fluid Whittle index policy prioritizes the content in the decreasing order of their Whittle indices $W_{n,k}(S_n(k,t))$ and caches the top $B_k^\star$ contents that have the largest index values.*

C. Asymptotic Optimality

Our *fluid Whittle Index Policy* achieves asymptotic optimality when the number of contents N and the cache dimensioning $B_k^\star$ go to infinity while holding $B_k^\star/N$ constant in any frame $k\in\mathcal{K}$. Hence we only present the main results in a particular frame k for the ease of exposition. This asymptotic regime is the same as Whittle [5] and many others [6], [7], [45], [49]. For the abuse of notation, we let the number of contents be ρN and the value of cache dimensioning be $\rho B_k^\star$ in the limit with $\rho\to\infty$. In other words, it represents the scenarios where there are N different classes of contents and each class contains ρ contents. Let B_k^{opt} be the optimal cache dimensioning for the original original problem (5) obtained by a genie-aided policy π_k^{opt}. Denote the corresponding cost as $C(\pi_k^{opt}, \rho B_k^{opt}, \rho N)$. Similarly, let $C(\pi_k, \rho B_k, \rho N)$ be the expected cost under a stationary policy π_k. Following the fact that the LP (7)-(10) is invariant with the scaling parameter ρ, the per-frame optimal cost of the fluid model(7)-(10) satisfies $\rho C_{fluid}(B_k^\star, N) \leq C(\pi_k^{opt}, \rho B_k^{opt}, \rho N)$.

Theorem 1. *Denote the fluid Whittle Index Policy under $B_k^\star$ as $\pi_k^\star$. Then, $\pi_k^\star$ achieves the asymptotic optimality as follows*

$$\lim_{\rho\to\infty}\frac{1}{\rho}\Big(C(\pi_k^\star,\rho B_k^\star,\rho N) - C(\pi_k^{opt},\rho B_k^{opt},\rho N)\Big)=0. \tag{18}$$

V. Reinforcement Learning Solutions

The knowledge of content request and delivery rates are needed for the computation of the *fluid Whittle index policy*. However, these parameters are often unknown and time-varying in cloud CDNs over different frames. Hence, we now adopt a learning perspective on top of the *fluid Whittle index policy*. We denote the resulting learning rule as *fW-UCB* and show that *fW-UCB* achieves an optimal sub-linear regret. Due to the decomposition nature of our problem across different frames as discussed in Section IV, we describe our proposed *fW-UCB* and its finite-time performance in a particular frame for the ease of readability, which applies to any frame $k\in\mathcal{K}$.

A. Algorithm Description

To simplify the exposition, we sample our continuous-time system at those discrete time-instants when the system state changes (see Section III). This results in a discrete-time MDP [4] and henceforth we will work exclusively with this discrete-time system. We adapt the upper confidence bound (UCB) [11] to our setting and design the *fW-UCB* policy as summarized in Algorithm 1. More precisely, *fW-UCB* have two phases: a planning phase and a policy execution phase. The planning phase focuses on defining a set of plausible MDPs [9] based on the number of visits to state-action pairs (s, a) and transitions tuples (s, a, s') as accurate as possible (Lines 1-5). We can define the corresponding *fluid Whittle index policy* by solving an optimistic planning problem, which is expressed as an LP (Lines 6-8). Our key contribution here is to choose the right value of $f(T_k)$ to balance the accuracy and complexity, which contributes to the sub-linear regret and low-complexity of *fW-UCB*. At the policy execution phase (Line 9), the derived *fluid Whittle index policy* is executed for the rest in frame k. Our key contribution here is to fully exploit our *fluid Whittle index policy*, instead of directly contending with the high-dimensional state-action space as in conventional RL algorithms. As a result, *fW-UCB* achieves a sub-linear regret and performs close to the offline optimum since our proposed *fluid Whittle index policy* is asymptotically optimal.

Optimistic Planning. In frame k, the CP first observes each content $n \in \mathcal{N}$ with a state-action pair (s, a) for $f(T_k)$ times (the value of $f(T_k)$ will be specified later). Since the transition is deterministic with $a = 0$, we only observe the transition under $a = 1$. We denote the number of times that a transition tuple $(s, 1, s')$ was observed within $f(T_k)$ as $T_n^k(s, 1, s') = \sum_{h=1}^{f(T_k)} \mathbb{1}(s_n(h+1) = s'|s_n(h) = s, a_n(h) = 1),\ \forall (s, 1, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$, where $s_n(h)$ represents the state for content n at time h, $a_n(h)$ is the corresponding action and $\mathcal{A} = \{0, 1\}$. Then the CP estimates the true transition probability $P_{n,k}(s'|s, 1)\ \forall (s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ by the corresponding empirical average as $\bar{P}_{n,k}(s'|s, 1) = T_n^k(s, 1, s')/f(T_k)$. It further defines the confidence interval such that the true transition probabilities lie in them with high probability. Formally, for any n at frame k, we define

$$\mathcal{P}_n^k(s,1) := \{\tilde{P}_{n,k}(s'|s,1) : |\tilde{P}_{n,k}(s'|s,1) - \bar{P}_{n,k}(s'|s,1)| \leq \delta_{n,k}\}, \tag{19}$$

