Optimizing TTL Caches under Heavy-Tailed Demands

Andrés Ferragut
Universidad ORT Uruguay
ferragut@ort.edu.uy

Ismael Rodríguez
Universidad ORT Uruguay
rodriguez_i@ort.edu.uy

Fernando Paganini
Universidad ORT Uruguay
paganini@ort.edu.uy

ABSTRACT

In this paper we analyze the hit performance of cache systems that receive file requests with general arrival distributions and different popularities. We consider timer-based (TTL) policies, with differentiated timers over which we optimize. The optimal policy is shown to be related to the monotonicity of the hazard rate function of the inter-arrival distribution. In particular for decreasing hazard rates, timer policies outperform the static policy of caching the most popular contents. We provide explicit solutions for the optimal policy in the case of Pareto-distributed inter-request times and a Zipf distribution of file popularities, including a compact fluid characterization in the limit of a large number of files. We compare it through simulation with classical policies, such as least-recently-used and discuss its performance. Finally, we analyze extensions of the optimization framework to a line network of caches.

1. INTRODUCTION

Caching systems are widely deployed in computer networks to improve performance. By storing frequently requested content near end users, latency and access time are reduced and a lighter load is imposed on network resources. In recent years, with the exponential growth in available content in the Internet, caching has become an integral component of network architectures. Content Delivery Networks (CDNs) have become the standard approach used by content providers to serve large client populations. Current trends take this point further, using terms such as Content-Centric Networking (CCN) [21] or Information-Centric Networking (ICN) [1] to describe architectures designed to connect end users directly to content instead of a traditional server.

With the advent of these new technologies, there is a renewed interest in the performance analysis of caching systems, which has old roots in studies of CPU processing and memory paging [28]. Most models for cache performance analysis fall into two categories; the first are *replacement policies*, where a finite amount of memory is actively administered to maintain the most requested contents at hand. Within these, the Least-Frequently-Used (LFU) policy is known to be optimal under independent requests, albeit requiring a large state. A simpler algorithm is the Least-Recently-Used (LRU) policy [11, 28], where upon arrival of a new request not presently stored, the object not requested for the longest time is evicted from the cache. Exact analysis of this policy has proven difficult, even with simple traffic models such as the *independent reference model (IRM)*, where successive requests are assumed independent and stationary. Further references on LRU models are given in Section 2.

Another class of cache management algorithms are *timer based* policies (Time-to-live, TTL). In this case the eviction of a file from the cache is performed after the expiration of some timer, which is independently set upon arrival of requests for each content. The main advantage of these policies is that eviction is decoupled amongst objects, thus simplifying analysis of the system. This simple policy also guarantees weak consistency, providing a bound on the storage of outdated content, which is the reason why it has been implemented in DNS systems and Web caching. The main disadvantage is that now memory usage can only be bounded on average. Another important point is that TTL caches with fixed timers can be used to approximate the behavior of the LRU system, as pointed out by Che et. al in [10]. This approximation procedure has been further developed in recent papers which we review in Section 2.

In this paper we study the *optimal* TTL policy for a cache, in the sense of maximizing the hit rate obtained, over the choice of timer parameters. Building upon the models in [5, 16, 29] and with very general hypotheses on the request arrival processes, we formulate the optimal timer policy as a nonlinear optimization program in Section 3. The key characteristic behind the structure of the optimal policy turns out to be the *hazard rate* function of the inter-request times. For the simple case where arrivals come from a Poisson process (constant hazard rate), it is well known that the optimal policy is to statically store the most popular files. A first contribution of this paper is to show that this holds as well for any interarrival distribution with *increasing* hazard rate. The most interesting case is, however, when hazard rates are *decreasing*; here the optimization problem can be transformed into a convex program, and the optimal policy characterized as choosing timer values that *equalize* the hazard rates of the stored contents. This implies in particular that the static policy (and hence, the LFU policy) is not always optimal. These results are described in Section 4.

