Serving Content with Unknown Demand: the High-Dimensional Regime

Sharayu Mohharir
Department of ECE
University of Texas at Austin
Austin, TX 78712
sharayu.moharir@gmail.com

Javad Ghaderi
Department of ECE
University of Texas at Austin
Austin, TX 78712
jghaderi@ee.columbia.edu

Sujay Sanghavi
Department of ECE
University of Texas at Austin
Austin, TX 78712
sanghavi@mail.utexas.edu

Sanjay Shakkottai
Department of ECE
University of Texas at Austin
Austin, TX 78712
shakkott@austin.utexas.edu

ABSTRACT

In this paper we look at content placement in the high-dimensional regime: there are n servers, and $\mathrm{O}(n)$ distinct types of content. Each server can store and serve $\mathrm{O}(1)$ types at any given time. Demands for these content types arrive, and have to be served in an online fashion; over time, there are a total of $\mathrm{O}(n)$ of these demands. We consider the algorithmic task of content placement: determining which types of content should be on which server at any given time, in the setting where the demand statistics (i.e. the relative popularity of each type of content) are not known a-priori, but have to be inferred from the very demands we are trying to satisfy. This is the high-dimensional regime because this scaling (everything being $\mathrm{O}(n)$) prevents consistent estimation of demand statistics; it models many modern settings where large numbers of users, servers and videos/webpages interact in this way.

We characterize the performance of *any* scheme that separates learning and placement (i.e. which use a portion of the demands to gain some estimate of the demand statistics, and then uses the same for the remaining demands), showing it is order-wise strictly suboptimal. We then study a simple adaptive scheme - which myopically attempts to store the most recently requested content on idle servers - and show it outperforms schemes that separate learning and placement. Our results also generalize to the setting where the demand statistics change with time. Overall, our results demonstrate that separating the estimation of demand, and the subsequent use of the same, is strictly suboptimal.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Content Delivery Systems; Large Scale Systems; Content Replication Strategies; Matchings

1. INTRODUCTION

Ever increasing volumes of multimedia content is now requested and delivered over the Internet. Content delivery systems (e.g., YouTube [23]), consisting of a large collection of servers (each with limited storage/service capability), process and service these requests. Naturally, the storage and content replication strategy (i.e., what content should be stored on each of these servers) forms an important part of the service and storage architecture.

Two trends have emerged in such settings of large-scale distributed content delivery systems. First, there has been a sharp rise in not just the volume of data, but indeed in *the number of content-types* (e.g., number of distinct YouTube videos) that are delivered to users [23]. Second, the popularity and demand for most of this content is *uneven and ephemeral*; in many cases, a particular content-type (e.g., a specific video clip) becomes popular for a small interval of time after which the demand disappears; further a large fraction of the content-types languish in the shadows with almost no demand [1, 5].

To understand the effect of these trends, we study a stylized model for the content placement and delivery in large-scale distributed content delivery systems. The system consists of n servers, each with constant storage and service capacities, and αn content-types (α is some constant number). We consider the scaling where the system size n tends to infinity. The requests for the content-types arrive dynamically over time and need to be served in an online manner by the free servers storing the corresponding contents. The requests that are "deferred" (i.e., cannot be immediately served by a free server with requested content-type) incur a high cost. To ensure reliability, we assume that there are alternate server resources (e.g., a central server with large enough backup storage and service capacity, or additional servers that can be freed up on-demand) that can serve such deferred requests.

The performance of any content placement strategy crucially depends on the popularity distribution of the content. Empirical studies in many services such as YouTube, Peer-to-Peer (P2P) VoD systems, various large video streaming systems, and web caching, [2,5,7,17,24] have shown that access for different content-types is very inhomogeneous and typically matches well with power-law (Zipf-like) distributions, i.e., the request rate for the i-th most popular content-type is proportional to $i^{-\beta}$, for some parameter $\beta > 0$. For the performance analysis, we assume that the content-types have a popularity that is governed by some power-law distribution with unknown β and further this distribution changes over time.

Our objective is to provide efficient content placement strategies that minimize the number of requests deferred. It is natural to expect that content placement strategies in which more popular content-types are replicated more will have a good performance. However, there is still a lot of flexibility in designing such strategies and the extent of replication of each content-type has to be determined. Moreover, the requests arrive dynamically over time and popularities of different content-types might vary significantly over time; thus the content placement strategy needs to be online and robust.

The fact that the number of contents is very large and their popularities are time-varying creates two new challenges that are not present in traditional queueing systems. First, it is imperative to *measure the performance of content replication strategies over the time scale in which changes in popularities occur*. In particular, the steady-state metrics typically used in queueing systems are not a right measure of performance in this context. Second, the number of content-types is enormous and *learning the popularities of all content-types over the time scale of interest is infeasible*. This is in contrast with traditional multi-class multi-server systems where the number of demand classes does not scale with the number of servers (low-dimensional setting) and thus learning the demand rates can be done in a time duration that does not scale with the system size.

1.1 Contributions

The main contributions of our work can be summarized as follows.

High dimensional vs. low dimensional: We consider the high dimensional regime where the number of servers, the number of content-types, and the number of requests to be served over any time interval all scale as $\mathrm{O}(n)$; further the demand statistics are not known a-priori. This scaling means that consistent estimation of demand statistics is not possible. We show that a "*learn-and-optimize*" approach, namely, learning the demand statistics based on requests and then locally caching content on servers according to this empirical statistics, is strictly sub-optimal (even when using high-dimensional estimators such as the Good-Turing estimator [13]). This is in contrast to the conventional low-dimensional setting (finite number of content-types) where the "learn-and-optimize" approach is asymptotically optimal.

Adaptive vs. learn-and-optimize: We study an adaptive content replication strategy which myopically attempts to cache the most recently requested content-types on idle servers. Our key result is that even this simple adaptive strategy strictly outperforms *any* content placement strategy based on the "learn-and-optimize" approach. Our results also generalize to the setting where the demand statistics change with time.

Overall, our results demonstrate that separating the estimation of demands and the subsequent use of the estimations to design optimal content placement policies is deprecated in the high-dimensional setting.

1.2 Organization and Basic Notations

The rest of this paper is organized as follows. We describe our system model and setting in Section 2. The main results are presented in Section 3. Our simulation results are discussed in Section 4. Section 5 contains the proofs of some of our key results. Section 7 gives an overview of related works. We finally end the paper with conclusions. The rest of the proofs can be found in the Appendix.

Some of the basic notations are as follows. Given two functions f and g, we write $f = \mathrm{O}(g)$ if $\limsup_{n\to\infty} |f(n)/g(n)| < \infty$. $f = \Omega(g)$ if $g = O(f)$. If both $f = \mathrm{O}(g)$ and $f = \Omega(g)$, then $f = \Theta(g)$. Similarly, $f = \mathrm{o}(g)$ if $\limsup_{n\to\infty} |f(n)/g(n)| = 0$, and $f = \omega(g)$ if $g = \mathrm{o}(f)$. The term *w.h.p.* means with high probability as $n \to \infty$.

2. SETTING AND MODEL

In this section, we consider a stylized model for large scale distributed content systems that captures two emerging trends, namely, a large number of content types, and uneven and time-varying demands.

2.1 Server and Storage Model

The system consists of n front-end servers, each with constant storage and service capacity, and a back-end server that contains a catalog of m content-types (one copy of each content-type, e.g., a copy of each YouTube video). The contents can be copied from the back-end server and placed on the front-end servers. Each front-end server can store at most d content pieces (d is a constant) and serve at most d requests at each time, under the constraint that no two requests can read the same content piece simultaneously on a server[1]. The system is essentially equivalent to a system of nd servers, each with 1 content piece storage. Since we are interested in the scaling performance, as $n, m \to \infty$, for clarity we assume that there are n servers and each server can store 1 content and can serve 1 request at any time.

2.2 Service Model

When a request for a content arrives, it is routed to an idle (front-end) server which has the corresponding content-type stored on it, if possible. We assume that the service time of each request is exponentially distributed with mean 1. The requests have to be served in an online manner; further service is non-preemptive, i.e., once a request is assigned to

[1]Even without the constraint "no two requests can read the same content piece simultaneously on a server", the performance can be bounded from above by the performance of a system with dn servers with a storage of 1 each, and from below by that of another system with n servers with a storage of 1 each. Thus asymptotically in a scaling-sense, the system is still equivalent to a system of n servers where each server can store 1 content and can serve 1 content request at any time.

a server, its service cannot be interrupted and also cannot be re-routed to another server. Requests that cannot be served (no free server with requested content-type) incur a high cost (e.g., need to be served by the back-end server, or content needs to be fetched from the back-end server and loaded on to a new server). As discussed before, we refer to such requests as deferred requests. The goal is to design content placement policies such that the number of requests deferred is minimized.

2.3 Content Request Model

There are m content-types (e.g., m distinct YouTube videos). We consider the setting where the number of content-types m is very large and scales linearly with the system size n, i.e., $m = \alpha n$ for some constant $\alpha > 1$. We assume that requests for each content arrive according to a Poisson process and request rates (popularities) follow a Zipf distribution. Formally, we make the following assumptions on the arrival process.