where the size of the confidence interval $\delta_{n,k}$ is built using the empirical Hoeffding inequality [50], i.e., $\forall \eta \in (0, 1)$, $\delta_{n,k} = \sqrt{\frac{1}{2f(T_k)} \log\left(|\mathcal{S}|Nf(T_k)/\eta\right)}$. The set of plausible MDPs associated with the confidence intervals is then $\mathcal{M}_n^k = \{M_{n,k} = (\mathcal{S}, \tilde{P}_n) : \tilde{P}_{n,k}(\cdot|s, 1) \in \mathcal{P}_n^k(s, a)\}$. Then *fW-UCB* computes a policy $\pi_k^\star$ by performing optimistic planning. Given the set of plausible MDPs, it selects an optimistic transition function and an optimistic policy by solving a "modified LP", which is similar to the LP defined in (7)-(10), but with the transition functions replaced by $\tilde{P}(\cdot|\cdot, \cdot)$ in the confidence ball (19) since the corresponding true values are not available.

Algorithm 1 *fW-UCB* Policy

```
Require: Horizon $T_k$ in frame $\forall k$ and learning counts $f(T_k)$.
1: for $n = 1, 2, ..., N$ do
2:     Observing content $n$ until there are $f(T_k)$ visits of pairs $(s, 1)$, $\forall s \in \mathcal{S}$.
3: end for
4: Construct $\mathcal{P}_n^k(s, 1)$ according to (19);
5: Construct the plausible set of MDPs $\mathcal{M}_n^k, \forall n$;
6: Solve the extended LP in (20) to determine $\tilde{P}_{n,k}^\star(s'|s, 1)$ for all $s'$, $s$ according to (21) and $B_k^\star$;
7: Compute Whittle indices (17) based on $\tilde{P}_{n,k}(s'|s, a)$;
8: Establish the corresponding fluid Whittle index policy $\pi_k^\star$;
9: Execute $\pi_k^\star$ for the rest of the time in frame $k$.
```

The Extended LP. We cannot directly solve the "modified LP" since he true transition probabilities are unknown. To this end, we rewrite it as an extended LP problem by leveraging the state-action-state occupancy measures $z_{n,k}(s, a, s') := x_{n,k,s}^a \tilde{P}_{n,k}(s'|s, a)$ to express the confidence intervals of the transition probabilities. The extended LP over $z = \{z_{n,k}(s, a, s')\}$ is expressed as

$$\begin{aligned}
\min_{B_k, \{z_{n,k}\}} \quad & \kappa c_d \sum_{n=1}^{N} \sum_{(s,a,s')} s z_{n,k}(s, a, s') + (1-\kappa) c_b B_k \\
\text{s.t.} \quad & \sum_{n=1}^{N} \sum_{s,s' \in \mathcal{S}} z_{n,k}(s, 1, s') \leq B_k, \ \forall k, \\
& \sum_{a} \sum_{s'} z_{n,k}(s, a, s') = \sum_{a} \sum_{s'} z_{n,k}(s', a, s), \forall k, \\
& \frac{z_{n,k}(s, 1, s')}{\sum_y z_{n,k}(s, 1, y)} - (\hat{P}_{n,k}(s'|s, 1) + \delta_{n,k}) \leq 0, \\
& -\frac{z_{n,k}(s, 1, s')}{\sum_y z_{n,k}(s, 1, y)} + (\hat{P}_{n,k}(s'|s, 1) - \delta_{n,k}) \leq 0.
\end{aligned} \tag{20}$$

This approach was also used for adversarial and constrained MDPs [51]–[53]. Once we compute the optimal $z_{n,k}^\star$ and $B_k^\star$, the transition probabilities are recovered by

$$\tilde{P}_{n,k}^\star(s'|s, 1) = \frac{z_{n,k}^\star(s, 1, s')}{\sum_y z_{n,k}^\star(s, 1, y)}. \tag{21}$$

Finally, using this transition probabilities, we can compute the Whittle indices in (17) for all states of all contents, and then define the *fluid Whittle index policy* as in Definition 4, and execute this policy for the rest of time in frame k.

B. The Learning Regret

We evaluate the efficiency of *fW-UCB* policy using *regret*, which is defined as the expected gap between the cost obtained by the offline optimum, i.e., the genie-aided policy with full knowledge of all transition probabilities, and that of *fW-UCB*. Denote the cumulative cost under policy π_k as $C(\pi_k, T_k) := \kappa c_d \sum_{t=1}^{T_k} \sum_{n=1}^{N} S_n(t)$, which is a random variable. Then the expected average cost under policy π_k satisfies $\gamma_k := \lim_{T_k \to \infty} \frac{1}{T_k} \mathbb{E}_{\pi_k}[C(\pi_k, T_k)]$, and the optimal

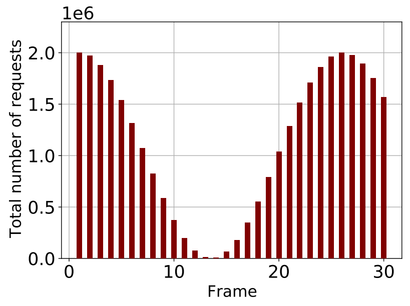

Fig. 2: Number of requests per frame.