SIGMETRICS '16, June 14-18, 2016, Antibes Juan-Les-Pins, France

DOI: http://dx.doi.org/10.1145/2896377.2901459

The decreasing hazard rate case is important because it appears when inter-request distributions are heavy tailed, a typical model of bursty behavior. In Section 5 we develop this case in detail, assuming Pareto inter-arrival times, in combination with a Zipf power-law distribution for file popularities, also a commonly used model. The optimal policy is identified as a function of the tail parameters, where in particular no contents are permanently cached. A streamlined characterization of this optimal policy in the case of large number of files is obtained through a fluid limit, with explicit analytic expressions in terms of the model parameters. In particular, we show that the maximal hit rate achievable by TTL policies is strictly less than 1 for power-law popularities with tail parameter $\beta < 1$. For the case $\beta \geq 1$, we show that the maximal hit rate approaches 1 and that also the static policy is asymptotically optimal. We also analyze the memory usage of the TTL policy, showing through a law of large numbers that the cache operates close to its average capacity.

In Section 6 we evaluate the performance of the optimal TTL policies by simulation experiments. We validate our results and analyze the speed of convergence to the asymptotic limits. We also provide a practical variant of the TTL policy that enforces constant memory usage, with small deviations from the ideal case. Comparisons with static (LFU) policies as well as LRU replacement policies are carried out. The main observation is that both classical policies may be suboptimal, but each performs well in a certain regime which we can characterize through the hazard rate function of the request process.

In Section 7 we include some generalizations. In the first place we discuss how to generalize our optimization framework to consider timers which are themselves random, in particular exponentially distributed. We also investigate the situation of a line of caches, where contents displaced from one are moved upstream to the next. In the exponential and deterministic timer cases, we show how this situation can be analyzed through an appropriate Markov model, and results are given on how to optimize performance.

Conclusions are provided in Section 8. Some of the proofs can be found in the extended version of this paper [13].

2. BACKGROUND AND RELATED WORK

The performance analysis of caching algorithms goes back to the seminal works of [19, 28] on replacement algorithms. Despite the simplicity of replacement policies, analysis becomes complex even for a single cache: in [28], the author gives an explicit expression of the hit probability of the LRU policy, with exponential complexity. In [19], it is shown that under the IRM assumption, similar replacement policies such as first-in-first-out achieve the same hit probability. Since exact computation is intractable, [11] provides an approximate computational procedure, again under the IRM assumption. This method has been further extended recently by [30] to the case of networks of cache systems. In [18], another approach is given for list-based replacement policies.

Another line of work related to our results is the asymptotic characterization of cache performance. A first step in this direction is [14], which addresses the limit cost of the Move-to-Front rule in self-organizing lists. By exploiting the connection between MTF and the LRU policy, [23] provides an asymptotic expression for the hit probability under IRM assumptions and Zipf popularities with parameter $\beta > 1$. These results are extended in [25] to the case $\beta \leqslant 1$ and some LRU generalizations. Following a fundamentally different approach, the same limit result is obtained in [4] by using the Laplace transform of the asymptotic search cost. In our work, we follow a similar limit procedure to characterize the performance of the optimal TTL policies. Going beyond IRM, [22, 24, 26] show some insensitivity properties of LRU in the asymptotic regime, with dependent requests and $\beta > 1$, and also provide conditions for the capacity scaling for such insensitivity to hold. In this paper, we will show that in fact several policies achieve an optimal hit rate in this scenario, but for $\beta < 1$ temporal locality may have an impact.

By far, the most popular technique for analyzing cache performance of LRU under IRM is the so called "Che approximation", introduced in [10] in the context of Web caching. Here, a *characteristic time* is defined for the cache system which amounts to the average eviction time of the files. The authors of [17] provide additional evidence on why the approximation is good through a central limit argument. It turns out that this approximation amounts to choosing a uniform timer to approximate LRU behavior.