Assumption 1. *(Arrival and Content Request Process)*

- *The arrival process for each content-type i is a Poisson process with rate λ_i.*
- *The load on the system at any time is $\bar{\lambda} < 1$, where $\bar{\lambda} = \frac{\sum_{i=1}^{m} \lambda_i}{n}$.*
- *Without loss of generality, content-types are indexed in the order of popularity. The request rate for content-type i is $\lambda_i = n\bar{\lambda}p_i$ where $p_i \propto i^{-\beta}$ for some $\beta > 0$. This is the Zipf distribution with parameter β.*

We have used the Zipf distribution to model the popularity distribution of various contents because empirical studies in many content delivery systems have shown that the distribution of popularities matches well with such distributions, see e.g., [5], [2], [24], [7], [17].

2.4 Time Scales of Change in Arrival Process

A key trend discussed earlier is the time-varying nature of popularities in content delivery systems [1, 5]. For example, the empirical study in [5] (based on 25 millions transactions on YouTube) shows that daily top 100 list of videos changes frequently. To understand the effect of this trend on the performance of content placement strategies, we consider the following two change models.

Block Change Model: In this model, we assume that the popularity of various content-types remains constant for some duration of time $T(n)$, and then changes to some other arbitrarily chosen distribution that satisfies Assumption 1. Thus $T(n)$ reflects the time-scale over which changes in popularities occur. Under this model, we characterize the performance of content placement strategies over such a time-scale $T(n)$.

Continuous Change Model: Under this model, we assume that each content-type has a Poisson clock at some constant rate $\nu > 0$. Whenever the clock of content-type i ticks, content-type i exchanges its popularity with some other content-type j, chosen uniformly at random. Note that if $T(n) = \theta(1)$, the average time over which the popularity distribution "completely" changes is comparable to that of the Block Change Model; however, here the change occurs incrementally and continuously. Note that this model ensures that the content-type popularity always has the Zipf distribution. Under this model, we characterize the performance of content placement strategies over constant intervals of time.

3. MAIN RESULTS AND DISCUSSION

In this section, we state and discuss our main results. The proofs are provided in Section 5.

3.1 Separating Learning from Content Placement

In this section, we analyze the performance of storage policies which separate the task of learning and that of content placement as follows. Consider time intervals of length $T(n)$. The operation of the policy in each time interval is divided into two phases:

Phase 1. Learning: Over this interval of time, use the demands from the arrivals (see Figure 1) to estimate the content-type popularity statistics.

Phase 2. Storage: Using the estimated popularity of various content-types, determine which content-types are to be replicated and stored on each server. The storage is fixed for the remaining time interval. The content-types not requested even once in the learning phase are treated equally in the storage phase. In other words, the popularity of all *unseen* content-types in the learning phase is assumed to be the same.

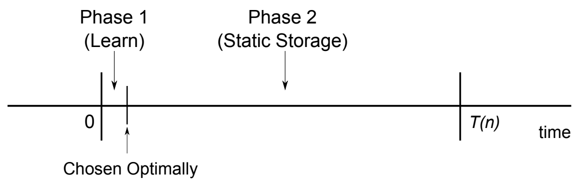

Figure 1: Learning-Based Static Storage Policies – *The interval $T(n)$ is split into the Learning and Storage phases. The length of time spent in the Learning phase can be chosen optimally using the knowledge of the value of $T(n)$ and the Zipf parameter β.*

Further, we allow the interval of time for the Learning phase potentially to be chosen optimally using knowledge of $T(n)$ (the interval over which statistics remain stationary) and β (the Zipf parameter for content-types popularity).

This is a natural class of policies to consider because it is obvious that popular content-types should be stored on more servers than the less popular content-types. Therefore, knowing the arrival rates can help in the design of better storage policies. Moreover, for the content-types which are not seen in the learning phase, the storage policy has no information about their relative popularity. It is therefore natural to treat them as if they are equally popular.

The replication and storage in Phase 2 (Storage) can be performed by *any* static policy that relies on the knowledge (estimate) of arrival rates, e.g., the proportional placement policy [10] where the number of copies of each content-type is proportional to its arrival rate, or the storage policy of [11] which was shown to be approximately optimal in the steady state.

We now analyze the performance of learning-based static storage policies under the Block Change Model defined in Section 2.4 where the statistics remain invariant over the time intervals of length $T(n)$. The performance metric of interest is the number of requests deferred by any policy belonging to class of learning-based static storage policies in the interval of interest. We assume that at the beginning of this interval, the storage policy has no information about the relative popularity of various content-types. Therefore, we start with an initial loading where each content-type is placed on exactly one server. This loading is not changed during Phase 1 (the learning phase) at the end of which, the content-type on idle servers is changed as per the new storage policy. As mentioned before, this storage is not changed for the remaining duration in the interval of interest.

The following theorem provides a lower bound on the number of requests deferred by any learning-based static storage policy.

Theorem 1. *Under Assumption 1 and the Block Change Model defined in Section 2.4, for $\beta > 2$, if $T(n) = \Omega(1)$, the expected number of requests deferred by any learning-based static storage policy is $\Omega(n^{0.5})$.*

We therefore conclude that even if the division of the interval of interest into Phase 1 (Learning) and Phase 2 (Storage) is done in the optimal manner, no learning-based static storage policy can defer fewer than $\Omega(n^{0.5})$ jobs in the interval of interest. Therefore, Theorem 1 provides a fundamental lower bound on the number of jobs deferred by any policy which separates learning and storage. It is worth pointing out that this result holds even when the time-scale of change in statistics is quite slow. Thus, even when $T(n)$, the time-scale over which statistics remains invariant, goes to infinity and the time duration of the two phases (Learning, Storage) is chosen optimally based on $\beta, T(n)$, $\Omega(n^{0.5})$ requests are still deferred.

Next, we explore *adaptive storage policies* which perform the task of learning and storage simultaneously.

3.2 Myopic Joint Learning and Placement

We next study a natural adaptive storage policy called MYOPIC. In an adaptive storage policy, depending on the requests that arrive and depart, the content-type stored on a server can be changed when the server is idle while other servers of the system might be busy serving requests. Therefore, adaptive policies perform the tasks of learning and placement jointly. Many variants of such adaptive policies have been studied for decades in the context of cache management (e.g. LRU, LRU-MIN [21]).

Let C_i refer to the i^{th} content-type, $1 \leq i \leq m$. The MYOPIC policy works as follows: When a request for content-type C_i arrives, it is assigned to a server if possible, or deferred otherwise. Recall that a deferred request is a request for which on arrival, no currently idle server can serve it and thus its service invokes a backup mechanism such as a back-end server which can serve it at a high cost. After the assigment/defer decision is made, if there are no currently idle servers with content-type C_i, MYOPIC replaces the content-type of one of the idle servers with C_i. This idle server is chosen as follows:

- If there is a content-type C_j stored on more than one currently idle servers, the content-type of one of those servers is replaced with C_i,
- Else, place C_i on that currently idle server whose content-type has been requested least recently among the content-types on the currently idle servers.

For a formal definition of MYOPIC, refer to Figure 2.

```
1: On arrival (request for $C_i$) **do**,
2: Allocate request to an idle server if possible.
3: **if** no other idle server has a copy of $C_i$, **then**
4:    **if** $\exists j$: $C_j$ stored on $> 1$ idle servers, **then**
5:       replace $C_j$ with $C_i$ on any one of them.
6:    **else**
7:       find $C_j$: least recently requested on idle servers,
         replace $C_j$ with $C_i$.
8:    **end if**
9: **end if**
```

Figure 2: MYOPIC – *An adaptive storage policy which changes the content stored on idle servers in a greedy manner to ensure that recently requested content pieces are available on idle servers.*

Remark 1. *Some key properties of MYOPIC are:*

1. *The content-types on servers can be potentially changed only when there is an arrival.*
2. *The content-type of at most one idle server is changed after each arrival. However, for many popular content-types, it is likely that there is already an idle server with the content-type, in which case there is no content-type change.*
3. *To implement MYOPIC, the system needs to maintain a list of content types ordered according to the time at which recent most request of each content-type was made.*

The following theorem provides an upper bound on the number of requests deferred by MYOPIC for the Block Change Model defined in Section 2.4.

Theorem 2. *Under Assumption 1 and the Block Change Model defined in Section 2.4, over any time interval $T(n)$ such that $T(n) = \mathrm{o}(n^{\beta-1})$, the number of requests deferred by MYOPIC is $\mathrm{O}((nT(n))^{1/\beta})$ w.h.p.*

We now compare this upper bound with the lower bound on the number of requests deferred by any learning-based static storage policy obtained in Theorem 1.

Corollary 1. *Under Assumption 1, the Block Change Model defined in Section 2.4, and for $\beta > 2$, over any time interval $T(n)$ such that $T(n) = \Omega(1)$ and $T(n) = \mathrm{o}(n^{\frac{\beta}{2}-1})$, the expected number of requests deferred by any learning-based static storage policy is $\Omega(n^{0.5})$ and the number of requests deferred by the MYOPIC policy is $\mathrm{o}(n^{0.5})$ w.h.p.*

From Corollary 1, we conclude that MYOPIC outperforms all learning-based static storage policies. Note that:

i. Corollary 1 holds even when the interval of interest $T(n)$ grows to infinity (scaling polynomially in n), or correspondingly, even when the content-type popularity changes very slowly with time.

ii. Even if the partitioning of the $(T(n))$ into a Learning phase and a Static Storage phase is done in an optimal manner with the help of some side information $(\beta, T(n))$, the MYOPIC algorithm outperforms any learning-based static storage policy.

iii. Since we consider the high-dimensional setting, the learning problem at hand is a large-alphabet learning problem. It is well known that standard estimation techniques like using the empirical values as estimates of the true statistics is suboptimal in this setting. Many learning algorithm like the classical Good-Turing estimator [13] and other linear estimators [16] have been proposed, and shown to have good performance for the problem of large-alphabet learning. From Corollary 1, we conclude that, even if the learning-based storage policy uses the best possible large-alphabet estimator, it cannot match the performance of the MYOPIC policy.