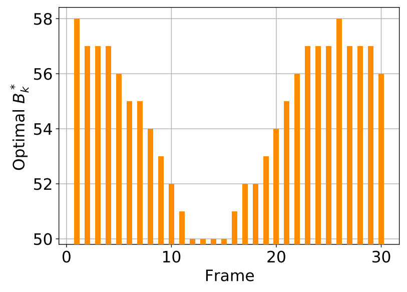

Fig. 3: Cache dimensioning per frame.

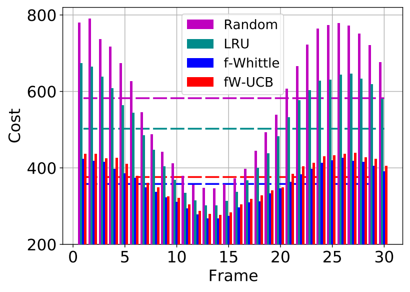

Fig. 4: Comparison of costs.

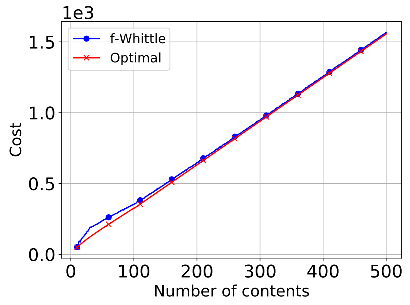

Fig. 5: Asymptotic optimality.

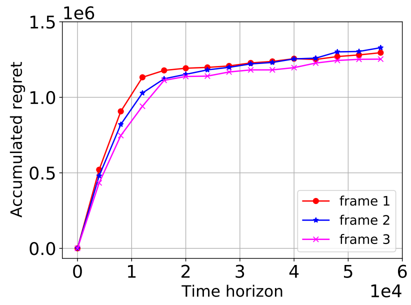

Fig. 6: Accumulated regret.

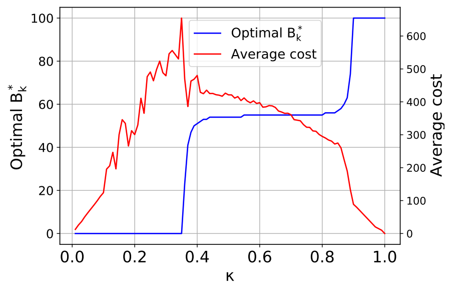

Fig. 7: Impact of tradeoff parameter κ.

average reward is $\gamma_k^{opt} := \sup_{\pi_k} \gamma_k$. Then the regret of π_k is defined as $\Delta(T_k) := T_k \gamma_k^{opt} - \mathbb{E}_{\pi_k}[C(\pi_k, T_k)]$.

Theorem 2. *The regret of fW-UCB policy satisfies*

$$\Delta(T_k) = \mathcal{O}((|\mathcal{S}|N + 2N(1+\eta))\sqrt{T_k}). \tag{22}$$

Proof Outline: Since there are two phases in *fW-UCB*, we decompose the regret in two parts as $\Delta(T_k) = \Delta(T_1) + \Delta(T_2)$, where $\Delta(T_1)$ is the regret for the planning phase with any random policy and $\Delta(T_2)$ is the regret for the policy execution phase, and $T_2 = T_k - T_1$. The regret of the planning phase is upper bounded by $\mathcal{O}(|\mathcal{S}|N\sqrt{T_k})$. The regret of the policy execution phase is caused by either "failure event" (confidence ball fails) or "good event" (true MDP is within confidence ball). We show that they are bounded by $\mathcal{O}(2N\eta\sqrt{T_k})$ and $\mathcal{O}(2N\sqrt{T_k})$, respectively. Combining them completes the proof. Please see Appendix B for the detailed proof.

Remark 2. *Although fW-UCB is a non-episodic algorithm in each frame k, it still achieves the $\mathcal{O}(\sqrt{T_k})$ regret no worse than the episodic UCRL2. Specifically, the proposed fW-UCB spends no time for searching a better MDP instance. Instead, it constructs a upper confidence ball for all plausible MDPs. Then, it determines the optimal policy by calculating the extended LP* (20) *for only once, which significantly reduces the exponential implementation complexity compared with UCRL2, colored-UCRL2 [30] to a linear scale.*

VI. Numerical Results

In this section, we numerically evaluate the performance of our proposed *fluid Whittle index policy* (f-Whittle) and *fW-UCB* using both synthetic and real traces. We compare to the widely used Least Recently Used (LRU) and Random policies. Since LRU and Random policies are designed for a fixed cache size and do not account for cache dimensioning, we evaluate them using a fixed cache size over all frames that have the same total dimemsioning cost as our policies.

A. Evaluation Using Synthetic Trace

We simulate a system with $N = 100$ contents over $K = 30$ frames. The requests in each frame are selected randomly following a Zipf distribution with a delivery rate of 0.1. The Zipf parameter is fixed to be 0.6 but the total number of requests varies across frames, as shown in Figure 2. We use $\kappa = 0.5$, $c_d = 1$ and $c_b = 10$. The leased storage in each frame is shown in Figure 3. The per-frame cost is presented in Figure 4, with the dashed lines representing the average cost over all frames. It is evident from Figure 3 that dynamic cache dimensioning can adaptively tune the leased storage to meet the time-varying trends of content requests, as it is common in cloud CDNs. This results in a much smaller cost than conventional policies as shown in Figure 4. These results reflect the intelligence of our proposed policies: when the number of requests is low, the CP can lease less cloud storage with a smaller total cost. We further note that our learning policy *fW-UCB* can almost achieve the same performance as the f-Whittle, which is consistent with our theoretical results.