The analysis of policies based on timers goes back to [27] who gave expressions for the steady state hit probabilities, later extended in [3] to include update delay. More recently, [15, 16] laid the foundations for analyzing TTL policies under deterministic and random timers with general arrival processes. In [5] the analysis is extended to a family of TTL policies with the focus of approximating LRU performance in linear and tree networks, and a different policy is proposed in [6] with the aim of maximizing hit ratios by variance reduction. In [7] the Che approximation is analyzed in a more general setting and in [29], it is extended, using TTL cache tools, to renewal arrivals, showing good accuracy in the case of small caches and Zipf popularities. In particular, temporal locality in requests is observed to have an impact in cache performance. A reverse engineering approach based on utility maximization links TTL policies with list based ones, as recently shown in [12].

Building upon these models, the main contribution of our paper is to identify the structure of the optimal TTL based policy, give an exact analysis of the asymptotic behavior in the case of heavy tailed arrivals and show that it can outperform static and LRU policies in this setting.

3. SYSTEM MODEL

We now introduce our TTL cache model with the aim of determining the optimal TTL policy. We build on the models of [5, 16, 29] and introduce a unified notation for the system. The main assumption under TTL caches that simplifies analysis is that per content metrics can be dealt with separately. We first derive these metrics and then introduce our definition for the optimal policies.

The main tradeoff we want to analyze in these systems is between the hit probability, which is a property of the arriving requests and how they find the system upon arrival, and the memory occupation in steady state, which is a system property. Following [16], the correct mathematical setting to deal with the analysis of these metrics is the Palm formulation for stationary point process in the line [2], which enables to compute system-wide and job-related measures

by conditioning upon arrivals. We follow this approach below to obtain the relevant metrics.

Consider a cache system that receives requests for files or content. Assume that requests for content n arrive into the system as a stationary point process [2] on $\mathbb{R}$ with intensity λ_n. Let $\{\tau_k^{(n)}\}_{k\in\mathbb{Z}}$ denote the arrival times, with the usual numbering convention $\tau_0^{(n)} \leqslant 0 < \tau_1^{(n)}$, and $X_k^{(n)} = \tau_{k+1}^{(n)} - \tau_k^{(n)}$ denote the interarrival times for content n. Define:

$$F_n(t) = P_0^{(n)}\left(X_k^{(n)} \leqslant t\right) \tag{1}$$

the interarrival time distribution in the Palm space of the arrival process, which is independent of k due to stationarity. We have:

$$E_0^{(n)}[X_k^{(n)}] = \int_0^\infty 1 - F_n(t)dt = \frac{1}{\lambda_n}.$$

The TTL policy is as follows: for every arrival k, a timer $T_n \in [0,\infty]$ is chosen; for most of this paper it will be assumed deterministic. Upon arrival of a request, the file is stored in the cache for a time T_n: if the next arrival occurs before T_n then the next request will be a *hit* and the timer is reset. If the interarrival time exceeds T_n the next request will be a *miss*, implying a performance penalty due to the need of retrieving the content from a remote repository.

From the above assumptions, the stationary hit probability is thus:

$$h_n := P_0^{(n)}\left(\tau_1^{(n)} \leqslant T_n\right) = P_0^{(n)}\left(X_0^{(n)} \leqslant T_n\right) = F_n(T_n). \tag{2}$$