Therefore, in the high-dimensional setting we consider, separating the task of estimation of the demand statistics, and the subsequent use of the same to design a static storage policy, is strictly suboptimal. This is the key message of this paper.

Theorem 2 characterizes the performance of MYOPIC under the Block Change Model, where the statistics of the arrival process do not change in interval of interest. To gain further insight into robustness of MYOPIC against changes in the arrival process, we now analyze the performance of MYOPIC when the arrival process can change in the interval of interest according to the Continuous Change Model defined in Section 2.4.

Recall that under the Continuous Change Model, on average, we expect $\Theta(n)$ shuffles in the popularity of various content-types in an interval of constant duration. For the Block Change Model, if $T(n) = \Theta(1)$, the entire popularity distribution can change at the end of the block, which is equivalent to n shuffles. Therefore, for both the change models, the expected number of changes to the popularity distribution in an interval of constant duration is of the same order. However, these changes occur constantly but slowly in the Continuous Change Model as opposed to a one-shot change in the Block Change Model.

Theorem 3. *Under Assumption 1, and the Continuous Change Model defined in Section 2.4, the number of requests deferred by the MYOPIC storage policy in any interval of constant duration is* $\mathrm{O}(n^{1/\beta})$ *w.h.p.*

In view of Theorem 2, if the arrival rates do not vary in an interval of constant duration, under the MYOPIC storage policy, the number of requests deferred in that interval is $\mathrm{O}(n^{1/\beta})$ w.h.p. Theorem 3 implies that the number of requests deferred in a constant duration interval is of the same order even if the arrival rates change according to the Continuous Change Model. This shows that the performance of the MYOPIC policy is robust to changes in the popularity statistics.

3.3 Genie-Aided Optimal Storage Policy

In this section, our objective is to study the setting where the content-type statistics is available "for free". We consider the setting where the popularity statistics are known, and show that a simple adaptive policy is optimal in the class of all policies which know popularity statistics of various content-types. We denote the class of such policies as $\mathbb{A}$ and refer to the optimal policy as the GENIE policy.
Let the content-types be indexed from $i = 1$ to m and let C_i be the i^{th} content-type. Without loss of generality, we assume that the content-types are indexed in the order of popularity, i.e, $\lambda_i \geq \lambda_{i+1}$ for all $i \geq 1$. Let $k(t)$ denote the number of idle servers at time t.

The key idea of the GENIE storage policy is to ensure that at any time t, if the number of idle servers is $k(t)$, the $k(t)$ most popular content-types are stored on exactly one idle server each. The GENIE storage policy can be implemented as follows. Recall C_i is the i^{th} most popular content-type. At time t,

- If there is a request for content-type C_i with $i < k(t^-)$, then allocate the request to the corresponding idle server. Further, replace the content-type on server storing $C_{k(t^-)}$ with content-type C_i.
- If there is a request for content-type C_i with $i > k(t^-)$, defer this request. There is no storage update.
- If there is a request for content-type C_i with $i = k(t^-)$, then allocate the request to the corresponding idle server. There is no storage update.
- If a server becomes idle (due to a departure), replace its content-type with $C_{k(t^-)+1}$.

For a formal definition, please refer to Figure 3.

```
1:  Initialize: Number of idle-servers := $k = n$.
2:  **while** true **do**
3:      **if** new request (for $C_i$) routed to a server, **then**
4:          **if** $i \neq k$, **then**
5:              replace content-type of idle server storing $C_k$ with $C_i$
6:          **end if**
7:          $k \leftarrow k - 1$
8:      **end if**
9:      **if** departure, **then**
10:         replace content-type of new idle server with $C_{k+1}$
11:         $k \leftarrow k + 1$
12:     **end if**
13: **end while**
```

Figure 3: GENIE – *An adaptive storage policy which has content popularity statistics available for "free". At time t, if the number of idle servers is $k(t)$, the $k(t)$ most popular content-types are stored on exactly one idle server each.*

Remark 2. *Some key properties of GENIE are:*

1. *The implementation of GENIE requires replacing the content-type of at most one server on each arrival and departure.*

2. *The GENIE storage policy only requires the knowledge of the relative popularity of various content types.*

To characterize the performance of GENIE, we assume that the system starts from the empty state (all servers are idle) at time $t = 0$. The performance metric for any policy $\mathcal{A}$ is $D^{(\mathcal{A})}(t)$, defined as the number of requests deferred by time t under the adaptive storage policy $\mathcal{A}$. We say that an adaptive storage policy $\mathcal{O}$ is optimal if

$$D^{(\mathcal{O})}(t) \leq_{st} D^{(\mathcal{A})}(t), \tag{1}$$

for any storage policy $\mathcal{A} \in \mathbb{A}$ and any time $t \geq 0$. Where Equation 1 implies that,

$$\mathbb{P}(D^{(\mathcal{O})}(t) > x) \leq \mathbb{P}(D^{(\mathcal{A})}(t) > x),$$

for all $x \geq 0$ and $t \geq 0$.

Theorem 4. *If the arrival process to the content-type delivery system is Poisson and the service times are exponential random variables with mean 1, for the Block Change Model defined in Section 2.4, let $D^{(\mathcal{A})}(t)$ be the number of requests deferred by time t under the adaptive storage policy $\mathcal{A} \in \mathbb{A}$. Then, we have that,*

$$D^{(GENIE)}(t) \leq_{st} D^{(\mathcal{A})}(t),$$

for any storage policy $\mathcal{A} \in \mathbb{A}$ and any time $t \geq 0$.

Note that this theorem holds even if the λ_is are not distributed according to the Zipf distribution. We thus conclude that GENIE is the optimal storage policy in the class of all storage policies which at time t, have no additional knowledge of the future arrivals except the values of λ_i for all content-types and the arrivals and departures in $[0, t)$. Next, we compute a lower bound on the performance of GENIE.

Theorem 5. *Under Assumption 1, for $\beta > 2$, the Block Change Model defined in Section 2.4 and if the interval of interest is of constant length, the expected number of requests deferred by GENIE is $\Omega(n^{2-\beta})$.*

From Theorems 2 and 5 we see that there is a gap in the performance of the MYOPIC policy and the GENIE policy (which has additional knowldge of the content-type popularity statistics). Since for the GENIE policy, learning the statistics of the arrival process comes for "free", this gap provides an upper bound on the cost of serving content-type with *unknown* demands. We compare the performance of the all the policies considered so far in the next section via simulations.

4. SIMULATION RESULTS

We compare the performance of the MYOPIC policy with the performance of the GENIE policy and the following two learning-based static storage policies:

- The *"Empirical + Static Storage"* policy uses the empirical popularity statistics of content types in the learning phase as estimates of the the true popularity statistics. At the end of the learning phase, the number of servers on which a content is stored is proportional to its estimated popularity.
- The *"Good Turing + Static Storage"* policy uses the Good-Turing estimator [13] to compute an estimate of the missing mass at the end of the learning phase. The missing mass is defined as total probability mass of the content types that were not requested in the learning phase. Recall that we assume that learning-based static storage policies treat all the missing content-types equally, i.e., all missing content-types are estimated to be equally popular.

  Let M_0 be the total probability mass of the content types that were not requested in the learning phase and S_1 be the set of content types which were requested exactly once in the learning phase. The Good-Turing estimator of the missing mass ($\widehat{M}_0$) is given by

  $$\widehat{M}_0 = \frac{|S_1|}{\text{number of samples}}.$$

  See [13] for details.

  Let N_i be the number of times content i was requested in the learning phase and $\mathcal{C}_{\text{missing}}$ be the set of content-types not requested in the learning phase. The "Good Turing + Static Storage" policy computes an estimate of the content-popularity as follows:

  i: If $N_i = 0$, $p_i = \dfrac{\widehat{M}_0}{|\mathcal{C}_{\text{missing}}|}$.

  ii: If $N_i > 0$, $p_i = (1 - \widehat{M}_0)\dfrac{N_i}{\text{number of samples}}$.

  At the end of the learning phase, the number of servers on which a content is stored is proportional to its estimated popularity.

We simulate the content distribution system for arrival and service process which satisfy Assumption 1 to compare the performance of the four policies mentioned above and also understand how their performance depends on various parameters like system size (n), load ($\bar{\lambda}$) and Zipf parameter (β). In Tables 1, 2 and 3, we report the mean and variance of the fraction of jobs served by the policies over a duration of 5 s ($T(n) = 5$) for $\alpha = 1$.

For each set of system parameters, we repeat the simulations between 1000 to 10000 times for each policy in order to ensure that the standard deviation of the quantity of interest (fraction of jobs served) is small and comparable. For the two adaptive policies (GENIE and MYOPIC), the results are averaged over 1000 iterations and for the learning-based policies ("Empirical + Static Storage" and "Good-Turing + Static Storage"), the results are averaged over 10000 iterations. In addition, the results for the learning-based policies are reported for empirically optimized values for the fraction of time spent by the policy in learning the distribution.

In Table 1, we compare the performance of the policies for different values of system size (n). For the results reported in Table 1, the "Empirical + Static Storage" policy learns for 0.1 s and the "Good Turing + Static Storage" policy learns for 0.7 s. The performance of all four policies improves as the system size increases and the adaptive policies significantly outperform the two learning-based static storage policies. Figure 4 is a plot of the mean values reported in Table 1.