We further validate the asymptotic optimality of our proposed *fluid Whittle index policy* (f-Whittle) (see Theorem 1) and the sub-linear regret of *fW-UCB* (see Theorem 2). Due to the decomposition nature of our problem, we use the requests from a frame randomly selected in Figure 2. Similar trends hold for all other frames and hence are omitted. We consider a similar setting as above. The average cost obtained by our f-Whittle and the theoretical upper bound achieved by solving the LP in (7)-(10) is shown in Figure 5. It is clear that f-Whittle achieves asymptotic optimality when the number of

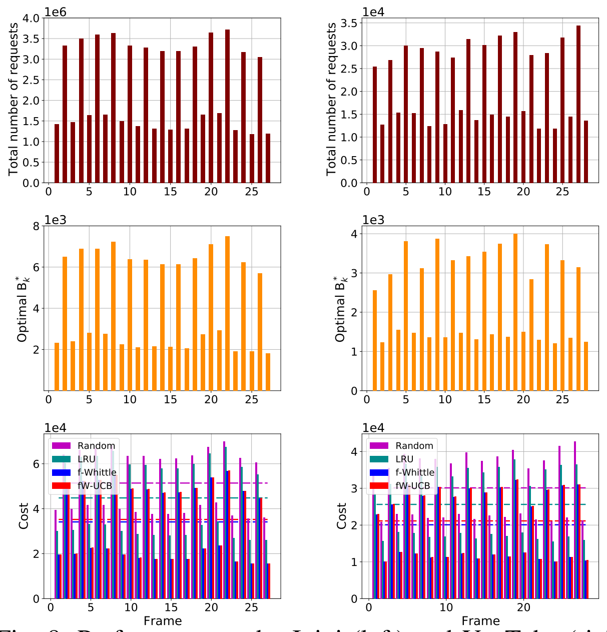

Fig. 8: Performance under Iqiyi (left) and YouTube (right).

contents increases. The learning regrets of *fW-UCB* in three randomly selected frames are shown in Figure 6, where we use the Monte Carlo simulation with $10,000$ independent trails. Finally, we investigate the impact of the system-wide parameter κ. The blue curve in Figure 7 presents the optimal cache dimensioning vs. κ in one frame and the red curve is the corresponding average cost. A key takeaway from Figure 7 is that the variation of cache dimensioning and content delivery latency costs as κ goes from 0 to 1. When κ is small, our problem (5) weighs more on minimizing cache dimensioning costs, hence a smaller leased storage is preferred. For example, when $\kappa < 0.35$, no leasing is the optimal decision. On the other hand, when κ is large, our problem tends to minimize the content delivery latency costs, and hence a larger leased storage is beneficial. This provides a tunable knob that can be used by the network operator to balance the dimensioning and latency costs for applications with different dimensioning-latency tradeoff requirement.

B. Evaluation Using Real Trace

We further evaluate our proposed policies for dynamic cache dimensioning using two real CDN traces: (i) *Iqiyi* [54], which contains mobile video behaviors; and (i) *YouTube* [55], which contains trace data about user requests for specific YouTube content collected from a campus network. For the Iqiyi (resp. YouTube) trace, there are more than 67 (resp. 0.6) million requests for more than 1.4 million (resp. 0.3) unique contents over a period of 335 (resp. 336) hours. To this end, we consider the cache dimensioning occurring every 1, 3, 6, 12, and 24 hours. Due to space constraints, we only present results for 12 hours and similar trends hold for other cases. The number of requests, the cache dimensioning and cost in each frame are shown in Figure 8. Again, it is clear that our dynamic cache dimensioning is able to tune the leased storage to follow the variations of content requests in real systems. As a result, our proposed policies significantly outperform conventional policies with a smaller cost. Finally, we note that *fW-UCB* can quickly learn the system dynamics and perform close to f-Whittle, which matches well with our theoretical results.

VII. Conclusion

We studied the problem of dynamic cache dimensioning in the cloud to minimize the total average costs for both storage and content delivery latency. Though it can be posed as an MDP, it is hard to solve due to the curse of dimensionality. To this end, we proposed a fluid Whittle index policy which is provably asymptotically optimal. Since the system parameters are often unknown and time-varying in the cloud, we proposed an RL algorithm entitled *fW-UCB* that fully exploit the structure of our index policy and hence is lightweight. We showed that it has an optimal sub-linear regret. Extensive simulations using real traces demonstrated the significant gains of our proposed policies over conventional ones.