From a system perspective, the main concern is cache occupation. Define by $Z_n(t) \in \{0,1\}$ the stationary process indicating whether content n is stored at time t, i.e.:

$$Z_n(t) = \mathbf{1}_{\{\tau_{k(t)}^{(n)} + T_n > t\}},$$

where $\tau_{k(t)}^{(n)}$ is the last request before time t. The stationary occupation probability is given by $u_n := E[Z_n(t)]$. Due to stationarity, this probability can be computed at time 0 by using the Palm inversion formula [2]:

$$E[Z_n(0)] = \lambda_n E_0^{(n)}\left[\int_0^{\tau_1^{(n)}} Z_n(s)ds\right]$$

Noting that under the Palm space $\tau_0^{(n)} \equiv 0$, $Z_n(s) = \mathbf{1}_{\{T_n > s\}}$ for $s \in [0, \tau_1^{(n)}]$ and therefore:

$$\begin{aligned} u_n = E[Z_n(0)] &= \lambda_n E_0^{(n)}\left[\int_0^{\tau_1^{(n)}} \mathbf{1}_{\{T_n>s\}}ds\right] \\ &= \lambda_n E_0^{(n)}\left[\min\{X_0^{(n)}, T_n\}\right] \\ &= \lambda_n \int_0^{T_n} 1 - F_n(s)ds =: \hat{F}_n(T_n). \end{aligned} \tag{3}$$

Here $\hat{F}_n(t)$ is the well known *residual lifetime* distribution associated to F_n. Equation (3) simply states that the stationary probability u_n of storing the content is just the probability that the time elapsed since the last request is less than the timer T_n.

We are now ready to state the problem: consider a TTL cache system where requests come from independent arrival processes for each file. Let N be the total number of files, each having a popularity q_n stemming from a probability distribution, i.e. $\sum_{n=1}^N q_n = 1$. Without loss of generality, we shall assume that the files are labeled in decreasing order of popularity, i.e. q_n is a strictly decreasing sequence. We assume that the arriving requests for content n form a stationary and simple point process of intensity $\lambda_n = \lambda q_n > 0$. Here $0 < \lambda < \infty$ represents the total request rate into the system. Each content is assigned a timer value $T_n \in [0,\infty]$ possibly dependent on n.

From equation (2), the overall hit rate in the system is given by:

$$H(T_1,\ldots,T_n) := \sum_{n=1}^N \lambda_n F_n(T_n). \tag{4}$$

Suppose now the system should maintain a certain occupation level, due to memory constraints. One way to keep cache occupation at bay is to impose:

$$E\left[\sum_{n=1}^N Z_n(0)\right] \leqslant C,$$

where $C \leqslant N$ is the average memory allocated to the cache. Typically one should have $C \ll N$. Using eq. (3) the above constraint translates into:

$$\sum_n \hat{F}_n(T_n) \leqslant C \tag{5}$$

Therefore, the optimal TTL policy should solve the following optimization in the timer variables $T_1,\ldots,T_N$:

PROBLEM 1 (TTL CACHE OPTIMIZATION).

$$\begin{aligned} &\max \sum_{n=1}^N \lambda_n F_n(T_n), \\ &s.t.: \sum_n \hat{F}_n(T_n) \leqslant C. \end{aligned}$$

Problem 1 is a non-linear optimization with a non-linear constraint. Note also that typical timer policies that are feasible for Problem 1 include:

- The *static* policy that stores the C most popular contents, i.e. $T_n = \infty$ for $1 \leqslant n \leqslant C$ and $T_n = 0$ otherwise.
- The *homogeneous timer policy*, used in [10, 29] to approximate cache systems operating under the *Least Recently Used (LRU)* replacement algorithm, which amounts to choose $T_n \equiv T_C$ as the solution of:

$$\sum_{n=1}^N \hat{F}_n(T_C) = C \tag{6}$$

We want to characterize the optimal timer policy in terms of the interarrival distribution characteristics. To this end, it is useful to express Problem 1 in terms of the occupation probabilities by performing the change of variables:

$$u_n = \hat{F}_n(T_n) \Leftrightarrow T_n = \hat{F}_n^{-1}(u_n),$$

where $t = \hat{F}_n^{-1}(u) := \inf\{t : \hat{F}_n(t) \geqslant u\}$ is the generalized inverse function. With the change of variables Problem 1 becomes:

PROBLEM 2.

$$\max_{u_n\in[0,1]} \sum_{n=1}^{N} \lambda_n F_n(\hat{F}_n^{-1}(u_n)),$$
$$s.t.: \sum_{n=1}^{N} u_n \leqslant C.$$

Now we are dealing with a non-linear optimization with a linear constraint. We have the following:

LEMMA 1. *Let $F_n(t)$ have a density $f_n(t)$ in its support $[0, F_n^{-1}(1)]$ and define*

$$\eta_n(t) := \frac{f_n(t)}{1-F_n(t)}, \quad t \in [0, F_n^{-1}(1)). \tag{7}$$

Then:

$$\frac{\partial}{\partial u_n} F_n(\hat{F}_n^{-1}(u_n)) = \frac{1}{\lambda_n}\eta_n\left(\hat{F}_n^{-1}(u_n)\right) = \frac{1}{\lambda_n}\eta_n(T_n). \tag{8}$$

PROOF. From eq. (3) we have that:

$$\frac{\partial}{\partial T_n}\hat{F}_n(T_n) = \lambda_n(1-F_n(T_n)).$$

Applying the chain-rule and the inverse function theorem to the continuously differentiable function $\hat{F}_n$ we then have:

$$\begin{aligned}\frac{\partial}{\partial u_n} F_n(\hat{F}_n^{-1}(u_n)) &= f_n(\hat{F}_n^{-1}(u_n))\frac{\partial}{\partial u_n}\hat{F}_n^{-1}(u_n)\\ &= \frac{f_n(\hat{F}_n^{-1}(u_n))}{\lambda_n(1-F_n(\hat{F}_n^{-1}(u_n)))},\end{aligned}$$

as stated. □

The function $\eta_n(t)$ is the well known *hazard rate* function associated to F_n and will play an important role in what follows.

4. OPTIMAL TIMER POLICY

We now discuss the structure of the optimal TTL policy under further assumptions on the hazard rate function. We begin with the simplest case where arrivals form a Poisson process.

4.1 Constant hazard-rate: the Poisson model

Assume requests for each file arrive according to a Poisson process of intensity $\lambda_n = \lambda q_n$. In this case $F_n(t) = \hat{F}_n(t) = 1 - e^{-\lambda q_n t}$, which is just a restatement of the memoryless property of the exponential distribution. Therefore, the hit and occupation probabilities coincide, $h_n = u_n$, as expected due to the PASTA property of Poisson arrivals. Problem 2 becomes:

$$\max_{u_n\in[0,1]} \sum_{n=1}^{N} \lambda_n u_n,$$
$$\text{s.t.: } \sum_{n=1}^{N} u_n \leqslant C.$$

Since this is a linear program, the solution should be attained at a vertex of the feasible region [8]. Since the weights $\lambda_n = \lambda q_n$ are strictly decreasing it is readily verified that the unique optimum is:

$$u_n^* = 1 \quad \text{for } 1 \leqslant n \leqslant C$$

and 0 otherwise. In the original timer variables, $T_n = \infty$ for the C most popular contents and 0 otherwise, i.e. the optimal policy is the static policy where the C most popular contents are stored. This result is well known in the context of the *independent reference model (IRM)* where the sequence of requests is *iid*. Under Poisson arrivals, the memoryless property ensures this is in fact the case, and thus we recover the static policy in the optimum.

The underlying fact here is that Problem 2 becomes linear because the hazard rate function of the exponential distribution is constant. We now analyze two cases where we deviate from this assumption, and hence of the IRM model.

4.2 The increasing hazard rate case

Assume now that the arrival processes are such that $\eta_n(t)$ is increasing in t within the support (IHR). In this case, once some time t has elapsed since a request it becomes increasingly likely to have a new request for the same file. This is associated with request processes that are in some sense more periodical than the Poisson process. A simple example is when $X_k^{(n)}$ follows an Erlang distribution with $m > 1$ stages, or the extreme case where $X_k^{(n)}$ approaches the deterministic distribution concentrated in $1/\lambda_n$.