In Table 2, we compare the performance of the policies for different values of Zipf parameter β. For the results reported in Table 2, the duration of the learning phase for

Policy	n	Mean	σ
GENIE	200	0.9577	0.0081
	400	0.9698	0.0045
	600	0.9752	0.0034
	800	0.9788	0.0030
	1000	0.9814	0.0025
MYOPIC	200	0.8995	0.0258
	400	0.9260	0.0167
	600	0.9380	0.0132
	600	0.9481	0.0101
	1000	0.9532	0.0080
Empirical + Static Storage	200	0.6292	0.0662
	400	0.6918	0.0443
	600	0.7246	0.0353
	800	0.7464	0.0304
	1000	0.7622	0.0268
Good Turing + Static Storage	200	0.6875	0.0274
	400	0.7249	0.0180
	600	0.7443	0.0140
	800	0.7566	0.0118
	1000	0.7651	0.0104

Table 1: *The performance of the four policies as a function of the system size (n) for fixed values of load $\bar{\lambda} = 0.8$ and $\beta = 1.5$. The values reported are the mean and standard deviation (σ) of the fraction of jobs served. Both adaptive policies (GENIE and MYOPIC) significantly outperform the two learning-based static storage policies.*

both learning based policies is fixed such that the expected number of arrivals in that duration is 100. The performance of all four policies improves as the value of the Zipf parameter β increases, however, the MYOPIC policy outperforms both learning-based static storage policies for all values of β considered.

In Table 3, we compare the performance of the policies for different values of load $\bar{\lambda}$. For the results reported in Table 3, the duration of the learning phase for both learning based policies is fixed such that the expected number of arrivals in that duration is 100. The performance of all four policies deteriorates as the load increases, however, for all loads considered, the MYOPIC policies outperforms the two learning-based static storage policies.

5. PROOFS OF MAIN RESULTS

In this section, we provide the proofs of our main results.

5.1 Proof of Theorem 1

We first present an outline of the proof of Theorem 1. We consider two cases. We first focus on the case when the learning-based storage policies use fewer than n arrivals to learn the distribution.

1. If the learning phase lasts for the first n^γ arrivals for some $0 < \gamma \leq 1$, we show that under Assumption 1, w.h.p., in the learning phase, there are no arrivals for content-types $k, k+1, ..m = \alpha n$ for $k = (n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$ (Lemma 1).

2. Next, we show that w.h.p., among the first n^γ arrivals, i.e., during the learning phase, $\Omega(n^\gamma)$ requests are deferred (Lemma 3).

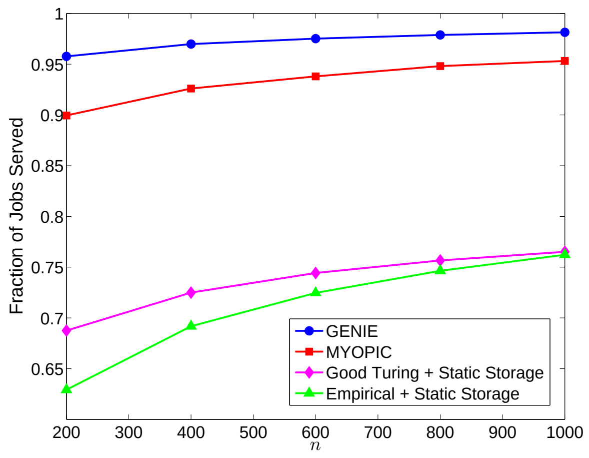

Figure 4: *Plot of the mean values reported in Table 1 – performance of the storage policies as a function of system size (n) for $\bar{\lambda} = 0.8$ and $\beta = 1.5$.*

3. Using Lemma 1, we compute a lower bound on the number of requests deferred in Phase 2 (after the learning phase) by any learning-based static storage policy (Lemma 4).

4. Using Steps 2 and 3, we lower bound the number of requests deferred in the interval of interest.

In the case when the learning phase lasts for more than n arrivals, we show that the number of requests deferred in the learning phase alone is $\Omega(n)$, thus proving the theorem for this case.

LEMMA 1. *Let E_1 be the event that in the first n^γ arrivals for $0 < \gamma \leq 1$, there are no arrivals of types $k, k+1, ...m = \alpha n$ where $k = (n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$. Then,*

$$\mathbb{P}(E_1) \geq \exp\left(-\frac{1}{2\log n}\right), \tag{2}$$

for n large enough.

PROOF. Recall $\lambda_i = \bar{\lambda} n p_i$ where $p_i = \frac{i^{-\beta}}{Z(\beta)}$ for $Z(\beta) = \sum_{i=1}^{m} i^{-\beta}$.

$$Z(\beta) = \sum_{i=1}^{\alpha n} i^{-\beta} \geq \int_1^{\alpha n+1} i^{-\beta} di \geq \frac{0.9}{\beta - 1}$$

for n large enough. Therefore, for all i,

$$p_i \leq \frac{\beta-1}{0.9} i^{-\beta}.$$

The total mass of all content types $i = k, ..m = \alpha n$ is

$$\sum_{i=k}^{\alpha n} p_i \leq \sum_{i=k}^{\alpha n} \frac{\beta-1}{0.9} i^{-\beta} \leq \int_{k-1}^{\alpha n} \frac{\beta-1}{0.9} i^{-\beta} di \leq \frac{1}{0.9} \frac{1}{(k-1)^{\beta-1}}.$$

For $k = (n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$, we have that,

$$\mathbb{P}(E_1) \geq \left(1 - \frac{1}{0.9} \frac{1}{(k-1)^{\beta-1}}\right)^{n^\gamma} \geq \exp\left(-\frac{1}{2\log n}\right),$$

for n large enough. $\square$

Policy	β	Mean	σ
GENIE	2.0	0.9940	0.0025
	3.5	0.9998	0.0013
	5.0	0.9997	0.0017
MYOPIC	2.0	0.9774	0.0063
	3.5	0.9981	0.0024
	5.0	0.9990	0.0014
Empirical + Static Storage	2.0	0.8592	0.0206
	3.5	0.9328	0.0138
	5.0	0.9458	0.0092
Good Turing + Static Storage	2.0	0.8448	0.0231
	3.5	0.9314	0.0139
	5.0	0.9457	0.0091

Table 2: *The performance of the four policies as a function of the Zipf parameter (β) for fixed values of system size $n = 500$ and load $\bar{\lambda} = 0.9$. The values reported are the mean and standard deviation (σ) of the fraction of jobs served. The MYOPIC policy outperforms the two learning-based static storage policies for all values of β considered.*

Policy	λ	Mean	σ
GENIE	0.500	0.9892	0.0025
	0.725	0.9788	0.0013
	0.950	0.9531	0.0017
MYOPIC	0.500	0.9605	0.0113
	0.725	0.9484	0.0105
	0.950	0.8973	0.0221
Empirical + Static Storage	0.500	0.7756	0.0222
	0.725	0.7705	0.0238
	0.950	0.7352	0.0235
Good Turing + Static Storage	0.500	0.7849	0.0230
	0.725	0.7589	0.0249
	0.950	0.6869	0.0348

Table 3: *The performance of the four policies as a function of the load ($\bar{\lambda}$) for fixed values of system size $n = 500$ and $\beta = 1.2$. The values reported are the mean and standard deviation (σ) of the fraction of jobs served. The MYOPIC policy significantly outperforms the two learning-based static storage policies for all loads considered.*

We use the following concentration result for Exponential random variables.

Lemma 2. *Let X_k for $0 \le k \le v$, be i.i.d. exponential random variables with mean 1, then,*

$$\mathbb{P}\left(\sum_{k=1}^{v} X_i \le a\right) \le \exp(v-a)\left(\frac{a}{v}\right)^{v}. \qquad (3)$$

The proof of Lemma 2 follows from elementary calculations.

Lemma 3. *Suppose the system starts with each content piece stored on exactly one server. Let E_2 be the event that in the first n^γ arrivals for γ such that $0 < \gamma \le 1$, at most $((n^\gamma \log n)^{\frac{1}{\beta-1}} + 1)(\log n + 1)$ are served (not deferred). Then, for $\beta > 2$,*

$$\mathbb{P}(E_2) \ge 1 - \frac{1}{2\log n}. \qquad (4)$$

Proof. This proof is conditioned on the event E_1 defined in Lemma 1. Conditioned on E_1, in the first n^γ arrivals, at most $(n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$ different content types are requested. Therefore, at most $(n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$ servers can serve requests during the first n^γ arrivals.

Let E_3 be the event that the time taken for the first n^γ arrivals is less than $\frac{2n^\gamma}{\bar{\lambda}n}$. Since the expected time for the first n^γ arrivals is $\frac{n^\gamma}{\bar{\lambda}n}$, by the Chernoff bound, $\mathbb{P}(E_3) \ge 1 - o(1/n)$. The rest of this proof is conditioned on the event E_3.