Appendix A
Proof Sketches on Fluid Whittle Index Policy

Proof Sketch of Proposition 1: Denote $R^* = \max\{S : A_n^{\pi^*}(S) = 0\}$, where $A_n^{\pi^*}(S)$ is the action at state S for content n under optimal policy π^*. By definition, $A_n^{\pi^*} = 1$, $\forall S_n > R^*$. To prove the threshold policy, we need to show $A_n^{\pi^*}(S_n) = 0$, $\forall S_n < R^*$. Denote $\phi_n^{\pi^*}$ the stationary distribution under π^* and assume there exists at least one state S that $A_n^{\pi'}(S_n) = 1$. Consider another policy π' which keeps all actions for states other than $S_n = R^*+1$ same and let $A_n^{\pi'}(R^*+1) = 0$, we can prove that $\sum_{S_n=0}^{R^*+1}\phi_n^{\pi'}(S_n)\mathbb{1}_{\{A_n^{\pi'}(S_n)=0\}} \geq \sum_{S_n=0}^{R^*}\phi_n^{\pi^*}(S_n)\mathbb{1}_{\{A_n^{\pi^*}(S_n)=0\}}$, i.e., π' achieves a lower average cost than $\pi^\star$. This is a contradiction with the fact that $\pi^\star$ is optimal. Thus, we have $\sum_{S_n=0}^{R^*-1} S_n\phi_n^{\pi^*}(S_n) = 0$, indicating the threshold policy with threshold R^*.

Proof Sketch of Proposition 2: This is straightforward by using the threshold property and transitions (1).

Proof Sketch of Proposition 3: Since the optimal policy for (14) is a threshold policy, for a given W_k, the optimal average cost under threshold R is concave non-increasing and hence $\mathcal{M}(W_k) \subseteq \mathcal{M}(W_k')$ if $W_k < W_k'$. Next we show that $\sum_{s=0}^R \phi_i^R(s)$ is strictly increasing in R using Proposition 2.

Proof Sketch of Proposition 4: This follows from Whittle index definition that the performance of a policy with threshold R equals to that of a policy with threshold $R+1$, and the stationary property under threshold policy R. See [45], [46].

Proof Sketch of Theorem 1: From the definition of π_k^{opt}, for policy $\pi_k^\star$ under $B_k^\star$, we have $\lim_{\rho\to\infty}\frac{1}{\rho}C(\pi_k^\star, \rho B_k^\star, \rho N) \geq \lim_{\rho\to\infty}\frac{1}{\rho}C(\pi_k^{opt}, \rho B_t^{opt}, \rho N)$. Hence it suffices to prove that $\lim_{\rho\to\infty}\frac{1}{\rho}C(\pi_k^\star, \rho B_k^\star, \rho N) \leq \lim_{\rho\to\infty}\frac{1}{\rho}C(\pi_k^{opt}, \rho B_t^{opt}, \rho N)$. Let $D_{n,k}(s)$ be the average number of class-n contents under state s in frame k under the stationary policy $\pi_k^\star$. Following [49], when $\rho N \to \infty$ and $B_k^\star/N$ is a constant, $\lim_{\rho\to\infty} D_{n,k}(s)/\rho = x_{n,k,s}, \forall n, k$. Therefore, we have

$$\lim_{\rho\to\infty} C(\pi_k^\star, \rho B_k^\star, \rho N)/\rho$$

$$= \lim_{\rho\to\infty} \frac{1}{\rho}\mathbb{E}_{\pi_k^\star}\left[\kappa c_d \limsup_{T_k\to\infty} \sum_{n=1}^{\rho N} \frac{1}{T_k}\int_{t=1}^{T_k} S_n(k,t)dt\right] + (1-\kappa)c_b B_k^\star$$

$$\overset{(a)}{=} \lim_{\rho\to\infty} \kappa c_d \sum_{n=1}^{N}\sum_{s} sD_{n,k}(s)/\rho + (1-\kappa)c_b B_k^\star$$

$$\overset{(b)}{=} \kappa c_d \sum_{n=1}^{N}\sum_{s} s x_{n,k,s}^\star + (1-\kappa)c_b B_k^\star = C_{fluid}(B_k^\star, N)$$

$$\overset{(c)}{\le} \lim_{\rho\to\infty} C(\pi_n^{opt}, \rho B_k^{opt}, \rho N)/\rho,$$

where (a) follows from definition of $D_{n,k}(s)$, (b) holds since $\lim_{\rho\to\infty} D_{n,k}(s)/\rho = x_{n,k,s}$, and (c) is due to definition.

Appendix B
Proof of Theorem 2

The Regret of the Planning Phase. In the planning phase, each state-action pair $(s,1)$ for each content is randomly sampled for $f(T_k)$ times. The performance gap between the genie-aided policy and the random policy for each time is bounded because the cost is bounded, and hence we have

Lemma 1. *Since the reward is bounded, the regret in the planning phase can be bounded by $\Delta(T_1) = \mathcal{O}(|\mathcal{S}|Nf(T_k))$.*

Proof. The result follows the fact that there are N contents with a total S state-action pairs and to guarantee each state-action pair being sampled for $f(T_k)$ times. □

The Regret of the Policy Execution Phase. We next analyze the regret of the policy execution phase, i.e., $\Delta(T_2)$, which is defined as $\Delta(T_2) := T_2\gamma_k^{opt} - \mathbb{E}[C(\pi_k^\star, T_2)]$, which characterizes the accumulated cost gap when the true MDP employs the optimal policy π^{opt} and the learned policy $\pi_k^\star$, respectively. For the entire parameter space, two possible events can occur at the policy execution phase, which separates the regret into two disjoint parts. Next we bound these two parts separately.