Applying Lemma 1, if $\eta_n(t)$ is increasing, then:

$$\frac{\partial h_n(u_n)}{\partial u_n} = \frac{1}{\lambda_n}\eta_n\left(\hat{F}_n^{-1}(u_n)\right)$$

is increasing in u_n, since $\hat{F}_n^{-1}(u_n)$ is also increasing. Therefore, the objective function of Problem 2 is convex.

We are thus faced with *maximizing* a convex function over a convex domain. It is easy to see that also in this case there is a solution at an extremal point of the convex set: if u is not an extremal point, it can be written as a convex combination of two points in the set, and thus the function value should be less that the maximum value at the extremes.[1]

The extreme points of the set are again the static policies where $u_n = 1$ or 0. Since the support of the interarrival and lifetime distributions coincide, $F_n(F_n^{-1}(u_n)) = u_n$ whenever $u_n \in \{0, 1\}$. Therefore, the hit rate at the extreme policies are simply the sum of the arrival rates of the stored files, and again the optimal policy is to store the most popular contents statically. We have thus proved:

THEOREM 1. *Provided that the arrival distributions satisfy the increasing hazard rate (IHR) property, the optimal TTL caching policy defined by Problem 1 is to statically store the C most popular contents.*

In this case, the optimal hit rate is given by:

$$H^* = \sum_{n=1}^{C} \lambda_n. \tag{9}$$

4.3 The decreasing hazard rate case

We now turn our attention to the case where the hazard rates $\eta_n(t)$ are (strictly) decreasing in t (DHR). This assumption implies that after some time t has elapsed, requests are less and less likely to come. This type of interarrival distribution is associated with heavy tailed arrival processes (e.g. Pareto). This constitutes an important case

[1]The optimum is necessarily extremal if the function is strictly convex, which occurs for strictly increasing hazard rates.

in practice, since it can be used to model requests that come in "bursts" separated by a few very long interarrival times.

In this case, due to Lemma 1, the objective function in Problem 2 is strictly *concave* and we thus have a proper convex optimization problem. Let us write the Lagrangian of Problem 2:

$$\mathcal{L}(u,p) = \sum_n \lambda_n F_n\left(\hat{F}_n^{-1}(u_n)\right) - p\left(\sum_n u_n - C\right), \quad (10)$$

where $p \geqslant 0$ is the multiplier associated with the memory constraint. The saddle point conditions for the optimal u^* and the associated multiplier p^* are: $\sum_n u_n^* \leq C$, $p^* \geq 0$,

$$u_n^* = \underset{u_n\in[0,1]}{\operatorname{argmax}}\, \mathcal{L}(u,p^*) \quad \forall n, \quad (11a)$$

$$p^*\left(\sum_n u_n^* - C\right) = 0. \quad (11b)$$

Applying Lemma 1, we have:

$$\frac{\partial \mathcal{L}}{\partial u_n} = \eta_n\left(\hat{F}_n^{-1}(u_n)\right) - p.$$

A first observation is that, if $C < N$, the multiplier p at the saddle point must be strictly positive, since otherwise $\mathcal{L}$ is increasing in $u_n \in [0,1]$ and therefore $u_n^* = 1\ \forall n$, violating the constraint. Given this, we have three possibilities:

$$\begin{cases} \eta_n(0) \leqslant p^* & \Longrightarrow \quad u_n^* = 0, \\ \eta_n(0) > p^* \geqslant \eta_n(\infty) & \Longrightarrow \quad \eta_n\left(\hat{F}_n^{-1}(u_n^*)\right) = p^*, \\ \eta_n(\infty) > p^* & \Longrightarrow \quad u_n^* = 1. \end{cases} \quad (12)$$

Expressing these conditions in terms of the original TTL values, we have proved:

THEOREM 2. *Provided that the arrival distributions satisfy the decreasing hazard rate (DHR) property, the optimal TTL caching policy defined by Problem 1 is such that:*

$$\eta_n(T_n^*) \equiv \text{constant} \quad (13)$$

for every n such that $0 < T_n^* < \infty$.