If the system serves (does not defer) more than

$$((n^\gamma \log n)^{\frac{1}{\beta-1}} + 1)(\log n + 1)$$

requests in this interval, at least one server needs to serve more than $\log n$ requests. By substituting $a = cn^{-1+\gamma}$ and $v = \log n$ in Lemma 2, we have that,

$$\begin{aligned}\mathbb{P}\left(\sum_{k=1}^{\log n} X_k \le cn^{-1+\gamma}\right) &\le \exp(\log n - cn^{-1+\gamma}) \\ &\qquad \times\left(\frac{cn^{-1+\gamma}}{\log n}\right)^{\log n} = o\left(\frac{1}{n}\right).\end{aligned}$$

Therefore, the probability that a server serves more than $\log n$ requests in an interval of $\frac{2n^\gamma}{\bar{\lambda}n}$ time is $o\left(\frac{1}{n}\right)$. Therefore, using the union bound, the probability that none of these $(n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$ servers serve more than $\log n$ requests each in $\frac{2n^\gamma}{\bar{\lambda}n}$ time is greater than $1 - ((n^\gamma \log n)^{\frac{1}{\beta-1}} + 1)o(\frac{1}{n})$. Therefore, we have that,

$$\begin{aligned}\mathbb{P}(E_2^c) &\le ((n^\gamma \log n)^{\frac{1}{\beta-1}} + 1)o\left(\frac{1}{n}\right) + P(E_1^c) + P(E_3^c) \\ &\le \frac{1}{2\log n}\end{aligned}$$

for n large enough. □

Lemma 4. *Let the interval of interest be $T(n)$ such that $T(n) = \Omega(1)$. If the learning phase of the storage policy lasts for the first n^γ arrivals, $0 < \gamma \le 1$, the expected number of requests deferred in Phase 2 is $\Omega\left(\frac{T(n)n^{1-\gamma}}{\log n}\right)$.*

Proof. Let N_2 be the number of arrivals in Phase 2, then we have that, $E[N_2] = T(n)\bar{\lambda}n - n^\gamma$.

Let E_4 be the event that $N_2 > E[N_2]/2$. Using the Chernoff bound, it can be shown that $P(E_4^c) = o(1/n)$.

The rest of this proof is conditioned on E_1 defined in Lemma 1 and E_4 defined above. We consider the following two cases depending on the number of servers allocated to content types not seen in Phase 1.

Case I: The number of servers allocated to content types not seen in Phase 1 is less than ϵn for some $\epsilon \le 1 - \frac{\bar{\lambda}}{1000}$. For $\beta > 2$,

$$Z(\beta) = \sum_{i=1}^{\alpha n} i^{-\beta} \le \sum_{i=1}^{\alpha n} i^{-2} \le \sum_{i=1}^{\infty} i^{-2} = \frac{\pi^2}{6}.$$

Therefore, for all i, $p_i \ge \frac{6}{\pi^2} i^{-\beta}$. The total mass of all content types $k, k+1, ..\alpha n$ is

$$\sum_{i=k}^{\alpha n} p_i \;\ge\; \sum_{i=k}^{\alpha n} \frac{6}{\pi^2} i^{-\beta} \ge \int_{k}^{\alpha n+1} \frac{6}{\pi^2} i^{-\beta} di = \frac{0.54}{\pi^2(\beta-1)} \frac{1}{k^{\beta-1}},$$

for n large enough. Therefore, the expected number of arrivals of types $k, k+1, ..m = \alpha n$, where $k = (n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$ in Phase 2 is at least $(\frac{T(n)\bar{\lambda}n - n^\gamma}{2})\frac{0.54}{\pi^2(\beta-1)}\frac{1}{n^\gamma \log n}$.

Let E_5 be the event that in Phase 2, there are at least $(\frac{T(n)\bar{\lambda}n-n^\gamma}{2})\frac{0.54}{2\pi^2(\beta-1)}\frac{1}{n^\gamma \log n}$ arrivals of type $k, k+1, ..m = \alpha n$, where $k = (n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$. Using the Chernoff bound, $\mathbb{P}(E_5^c) = \mathrm{o}(1/n)$.

Conditioned on E_1, all content types $k, k+1, ..m = \alpha n$, where $k = (n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$, are not requested in Phase 1. Recall that all learning-based policies treat all these content types equally and that the total number of servers allocated to store the content types not seen in Phase 1 is less than ϵn. Let η be the probability that C_k for $k \geq (n^\gamma \log n)^{\frac{1}{\beta-1}} + 1$ is not stored by the storage policy under consideration. Then,

$$\eta \geq 1 - \frac{\epsilon n}{n - (n^\gamma \log n)^{\frac{1}{\beta-1}} - 1} \geq 1 - \frac{\epsilon}{2},$$

for n large enough.

Let $E_6 = E_1 \cap E_3 \cap E_4 \cap E_5$ and D_2 be the number of requests deferred in Phase 2.

$$\begin{aligned} E[D_2|E_6] &\geq \eta\bigg(\bigg(\frac{T(n)\bar{\lambda}n - n^\gamma}{2}\bigg)\frac{0.54}{2\pi^2(\beta-1)}\frac{1}{n^\gamma \log n}\bigg) \\ &\geq \bigg(1 - \frac{\epsilon}{2}\bigg)\bigg(\frac{T(n)\bar{\lambda}n - n^\gamma}{2}\bigg)\frac{0.54}{2\pi^2(\beta-1)}\frac{1}{n^\gamma \log n} \\ &= \Omega\bigg(\frac{T(n)n^{1-\gamma}}{\log n}\bigg). \end{aligned}$$

Therefore,

$$\begin{aligned} E[D_2] &\geq E[D_2|E_6]\mathbb{P}(E_6) \\ &\geq E[D_2|E_6]\bigg(1 - \frac{1}{\log n} - \frac{3}{n}\bigg) = \Omega\bigg(\frac{T(n)n^{1-\gamma}}{\log n}\bigg). \end{aligned}$$

Case II: The number of servers allocated to content types not seen in Phase 1 is more than ϵn for some $\epsilon > 1 - \frac{\bar{\lambda}}{1000}$.

Let $f(n)$ be the number of servers allocated to store all content types that are requested in Phase 1. By our assumption, $f(n) \leq \frac{\bar{\lambda}}{1000}n$.

Let $\mathbf{C}_1$ be the set of content types requested in Phase 1. Let $p = \sum_{c\in\mathbf{C}_1} p_c$ be the total mass of all content types $c \in \mathbf{C}_1$. Let $\hat{p}_c$ be the fraction of requests for content-type c in Phase 1. By the definition of $\mathbf{C}_1$, the total empirical mass of all content types $c \in \mathbf{C}_1$ is obviously $\hat{p} = \sum_{c\in\mathbf{C}_1} \hat{p}_c = 1$.

Recall that there are n^γ arrivals in Phase 1. Let $r = n^\gamma$. We now use the Chernoff bound to compute a lower bound on the true mass p, using a technique similar to that used in [13] (Lemma 4). By the Chernoff bound, we know that,

$$\mathbb{P}(\hat{p} > (1+\kappa)p) \leq \exp\bigg(-\frac{pr\kappa^2}{3}\bigg).$$

Let $\delta = \exp\bigg(-\frac{pr\kappa^2}{3}\bigg)$, then, we have that, with probability greater than $1 - \delta$,

$$\hat{p} - p > \sqrt{\frac{-3p\log\delta}{r}}.$$

Solving for p, we get that, with probability greater than $1 - \delta$, $p > 1 - \dfrac{3\log(1/\delta)}{2r}$, for n large enough. Let $\delta = 1/n$, then we have that, with probability greater than $1 - 1/n$, $p > 1 - \dfrac{3\log n}{2n^\gamma}$. Conditioned on the event E_4, there are at least $\frac{T(n)\bar{\lambda}n - n^\gamma}{2}$ arrivals in Phase 2. The remainder of this proof is conditioned on E_4. Let A_2 be the number of arrivals of types $c \in \mathbf{C}_1$ in phase 2. Let E_7 be the event that

$$A_2 > \frac{T(n)\bar{\lambda}n - n^\gamma}{2}\bigg(1 - \frac{3\log n}{2n^\gamma}\bigg).$$

Since the expected number of arrivals of content types $c \in \mathbf{C}_1$ in Phase 2 is at least

$$(T(n)\bar{\lambda}n - n^\gamma)\bigg(1 - \frac{3\log n}{2n^\gamma}\bigg),$$

using the Chernoff bound, we can show that $\mathbb{P}(E_7^c) = \mathrm{o}(1/n)$. The rest of this proof is conditioned on E_7. By our assumption, the number of servers which can serve arrivals of types $c \in \mathbf{C}_1$ in Phase 2 is $f(n)$. Therefore, if at least $A_2/2$ requests are to be served in Phase 2, the sum of the service times of these $A_2/2$ requests should be less than $T(n)f(n)$ (since the number of servers which can serve these requests is $f(n)$). Let E_8 be the event that the sum of $A_2/2$ independent Exponential random variables with mean 1 is less than $T(n)f(n)$. By substituting $v = A_2/2$ and $a = T(n)f(n)$ in Lemma 2, we have that,

$$\begin{aligned} \mathbb{P}(E_8) &\leq \exp\bigg(\frac{A_2}{2} - T(n)\bigg)\bigg(\frac{2T(n)f(n)}{A_2}\bigg)^{\frac{A_2}{2}} \\ &\leq \exp\bigg(\frac{A_2}{2}\bigg)\bigg(\frac{2T(n)f(n)}{A_2}\bigg)^{\frac{A_2}{2}} = \mathrm{o}\bigg(\frac{1}{n}\bigg) \end{aligned}$$

for n large enough. Hence,

$$\begin{aligned} \mathbb{P}\bigg(D_2 \geq \frac{A_2}{2}\bigg) &\geq 1 - \mathbb{P}(E_1^c) - o\bigg(\frac{1}{n}\bigg) \\ \Rightarrow E[D_2] &= \Omega\bigg(\frac{T(n)n^{1-\gamma}}{\log n}\bigg). \end{aligned}$$

□

PROOF. (of Theorem 1)
We consider two cases:
Case I: The learning phase lasts for the first n^γ arrivals where $0 \leq \gamma \leq 1$.
Let D_1 be the number of requests deferred in Phase 1 and D be total number of requests deferred in the interval of interest. Then, we have that,

$$E[D] = E[D_1] + E[D_2].$$

By Lemmas 3 and 4 and since $T(n) = \Omega(1)$, we have that,

$$\begin{aligned} E[D] &\geq n^\gamma - (n^\gamma \log n)^{\frac{1}{\beta-1}}\log n + E[D_2] \\ &= \Omega(n^{0.5}). \end{aligned}$$

Case II: The learning phase lasts for longer than the time taken for the first n arrivals.
By Lemma 3, the number of requests deferred in the first n arrivals is at least $n - (n\log n)^{\frac{1}{\beta-1}}\log n$ with probability greater than $1 - 1/\log n$. Therefore, we have that,

$$E[D] \geq \bigg(n - (n\log n)^{\frac{1}{\beta-1}}\log n\bigg)\bigg(1 - \frac{1}{\log n}\bigg) = \Omega(n^{0.5}).$$

□

5.2 Proof of Theorem 2

We first present an outline the proof of Theorem 2.