The first event is called *the failure event*, which occurs when the true MDP $M_k := \{M_{n,k}, \forall n\}$ lies outside the plausible MDPs set $\mathcal{M}^k := \{\mathcal{M}_n^k, \forall n\}$ (see definition in (19)), and the second is *the good event* when true MDP M_k lies inside the plausible MDPs set $\mathcal{M}^k$. Therefore, the regret of the policy execution phase can be decomposed into two parts as follows

$$\Delta(T_2) = \Delta(T_2)\mathbb{1}(M_k \notin \mathcal{M}^k) + \Delta(T_2)\mathbb{1}(M_k \in \mathcal{M}^k).$$

Regret Conditioned on the Failure Event. Define the failure event as $\mathcal{E}_p := \{\exists s, n, |P_{n,k}(s'|s,1) - \bar{P}_{n,k}(s'|s,1)| > \delta_{n,k}\}$, which means that the true parameters are outside the confidence interval constructed in (19). The associated complementary event is denoted as $\mathcal{E}_p^c$. Therefore, we have the following relations: $\{M_k \notin \mathcal{M}^k\} := \mathcal{E}_p$, and $\{M_k \in \mathcal{M}^k\} := \mathcal{E}_p^c$. We now characterize the probability that the failure event occurs.

Lemma 2. *With $\delta_{n,k} = \sqrt{\frac{1}{2f(T_k)}\log\left(|\mathcal{S}|Nf(T_k)/\eta\right)}$, we have $\mathbb{P}(M_k \notin \mathcal{M}^k) \le 2\eta/f(T_k)$.*

Proof. By Chernoff-Hoeffding inequality [50], we have

$$\mathbb{P}\big(|P_{n,k}(s'|s,1) - \hat{P}_{n,k}(s'|s,1)| > \delta_{n,k}\big) \le 2\eta/|\mathcal{S}|Nf(T_k).$$

By leveraging union bound over all states, actions, number of arms, we have $\mathbb{P}(M_k \notin \mathcal{M}^k) \le \sum_{n=1}^N \sum_s \mathbb{P}\big(|P_{n,k}(s'|s,1) - \bar{P}_{n,k}(s'|s,1)| > \delta_{n,k}\big) \le 2\eta/f(T_k)$. □

Lemma 3. *The regret conditioned on the failure event is*

$$\Delta(T_2)\mathbb{1}(M_k \notin \mathcal{M}^k) = \mathcal{O}\Big(2NT_2\eta/f(T_k)\Big).$$

Proof. From Lemma 2, we have $\Delta(T_2)\mathbb{1}(M_k \notin \mathcal{M}^k) = \mathcal{O}(NT_2\mathbb{1}(M_k \notin \mathcal{M}^k)) = \mathcal{O}(2NT_2\eta/f(T_k))$, where the first equality is due to that $\Delta(T_2)$ is bounded by $\mathcal{O}(NT_2)$. □

Regret Conditioned on the Good Event. From Lemma 5, we have that the true MDP is inside the plausible MDPs set, i.e., $M_k \in \mathcal{M}^k$, with probability at least $1 - 2\eta/f(T_k)$. Now we consider the regret conditioned on the good event $M_k \in \mathcal{M}^k$. Define γ_k^{opt} as the optimal average cost achieved by the optimal policy π_k^{opt} and $\gamma_k^\star$ as optimistic average reward achieved by the learned policy $\pi_k^\star$ for the ture MDP M_k. Then

$$\Delta(T_2)\mathbb{1}(M_k \in \mathcal{M}^k) = T_2\gamma_k^\star - T_2\gamma_k^{opt}.$$

Before showing the regret conditioned on the good event, we first present a key lemma.

Lemma 4. *(Optimism) Conditioned on the good event, there exists a transition $\tilde{P}_{n,k} \in \mathcal{P}_{n,k}, \forall n$ such that $\sum_{t=1}^{T_2}\sum_{n=1}^{N}\sum_s \phi_n^{opt}(s)s \ge \sum_{t=1}^{T_2}\sum_{n=1}^{N}\sum_s \tilde{\phi}_n(s)s$, where $\tilde{\phi}_n$ is the stationary distribution derived from $\{\tilde{P}_{n,k}, \forall n\}$.*

Proof. The true $P_{n,k}$ is contained in $\mathcal{P}_{n,k}, \forall n$ for a good event. The result directly comes from the fact that confidence interval expands the feasibility region of original problem. □

Remark 3. *Lemma 4 indicates that inside the plausible MDPs set $\mathcal{M}^k$, there exists an MDP $\tilde{M}_k$ with parameters $\{\tilde{P}_{n,k}\}$ achieving no more accumulated cost compared to the cost achieved by optimal policy for the true MDP M_k.*

Lemma 5. *The regret conditioned on the good event is*

$$\Delta(T_2)\mathbb{1}(M_k \in \mathcal{M}^k) = \mathcal{O}(2N\sqrt{T_k}).$$

Proof. Based on Remark 2, we have

$$\Delta(T_2) = T_2\gamma_k^\star - T_2\gamma_k^{opt} \le T_2\gamma_k^\star - T_2\tilde{\gamma}_k$$

$$= T_2\sum_{n=1}^{N}\sum_{s}\phi_n^\star(s)s - T_2\sum_{n=1}^{N}\sum_{s}\tilde{\phi}_n(s)(s-\delta_{n,k})$$

$$\overset{(a)}{\le} \underbrace{T_2 S_{\max}\sum_{n=1}^{N}\sum_{s}(\phi_n^\star(s) - \tilde{\phi}_n(s))}_{\text{term 1}} + \underbrace{T_2\sum_{n=1}^{N}\sum_{s}\tilde{\phi}_n(s)\delta_{n,k}}_{\text{term 2}}$$

$$\overset{(b)}{\le} \mathcal{O}(2NT_2/f(T_k)),$$

where (a) follows the fact that s is bounded by $S_{\max}$. Since $\tilde{\phi}_n$ and $\phi_n^\star$ are probability distributions, we have term $1 = 0$ and term $2 \le NT_2\delta_{n,k}$. (b) is based on the definition of $\delta_{n,k}$. □