REMARK 1. *The optimality condition* (13) *has the following economic interpretation: $\eta_n(T_n)$ is the marginal utility derived from increasing the timer to $T_n + dt$, since this is the probability that an arrival will occur immediately after T_n. For a given budget, then the optimal allocation is to equalize the marginal gains whenever possible.*

Despite this characterization, it is difficult to give an explicit expression for the timer values without choosing a suitable parametric family for the problem primitives F_n and q_n. We now deal with an important case that can be solved in detail.

5. HEAVY TAILED ARRIVALS AND POPULARITIES.

To obtain further results we now focus on a specific parametric form for the inter-arrival time distributions and file popularities. In both we will consider heavy-tailed distributions, reflecting the burstiness in time of demands for a specific content, and the natural fact that content files have widely disparate popularities. For simplicity we assume from now on that the total arrival rate of requests is $\lambda = 1$, which amounts to a choice of units.

A typical distribution for content popularities is the well known Zipf(β), defined by:

$$\lambda_n = \frac{1}{n^\beta S_N(\beta)}, \quad n = 1,\ldots,N; \quad (14)$$

where for the normalizing constant we are using the notation $S_N(\gamma) := \sum_{m=1}^N \frac{1}{m^\gamma}$.

Note that for $\gamma \in (-\infty, 1)$ when the series diverges we have the equivalent

$$S_N(\gamma) = \sum_{m=1}^N \frac{1}{m^\gamma} \sim \frac{N^{1-\gamma}}{1-\gamma} \quad \text{as } N \to \infty. \quad (15)$$

For the distribution F_n of inter-arrival times in requests for file n, we adopt a Pareto distribution with with parameter $\alpha > 1$, namely:

$$F_n(t) = 1 - \left(\frac{\theta_n}{\theta_n + t}\right)^\alpha.$$

Here θ_n is a scale-parameter, that must satisfy $\lambda_n = \frac{\alpha-1}{\theta_n}$ to be consistent with the mean arrival rate. It follows directly from the definition in (3) that $\hat{F}_n$ is also Pareto, given by:

$$\hat{F}_n(t) = 1 - \left(\frac{\theta_n}{\theta_n + t}\right)^{\alpha-1}.$$

Applying the change of variables of the preceding Section we obtain the convex problem:

$$\max_{u_n\in[0,1]} \sum_{n=1}^N \lambda_n\left[1 - (1-u_n)^{\frac{\alpha}{\alpha-1}}\right], \quad (16)$$

$$\text{s.t.:} \ \sum_{n=1}^N u_n \leqslant C. \quad (17)$$

We assume as before that $C < N$. Later on we will let the problem size N grow and choose a specific growth rate for $C(N)$.

The Lagrangian of our problem with respect to the capacity constraint is

$$\mathcal{L}(u,p) = \sum_{n=1}^N \left\{\lambda_n\left[1 - (1-u_n)^{\frac{\alpha}{\alpha-1}}\right] - pu_n\right\} + pC,$$

where $p \geqslant 0$. We have

$$\frac{\partial \mathcal{L}}{\partial u_n} = \lambda_n \frac{\alpha}{\alpha-1}(1-u_n)^{\frac{1}{\alpha-1}} - p.$$

As before, at the saddle point $p > 0$, otherwise $u_n = 1\ \forall n$, which would violate the constraint. Given $p > 0$, we have two possibilities for the optimal u_n^*, corresponding to the two first cases in (12):

$$\begin{cases} \lambda_n \frac{\alpha}{\alpha-1} \leqslant p & \Rightarrow \ u_n^* = 0; \\ \lambda_n \frac{\alpha}{\alpha-1} > p & \Rightarrow \ u_n^* = 1 - \left(\frac{p(\alpha-1)}{\alpha\lambda_n}\right)^{\alpha-1}. \end{cases} \quad (18)$$

In this case, $u_n^* < 1$ always, so it is never optimal to store a file all the time.