1. We first show that under Assumption 1, on every arrival in the interval of interest $(T(n))$, there are $\Theta(n)$ idle servers w.h.p. (Lemma 6).
2. Next, we show that w.h.p., in the interval of interest of length $T(n)$, only $\mathrm{O}\big((nT(n))^{\frac{1}{\beta}}\big)$ unique content types are requested (Lemma 7).
3. Conditioned on Steps 1 and 2, we show that, the MYOPIC policy ensures that in the interval of interest, once a content type is requested for the first time, there is always at least one idle server which can serve an incoming request for that content.
4. Using Step 3, we conclude that, in the interval of interest, only the first request for a particular content type will be deferred. The proof of Theorem 2 then follows from Step 2.

Lemma 5. *Let the cumulative arrival process to the content delivery system be a Poisson process with rate $\bar{\lambda}n$. At time t, let $\chi(t)$ be the number of occupied servers under the MYOPIC storage policy. Then, we have that, $\chi(t) \leq_{st} S(t)$, where $S(t)$ is a poisson random variable with rate $\bar{\lambda}n(1-e^{-t})$.*

Proof. Consider an $M/M/\infty$ queue where the arrival process is Poisson($\bar{\lambda}n$). Let $S(t)$ be the number of occupied servers at time t in this system. It is well known that $S(t)$ is a Poisson random variable with rate $\bar{\lambda}n(1-e^{-t})$.

Here we provide a proof of this result for completeness. Consider a request r^* which arrived into the system at time $t_0 < t$. If the request is still being served by a server, we have that, $t_0 + \mu(r^*) > t$, where $\mu(r^*)$ is the service time of request r^*. Since $\mu(r^*) \sim \mathrm{Exp}(1)$, we have that, $\mathbb{P}(\mu(r^*) > t - t_0 | t_0) = e^{-(t-t_0)}$. Therefore, $\mathbb{P}(r^*$ in the system at time $t) \leq \int_0^t \frac{1}{t} e^{-(t-t_0)} dt_0$.

To show $\chi(t) \leq_{st} S(t)$, we use a coupled construction similar to Figure 5. The intuition behind the proof is the following: the rate of arrivals to the content delivery system and the $M/M/\infty$ system (where each server can serve all types of requests) is the same. The content delivery system serves fewer requests than the $M/M/\infty$ system because some requests are deferred even when the servers are idle. Hence, the number of busy servers is the content delivery system is stochastically dominated by the number of busy servers in the $M/M/\infty$ queueing system. □

Lemma 6. *Let the interval of interest be $[t_0, t_0 + T(n)]$ where $T(n) = \mathrm{o}(n^{\beta-1})$ and $\varepsilon \leq \frac{1-\bar{\lambda}}{2}$. Let F_1 be the event that at the instant of each arrival in the interval of interest, the number of idle servers in the system is at least $\big(1 - \bar{\lambda} - \varepsilon\big)n$. Then, $\mathbb{P}(F_1^c) = \mathrm{o}\big(\frac{1}{n}\big)$.*

Proof. Let F_2 be the event that the number of arrivals in $[t_0, t_0 + T(n)] \leq nT(n)(\bar{\lambda} + \varepsilon)$. Using the Chernoff bound for the Poisson process, we have that,

$$\mathbb{P}(F_2^c) = o\bigg(\frac{1}{n}\bigg).$$

Consider any $t \in [t_0, t_0 + T(n)]$. By Lemma 5, $\chi(t) \leq_{st} S(t)$, where $S(t) \sim \mathrm{Poisson}(\bar{\lambda}n(1-e^{-t}))$. Therefore,

$$\mathbb{P}(\chi(t) > (\bar{\lambda}+\varepsilon)n) \leq \mathbb{P}(S(t) > (\bar{\lambda}+\varepsilon)n).$$

Moreover, $S(t) \leq_{st} W(t)$ where $W(t) = \mathrm{Poisson}(\bar{\lambda}n)$. Therefore, using the Chernoff bound for $W(t)$, we have that,

$$\mathbb{P}(S(t) > (\bar{\lambda}+\varepsilon)n) \leq \mathbb{P}(W(t) > (\bar{\lambda}+\varepsilon)n) = e^{-c_1 n},$$

for some constant $c_1 > 0$. Therefore,

$$\begin{aligned}\mathbb{P}(F_1^c) &\leq \mathbb{P}(F_2^c) + (\bar{\lambda}+\varepsilon)nT(n)\mathbb{P}(\chi(t) > (\bar{\lambda}+\varepsilon)n) \\ &= o\bigg(\frac{1}{n}\bigg).\end{aligned}$$

□

Lemma 7. *Let F_3 be the event that in the interval of interest of duration $T(n)$ such that $T(n) = \mathrm{o}(n^{\beta-1})$, no more than $\mathrm{O}((nT(n))^{1/\beta})$ different types of contents are requested. Then, $\mathbb{P}(F_3^c) = \mathrm{o}\big(\frac{1}{n}\big)$.*

Proof. Recall from the proof of Lemma 1 that the total mass of all content types $k, ..m = \alpha n$ is

$$\sum_{i=k}^{\alpha n} p_i \quad \leq \quad \frac{1}{0.9}\frac{1}{(k-1)^{\beta-1}}.$$

Now, for $k = (nT(n))^{1/\beta} + 1$, we have that,

$$\sum_{i=k}^{\alpha n} p_i \leq \frac{1}{0.9}(nT(n))^{-\frac{\beta-1}{\beta}}.$$

Conditioned on the event F_2 defined in Lemma 6, the expected number of requests for content types $k, k+1, ..\alpha n$ is less than $\frac{1}{0.9}(\bar{\lambda}+\varepsilon)(nT(n))^{1/\beta}$. Using the Chernoff bound, the probability that there are more than $\frac{2}{0.9}(\bar{\lambda}+\varepsilon)(nT(n))^{1/\beta}$ requests for content types $k, k+1, ..\alpha n$ in the interval of interest is less than $\frac{1}{n^2}$ for n large enough.

Therefore, with probability greater than $1 - 1/n^2 - \mathbb{P}(F_2^c)$, the number different types of contents requests for in the interval of interest is less than $(nT(n))^{1/\beta} + \frac{2}{0.9}(\bar{\lambda}+\varepsilon)(nT(n))^{1/\beta}$. Hence the result follows. □

Proof. (of Theorem 2)
Let F_4 be the event that, in the interval of interest, every request for a particular content type except the first request is not deferred. The rest of this proof is conditioned on F_1 and F_3. Let $U(t)$ be the number of unique contents which have been requested in the interval of interest before time t for $t \in [t_0, t_0 + T(n)]$. Conditioned on F_3, as defined in Lemma 7, $U(t) \leq k_1(nT(n))^{1/\beta}$ for some constant $k_1 > 0$ and n large enough. Conditioned on F_1, there are always $(1 - \bar{\lambda} - \varepsilon)n$ idle servers in the interval of interest.

CLAIM: For every i and n large enough, once a content C_i is requested for the first time in the interval of interest, the MYOPIC policy ensures that there is always at least 1 idle server which can serve a request for C_i.

Note that since $T(n) = \mathrm{o}(n^{\beta-1})$, $(nT(n))^{1/\beta} = \mathrm{o}(n)$. Let n be large enough such that $k_1(nT(n))^{1/\beta} < (1 - \bar{\lambda} - \varepsilon)n$, i.e., at any time $t \in [t_0, t_0 + T(n)]$, the number of idle servers is greater than $U(t)$. We prove the claim by induction. Let the claim hold for time t^- and let there be a request at time t for content C_i. If this is not the first request for C_i in $[t_0, t_0 + T(n)]$, by the claim, at $t = t^-$, there is at least one idle server which can serve this request. In addition, if there is exactly one server which can serve C_i at t^-, then the MYOPIC policy replaces the content of some other idle server with C_i. Since there are more than $k_1(nT(n))^{1/\beta}$ idle servers and $U(t) < k_1(nT(n))^{1/\beta}$, at t^+, each content type requested in the interval of interest so far, is stored on at least one currently idle server. Therefore, conditioned on F_1 and F_3, every request for a particular content type except the first request, is not deferred.