Total Regret. Combining Lemma 1, 3 and 5 and let $f(T_k) = \sqrt{T_k}$, the total regret is given by

$$\Delta(T_k) = \Delta(T_1) + \Delta(T_2) = \mathcal{O}((|\mathcal{S}|N + 2N(1+\eta))\sqrt{T_k}).$$

References

[1] G. Forecast *et al.*, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022," *Update*, 2019.

[2] "Amazon AWS," https://aws.amazon.com/.

[3] "Amazon Cloudfront," https://aws.amazon.com/.

[4] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

[5] P. Whittle, "Restless Bandits: Activity Allocation in A Changing World," *Journal of Applied Probability*, pp. 287–298, 1988.

[6] J. Gittins, K. Glazebrook, and R. Weber, *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons, 2011.

[7] I. M. Verloop, "Asymptotically Optimal Priority Policies for Indexable and Nonindexable Restless Bandits," *The Annals of Applied Probability*, vol. 26, no. 4, pp. 1947–1995, 2016.

[8] J. Niño-Mora, "Dynamic Priority Allocation via Restless Bandit Marginal Productivity Indices," *Top*, vol. 15, no. 2, pp. 161–198, 2007.

[9] T. Jaksch, R. Ortner, and P. Auer, "Near-Optimal Regret Bounds for Reinforcement Learning," *Journal of Machine Learning Research*, vol. 11, no. 4, 2010.

[10] A. Gopalan and S. Mannor, "Thompson Sampling for Learning Parameterized Markov Decision Processes," in *Proc. of COLT*, 2015.

[11] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-Time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, 2002.

[12] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib, "A Survey on Replica Server Placement Algorithms for Content Delivery Networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 1002–1026, 2016.

[13] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The Role of Caching in Future Communication Systems and Networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 6, pp. 1111–1125, 2018.

[14] D. Carra, G. Neglia, and P. Michiardi, "Elastic Provisioning of Cloud Caches: A Cost-Aware TTL Approach," *IEEE/ACM Transactions on Networking*, vol. 28, no. 3, pp. 1283–1296, 2020.

[15] J. Kwak, G. Paschos, and G. Iosifidis, "Dynamic Cache Rental and Content Caching in Elastic Wireless CDNs," in *Proc. of IEEE WiOpt*, 2018.

[16] A. Cidon, A. Eisenman, M. Alizadeh, and S. Katti, "Dynacache: Dynamic Cloud Caching," in *Proc. of USENIX HotCloud*, 2015.

[17] K. Liu and Q. Zhao, "Indexability of Restless Bandit Problems and Optimality of Whittle Index for Dynamic Multichannel Access," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5547–5567, 2010.

[18] W. Dai, Y. Gai, B. Krishnamachari, and Q. Zhao, "The Non-Bayesian Restless Multi-Armed Bandit: A Case of Near-Logarithmic Regret," in *Proc. of IEEE ICASSP*, 2011.

[19] S.-P. Sheng, M. Liu, and R. Saigal, "Data-Driven Channel Modeling Using Spectrum Measurement," *IEEE Transactions on Mobile Computing*, vol. 14, no. 9, pp. 1794–1805, 2014.

[20] V. S. Borkar, K. Ravikumar, and K. Saboo, "An Index Policy for Dynamic Pricing in Cloud Computing under Price Commitments," *Applicationes Mathematicae*, vol. 44, pp. 215–245, 2017.

[21] T. W. Archibald, D. Black, and K. D. Glazebrook, "Indexability and Index Heuristics for A Simple Class of Inventory Routing Problems," *Operations Research*, vol. 57, no. 2, pp. 314–326, 2009.

[22] C. H. Papadimitriou and J. N. Tsitsiklis, "The Complexity of Optimal Queueing Network Control," in *Proc. of IEEE Conference on Structure in Complexity Theory*, 1994.

[23] J. Nino-Mora, "Restless Bandits, Partial Conservation Laws and Indexability," *Advances in Applied Probability*, pp. 76–98, 2001.

[24] D. Bertsimas and J. Niño-Mora, "Restless Bandits, Linear Programming Relaxations, and A Primal-Dual Index Heuristic," *Operations Research*, vol. 48, no. 1, pp. 80–90, 2000.

[25] W. Hu and P. Frazier, "An Asymptotically Optimal Index Policy for Finite-Horizon Restless Bandits," *arXiv preprint arXiv:1707.00205*, 2017.

[26] G. Zayas-Cabán, S. Jasin, and G. Wang, "An Asymptotically Optimal Heuristic for General Nonstationary Finite-Horizon Restless Multi-Armed, Multi-Action Bandits," *Advances in Applied Probability*, vol. 51, no. 3, pp. 745–772, 2019.