Hence, putting everything together,

$$\mathbb{P}(F_4) \geq 1 - \mathbb{P}(F_1^c) - \mathbb{P}(F_3^c),$$

thus $\mathbb{P}(F_4) \to 1$ as $n \to \infty$ and the result follows. $\square$

PROOF. (of Corollary 1)
From Theorem 2 we have that, w.h.p., the number of requests deferred by the MYOPIC storage policy is $\mathrm{O}(nT(n))^{1/\beta} = \mathrm{o}(n^{0.5})$ and by Theorem 1, we know that, the expected number of requests deferred by any learning-based storage policy is $\Omega(n^{0.5})$. $\square$

5.3 Proof of Theorem 3

In this section, we first present an outline of the proof of Theorem 3 followed by the proof details.

1. Since we are studying the performance of the MYOPIC policy for the Continuous Change Model, the relative order of popularity of contents keeps changing in the interval of interest. We show that w.h.p., the number of content types which are in the $n^{1/\beta}$ most popular content types at least once in the interval of interest is $\mathrm{O}(n^{1/\beta})$ (Lemma 8).
2. Next, we show that w.h.p., in the interval of interest of length b, only $\mathrm{O}(n^{1/\beta})$ content types are requested (Lemma 9).
3. By Lemma 6 and the proof of Theorem 2, we know that, conditioned on Step 3, the MYOPIC storage policy ensures that in the interval of interest, once a content type is requested for the first time, there is always at least one idle server which can serve an incoming request for that content. Using this, we conclude that, in the interval of interest, only the first request for a particular content type will be deferred. The proof of Theorem 3 then follows from Step 2.

LEMMA 8. *Let G_1 be the event that, in the interval of interest of length b, the number of times that a content among the current top $n^{1/\beta}$ most popular contents changes its position in the popularity ranking is at most $\frac{4b}{\alpha}n^{1/\beta}\nu$. Then, $P(G_1) \geq 1 - o\big(\frac{1}{n}\big)$.*

PROOF. The expected number of clock ticks in b time-units is $bn\nu$. The probability that a change in arrival process involves at least one of the current $n^{1/\beta}$ most popular contents is $\frac{n^{1/\beta}}{\alpha n}$. Therefore, the expected number of changes in arrival process which involve at least one of the current $n^{1/\beta}$ most popular contents is $\frac{2b\nu}{\alpha}n^{1/\beta}$. By the Chernoff bound, we have that $P(G_1) \geq 1 - \mathrm{o}\big(\frac{1}{n}\big)$. $\square$

LEMMA 9. *Let G_2 be the event that in the interval of interest, no more than $\mathrm{O}(n^{1/\beta})$ different types of contents are requested. Then, $\mathbb{P}(G_2^c) = \mathrm{o}\big(\frac{1}{n}\big)$.*

PROOF. Conditioned on the event G_1 defined in Lemma 8, we have that in the interval of interest, at most $\big(\frac{2b}{\alpha}\nu + 1\big)n^{1/\beta}$ different contents are among the top $n^{1/\beta}$ most popular contents. Given this, the proof follows the same lines of arguments as in the proof of Lemma 7. $\square$

PROOF. (of Theorem 3) The proof of the theorem 3 follows from Lemma 9 and uses the same line of arguments as in the proof of Theorem 2. $\square$

5.4 Proof of Theorem 4

To show that GENIE is the optimal policy, we consider the process $X(t)$ which is the number of occupied servers at time t when the storage policy is GENIE. Let $Y(t)$ be the number of occupied servers at time t for some other storage policy $\mathcal{A} \in \mathbb{A}$. We construct a coupled process $(X^*(t), Y^*(t))$ such that the marginal rates of change in $X^*(t)$ and $Y^*(t)$ is the same as that of $X(t)$ and $Y(t)$ respectively.

Recall $\bar{\lambda} = \frac{\sum_{i=1}^{m} \lambda_i}{n}$. At time t, let $\mathbf{C}^{\mathbf{GENIE}}(t)$ and $\mathbf{C}^{\mathcal{A}}(t)$ be the sets of contents stored on idle servers by GENIE and $\mathcal{A}$ respectively. The construction of the coupled process $(X^*(t), Y^*(t))$ is described in Figure 5. We assume that the system starts at time $t = 0$ and $X^*(0) = Y^*(0) = 0$. In this construction, we maintain two counters Z_{X^*} and Z_{Y^*} which keep track of the number of departures from the system. Let $Z_{X^*}(0) = Z_{Y^*}(0) = 0$. Let $\mathrm{Exp}(\mu)$ be an Exponential random variable with mean $\frac{1}{\mu}$ and $\mathrm{Ber}(p)$ be a Bernoulli random variable which is 1 with probability (w.p.) p.

LEMMA 10. *$X^*(t)$ and $Y^*(t)$ have the same marginal rates of transition as $X(t)$ and $Y(t)$ respectively.*

PROOF. This follows from the properties of the coupled process described in Figure 5 using standard arguments. $\square$

LEMMA 11. *Let $D^{(GENIE)}(t)$ be the number of jobs deferred by time t by the GENIE adaptive storage policy and $D^{(\mathcal{A})}(t)$ to be the number of jobs deferred by time t by a policy $\mathcal{A} \in \mathbb{A}$. In the coupled construction, let $W^*(t)$ be the number of arrivals by time t. Let, $D^{X^*}(t) = W^*(t) - Z_{(X^*)}(t) - X^*(t)$ and $D^{Y^*}(t) = W^*(t) - Z_{(Y^*)}(t) - Y^*(t)$. Then, $D^{X^*}(t)$ and $D^{Y^*}(t)$ have the same marginal rates of transition as $D^{(GENIE)}(t)$ and $D^{(\mathcal{A})}(t)$ respectively.*

PROOF. This follows from Lemma 10 due to the fact that $X(t)$ have the same distribution as $X^*(t)$ and the marginal rate of increase of $D^{X^*}(t)$ given $X^*(t)$ is the same as the rate of increase of $D^{(GENIE)}(t)$ given $X(t)$. The result for $D^{Y^*}(t)$ follows by the same argument. $\square$

LEMMA 12. *$X^* \geq Y^*$ for all t on every sample path.*

PROOF. The proof follows by induction. $X^*(0) = Y^*(0)$ by construction. Let $X^*(t_0^-) \geq Y^*(t_0^-)$ and let there be an arrival or departure at time t_0. There are 4 possible cases:

i: If ARR<DEP and $X^*(t_0^-) = Y^*(t_0^-)$, $Y^*(t_0) = Y^*(t_0^-)+1$ only if $X^*(t_0) = X^*(t_0^-) + 1$. Therefore, $X^*(t_0) \geq Y^*(t_0)$.

ii: If ARR<DEP and $X^*(t_0^-) > Y^*(t_0^-)$, $Y^*(t_0) \leq Y^*(t_0^-)+1 \leq X^*(t_0^-) \leq X^*(t_0)$. Therefore, $X^*(t_0) \geq Y^*(t_0)$.

iii: If DEP<ARR and $X^*(t_0^-) = Y^*(t_0^-)$, $X^*(t_0) = Y^*(t_0)$.

iv: If DEP<ARR and $X^*(t_0^-) > Y^*(t_0^-)$, $X^*(t_0) = X^*(t_0^-) - 1 \geq Y^*(t_0^-) \geq Y^*(t_0)$. Therefore, $X^*(t_0) \geq Y^*(t_0)$.

```
1: Generate: ARR $\sim \text{Exp}(n\bar{\lambda})$, DEP $\sim \text{Exp}(\max\{X^*, Y^*\})$
2: $t = t + \min\{\text{ARR},\text{DEP}\}$
3: if ARR<DEP, then
4:   if $(X^* = Y^*)$ then
5:     Generate $u_1 \sim \text{Ber}\left(\frac{\sum_{i\in \mathbf{C}^{\mathbf{GENIE}}(t)} \lambda_i}{n\bar{\lambda}}\right)$
6:     if $(u_1 = 1)$ then
7:       $X^* \leftarrow X^* + 1$
8:       Generate $u_2 \sim \text{Ber}\left(\frac{\sum_{i\in \mathbf{C}^{\mathcal{A}}(t)} \lambda_i}{\sum_{i\in \mathbf{C}^{\mathbf{GENIE}}(\mathbf{t})} \lambda_i}\right)$
9:       if $(u_2 = 1)$ then $Y^* \leftarrow Y^* + 1$
10:    end if
11:  else
12:    Generate $u_1 \sim \text{Ber}\left(\frac{\sum_{i\in \mathbf{C}^{\mathbf{GENIE}}(t)} \lambda_i}{n\bar{\lambda}}\right)$
13:    if$(u_1 = 1)$ then $X^* \leftarrow X^* + 1$
14:    Generate $u_2 \sim \text{Ber}\left(\frac{\sum_{i\in \mathbf{C}^{\mathcal{A}}(t)} \lambda_i}{\sum_{i\in \mathbf{C}^{\mathbf{GENIE}}(\mathbf{t})} \lambda_i}\right)$
15:    if$(u_2 = 1)$ then $Y^* \leftarrow Y^* + 1$
16:  end if
17: else
18:  if $(X^* \geq Y^*)$ then
19:    $X^* \leftarrow X^* - 1$, $Z_{X^*} \leftarrow Z_{X^*} + 1$
20:    Generate $u_3 \sim \text{Ber}\left(\frac{Y^*}{X^*}\right)$
21:    if $(u_3 = 1)$ then $Y^* \leftarrow Y^* - 1$, $Z_{Y^*} \leftarrow Z_{Y^*} + 1$
22:  else
23:    $Y^* \leftarrow Y^* - 1$, $Z_{Y^*} \leftarrow Z_{Y^*} + 1$
24:    Generate $u_4 \sim \text{Ber}\left(\frac{X^*}{Y^*}\right)$
25:    if $(u_4 = 1)$ then $X^* \leftarrow X^* - 1$, $Z_{X^*} \leftarrow Z_{X^*} + 1$
26:  end if
27: end if
28: Goto 1
```

Figure 5: Coupled Process

□

LEMMA 13. *$Z_{X^*} \geq Z_{Y^*}$ for all t on every sample path.*

PROOF. The proof follows by induction. Since the system starts at time $t = 0$, $Z_{X^*}(0) = Z_{Y^*}(0)$. Let $Z_{X^*}(t_0^-) \geq Z_{Y^*}(t_0^-)$ and let there be a departure at time t_0. By Lemma 12, we know that, $X^*(t_0^-) \geq Y^*(t_0^-)$. Therefore, $Z_{X^*}(t_0) \geq Z_{Y^*}(t_0)$ by the coupling construction. □

PROOF. (of Theorem 4)
By Lemmas 12 and 13, for any sample path,

$$X^*(t) + Z_{X^*}(t) \geq Y^*(t) + Z_{Y^*}(t).$$

Therefore, for every sample path, the number of requests already served (not deferred) or being served by the servers by a content delivery system implementing the GENIE policy is more than that by any other storage policy. This implies that for each sample path, the number of requests deferred by GENIE is less than that of any other storage policy. Sample path dominance in the coupled system implies stochastic dominance of the original process. Using this and Lemma 11, we have that,

$$D^{(GENIE)}(t) \leq_{st} D^{(\mathcal{A})}(t).$$

□

6. PROOF OF THEOREM 5

PROOF. (of Theorem 5) The key idea of the GENIE policy is to ensure that at any time t, if the number of idle servers is $k(t)$, the $k(t)$ most popular contents are stored on exactly one idle server each. Since the total number of servers is n, and the number of content-types is $m = \alpha n$ for some constant $\alpha > 1$, all content-types C_i for $i > n$ are never stored on idle servers by the GENIE policy. This means that under the GENIE policy, all arrivals for content types C_i for $i > n$ are deferred. For $\beta > 2$, for all i, $p_i \geq \frac{6}{\pi^2} i^{-\beta}$. The cumulative mass of all content types $i = n+1, ..\alpha n$ is

$$\begin{aligned}\sum_{i=n+1}^{\alpha n} p_i &\geq \sum_{i=k}^{\alpha n} \frac{6}{\pi^2} i^{-\beta} \geq \int_{n+1}^{\alpha n+1} \frac{6}{\pi^2} i^{-\beta} di \\ &\geq \frac{0.54}{\pi^2(\beta-1)} \frac{1}{(n+1)^{\beta-1}},\end{aligned}$$

for n large enough.

Let the length of the interval of interest be b. The expected number of arrivals of types $n+1, n+2, ..\alpha n$, in the interval of interest is at least $\frac{0.54 b\bar{\lambda} n}{\pi^2(\beta-1)} \frac{1}{(n+1)^{\beta-1}}$. Therefore, the expected number of jobs deferred by the GENIE policy in an interval of length b is $\Omega(n^{2-\beta})$. □

7. RELATED WORK

Our model of content delivery systems shares several features with recent models and analyses for content placement and request scheduling in multi-server queueing systems [10, 11, 15, 18]. All these works either assume known demand statistics, or a low-dimensional regime (thus permiting "easy" learning). Our study is different in its focus on unknown, high-dimensional and time-varying demand statistics, thus making it difficult to consistently estimate statistics. Our setting also shares some aspects of estimating large alphabet distributions with only limited samples, with early contributions from Good and Turing [6], to recent variants of such estimators [13,16].

Our work is also related to the rich body of work on the content replication strategies in peer-to-peer networks, e.g., [3, 4, 8, 9, 12, 14, 22, 25, 26]. Replication is used in various contexts: [14] utilizes it in a setting with large storage limits, [9, 12] use it to decrease the time taken to locate specific content, [3,25,26] use it to increase bandwidth in the setting of video streaming, and [4] uses it to minimize the number of hosts that need to be probed to resolve a query for a file in unstructured peer-to-peer networks.

However, the common assumption is that the number of content-types does not scale with the number of peers, and that a request can be served in parallel by multiple servers (and with increased network bandwidth as the number of peers with a specific content-type increases) which is fundamentally different from our setting.

Finally, our work is also related to the vast literature on content replacement algorithms in server/web cache management. As discussed in [19], parameters of the content (e.g., how large is the content, when was it last requested) are used to derive a cost, which in-turn, is used to replace content. Examples of algorithms that have a cost-based interpretation include the Least Recently Used (LRU) policy, the Least Frequently Used (LFU) policy, and the Max-Size policy [20]. We refer to [19] for a survey of web caching schemes. There is a huge amount of work on the performance of replication strategies in single-cache systems; however the analysis of adaptive caching schemes in distributed cache systems under stochastic models of arrivals and departures is very limited.

8. ACKNOWLEDGEMENTS

We acknowledge the support of ** grants ***.

9. CONCLUSIONS

In this paper, we considered the high dimensional setting where the number of servers, the number of content-types, and the number of requests to be served over any time interval all scale as $O(n)$; further the demand statistics are not known a-priori. This setting is motivated by the enormity of the contents and their time-varying popularity which prevent the consistent estimation of demands.

The main message of this paper is that in such settings, separating the estimation of demands and the subsequent use of the estimations to design optimal content placement policies ("learn-and-optimize" approach) is order-wise suboptimal. This is in contrast to the low dimensional setting, where the existence of a constant bound on the number of content-types allows asymptotic optimality of a learn-and-optimize approach.

10. REFERENCES

[1] M. Ahmed, S. Traverso, M. Garetto, P. Giaccone, E. Leonardi, and S. Niccolini. Temporal locality in today's content caching: why it matters and how to model it. *ACM SIGCOMM Computer Communication Review*, 43(5):5–12, October 2013.

[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. In *IEEE INFOCOM'99*, pages 126–134, 1999.

[3] D. Ciullo, V. Martina, M. Garetto, E. Leonardi, and G. Torrisi. Stochastic analysis of self-sustainability in peer-assisted VoD systems. In *IEEE INFOCOM*, pages 1539–1547, 2012.

[4] E. Cohen and S. Shenker. Replication strategies in unstructured peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 177–190. ACM, 2002.

[5] P. Gill, M. Arlitt, Z. Li, and A. Mahanti. YouTube traffic characterization: A view from the edge. In *7th ACM SIGCOMM Conference on Internet Measurement*, pages 15–28, 2007.

[6] I. J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264, 1953.

[7] A. Iamnitchi, M. Ripeanu, and I. Foster. Small-world file-sharing communities. In *IEEE INFOCOM*, March 2004.

[8] J. Kangasharju, K. Ross, and D. Turner. Optimizing file availability in peer-to-peer content distribution. In *INFOCOM*, 2007.

[9] J. Kangasharjua, J. Roberts, and K. Ross. Object replication strategies in content distribution networks. *Computer Communications*, 25:376–383, 2002.

[10] M. Leconte, M. Lelarge, and L. Massoulie. Bipartite graph structures for efficient balancing of heterogeneous loads. In *the 12th ACM SIGMETRICS Conference*, pages 41–52, 2012.

[11] M. Leconte, M. Lelarge, and L. Massoulie. Adaptive replication in distributed content delivery networks. *Preprint*, 2013.

[12] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *16th international conference on Supercomputing*, 2002.

[13] A. McAllester and R. Schapire. On the convergence rate of Good-Turing estimators. In *COLT Conference*, pages 1 – 6, 2000.

[14] B. Tan and L. Massoulie. Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM Trans. Networking*, 21:566–579, 2013.

[15] J. Tsitsiklis and K. Xu. Queueing system topologies with limited flexibility. In *SIGMETRICS '13*, 2013.

[16] G. Valiant and P. Valiant. Estimating the unseen: An n/log (n)-sample estimator for entropy and support size, shown optimal via new clts. In *Proceedings of the 43rd annual ACM Symposium on Theory of Computing*, pages 685–694, 2011.

[17] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin. A hierarchical characterization of a live streaming media workload. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 117–130, 2002.

[18] R. B. Wallace and W. Whitt. A staffing algorithm for call centers with skill-based routing. *Manufacturing and Service Operations Management*, 7:276–294, 2007.

[19] J. Wang. A survey of web caching schemes for the Internet. *ACM SIGCOMM Computer Communication Review*, 29:36–46, 1999.

[20] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox. Removal policies in network caches for world-wide web documents. In *SIGCOMM'96*, 1996.

[21] S. Williams, M. Abrams, C. Standridge, G. Abdulla, and E. Fox. Caching proxies: limitations and potentials. In *the 4th International WWW Conference*, December 1995.

[22] W. Wu and J. Lui. Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation. *IEEE Transactions on Parallel and Distributed Systems*, 23, August 2012.

[23] www.youtube.com/yt/press/statistics.html.

[24] H. Yu, D. Zheng, B. Zhao, and W. Zheng. Understanding user behavior in large scale video-on-demand systems. In *EuroSys*, April 2006.

[25] X. Zhou and C. Xu. Optimal video replication and placement on a cluster of video-on-demand servers. In *International Conference on Parallel Processing*, pages 547–555, 2002.

[26] Y. Zhou, T. Fu, and D. Chiu. On replication algorithm in P2P-VoD. *IEEE/ACM Transactions on Networking*, pages 233 – 243, 2013.