[27] H. Liu, K. Liu, and Q. Zhao, "Logarithmic Weak Regret of Non-Bayesian Restless Multi-Armed Bandit," in *Proc of IEEE ICASSP*, 2011.

[28] ——, "Learning in A Changing World: Restless Multi-Armed Bandit with Unknown Dynamics," *IEEE Transactions on Information Theory*, vol. 59, no. 3, pp. 1902–1916, 2012.

[29] C. Tekin and M. Liu, "Online Learning of Rested and Restless Bandits," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5588–5611, 2012.

[30] R. Ortner, D. Ryabko, P. Auer, and R. Munos, "Regret Bounds for Restless Markov Bandits," in *Proc. of ALT*, 2012.

[31] Y. H. Jung and A. Tewari, "Regret Bounds for Thompson Sampling in Episodic Restless Bandit Problems," *Proc. of NeurIPS*, 2019.

[32] Y. H. Jung, M. Abeille, and A. Tewari, "Thompson Sampling in Non-Episodic Restless Bandits," *arXiv preprint arXiv:1910.05654*, 2019.

[33] S. Wang, L. Huang, and J. Lui, "Restless-UCB, an Efficient and Low-complexity Algorithm for Online Restless Bandits," in *Proc. of NeurIPS*, 2020.

[34] C. Tekin and M. Liu, "Adaptive Learning of Uncontrolled Restless Bandits with Logarithmic Regret," in *Proc. of Allerton*, 2011.

[35] V. S. Borkar and K. Chadha, "A Reinforcement Learning Algorithm for Restless Bandits," in *Proc. of Indian Control Conference*, 2018.

[36] K. Avrachenkov and V. S. Borkar, "A Learning Algorithm for the Whittle Index Policy for Scheduling Web Crawlers," in *Proc. of Allerton*, 2019.

[37] J. Fu, Y. Nazarathy, S. Moka, and P. G. Taylor, "Towards Q-Learning the Whittle Index for Restless Bandits," in *Proc. of Australian & New Zealand Control Conference*, 2019.

[38] G. Xiong, J. Li, and R. Singh, "Reinforcement Learning Augmented Asymptotically Optimal Index Policies for Finite-Horizon Restless Bandits," in *Proc. of AAAI*, 2022.

[39] S. Ioannidis and E. Yeh, "Adaptive Caching Networks with Optimality Guarantees," *Proc. of ACM SIGMETRICS*, 2016.

[40] J. Li, T. K. Phan, W. K. Chai, D. Tuncer, G. Pavlou, D. Griffin, and M. Rio, "DR-Cache: Distributed Resilient Caching with Latency Guarantees," in *Proc. of IEEE INFOCOM*, 2018.

[41] M. Mahdian, A. Moharrer, S. Ioannidis, and E. Yeh, "Kelly Cache Networks," in *Proc. of IEEE INFOCOM*, 2019.

[42] N. K. Panigrahy, J. Li, D. Towsley, and C. V. Hollot, "Network Cache Design under Stationary Requests: Exact Analysis and Poisson Approximation," *Computer Networks*, vol. 180, p. 107379, 2020.

[43] N. K. Panigrahy, J. Li, F. Zafari, D. Towsley, and P. Yu, "A TTL-based Approach for Content Placement in Edge Networks," in *Proc. of VALUETOOLS*, 2021.

[44] F. Baccelli and P. Brémaud, *Elements of Queueing Theory: Palm Martingale Calculus and Stochastic Recurrences*. Springer Science & Business Media, 2013, vol. 26.

[45] M. Larrañaga, U. Ayesta, and I. M. Verloop, "Index Policies for A Multi-Class Queue with Convex Holding Cost and Abandonments," in *Proc. of ACM SIGMETRICS*, 2014.

[46] M. Larrnaaga, U. Ayesta, and I. M. Verloop, "Dynamic Control of Birth-and-Death Restless Bandits: Application to Resource-Allocation Problems," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3812–3825, 2016.

[47] Y.-P. Hsu, "Age of Information: Whittle Index for Scheduling Stochastic Arrivals," in *Proc. of IEEE ISIT*, 2018.

[48] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT press, 2018.

[49] R. R. Weber and G. Weiss, "On An Index Policy for Restless Bandits," *Journal of Applied Probability*, pp. 637–648, 1990.

[50] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[51] C. Jin, T. Jin, H. Luo, S. Sra, and T. Yu, "Learning Adversarial MDPs with Bandit Feedback and Unknown Transition," *arXiv preprint arXiv:1912.01192*, 2019.

[52] A. Rosenberg and Y. Mansour, "Online Convex Optimization in Adversarial Markov Decision Processes," in *Proc. of ICML*, 2019.

[53] K. C. Kalagarla, R. Jain, and P. Nuzzo, "A Sample-Efficient Algorithm for Episodic Finite-Horizon MDP with Constraints," in *Proc. of AAAI*, 2021.

[54] G. Ma, Z. Wang, M. Zhang, J. Ye, M. Chen, and W. Zhu, "Understanding Performance of Edge Content Caching for Mobile Video Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 5, pp. 1076–1089, 2017.

[55] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Watch Global, Cache Local: YouTube Network Traffic at A Campus Network: Measurements and Implications," in *Multimedia Computing and Networking*, 2008.