Also, at least one file must be stored with positive probability, so the second case in (18) must hold for the largest λ_n, which occurs at $n = 1$; it will continue to hold until a maximum value

$$M := \max\left\{1 \leqslant n \leqslant N : \lambda_n = \frac{1}{n^\beta S_N(\beta)} > p\frac{(\alpha-1)}{\alpha}\right\}, \quad (19)$$

which corresponds to the number of files that are cached a nonzero amount of time in the optimal policy. The remaining files, if any, are never stored. Note that

$$C = \sum_{m=1}^{M} u_m^* < M, \tag{20}$$

so the number of stored files always exceeds capacity. In practice, if capacity is large the system will store all files a positive amount of time ($M = N$), but in a situation of more scarcity it is optimal to store only a number $M < N$ of objects.

5.1 Asymptotic behavior of the optimal policy

A more streamlined representation of the optimal storage policy can be obtained by taking the limit in the problem size, $N \to \infty$, where it is natural to also let the capacity grow, $C_N \to \infty$. A first observation that follows from (20) is that $M_N \to \infty$. This holds regardless of the growth rate of C_N.

From now on we will focus mainly on the case $C_N = cN$, $c < 1$, i.e. the cache size grows linearly with the number of files. We will characterize the asymptotic behavior of the optimal storage policy and the resulting cost. To analyze the growth rate of the resulting M_N we define

$$x_0 := \limsup_{N\to\infty} \frac{M_N}{N}. \tag{21}$$

Clearly from (20) we have $x_0 \in [c, 1]$. The following proposition provides an exact characterization. Its proof can be found in the extended version of this paper [13].

PROPOSITION 1. *$x_0 = \min\{\frac{c}{c_{\alpha,\beta}}, 1\}$, where*

$$c_{\alpha,\beta} := \frac{\beta(\alpha-1)}{1+\beta(\alpha-1)}. \tag{22}$$

Also the lim sup *in* (21) *is an actual limit.*

We thus characterize the optimal fraction of contents that get stored a positive amount of time in the system, in the asymptotic limit. There are two relevant cases, corresponding to those in the preceding example, characterized by the structural threshold $c_{\alpha,\beta} < 1$ defined in (22).

If $c < c_{\alpha,\beta}$, only a strict fraction $x_0 < 1$ of the objects will be cached. If $c > c_{\alpha,\beta}$, then $x_0 = 1$, furthermore in the proof we show that for large N we have $M_N = N$, i.e. it is optimal to cache all contents. In the border case we could have convergence to unit fraction from below, a case we will for simplicity not consider further.

5.1.1 Asymptotic file caching probabilities

Having characterized asymptotically how many objects are cached, we now turn to the fraction of time they must be stored. We begin by recalling the optimal solution policy (18), reproduced below with emphasis on the dependence with N:

$$u_n^{*,N} = \left[1 - \left(\frac{p_N S_N(\beta)(\alpha-1)}{\alpha} n^{\beta}\right)^{\alpha-1}\right] \mathbf{1}_{\{n \leqslant M_N\}}. \tag{23}$$

In order to characterize its asymptotic behavior in N, we will express this policy in terms of a continuous variable that represents the fraction n/N of the popularity rank. Specifically, define the function $u^{*,N}(x) = u^{*,N}_{\lfloor xN \rfloor}$, $x \in [0, 1]$.

The following theorem characterizes the limiting behavior of this function.

THEOREM 3. *If $C_N = cN$ for $0 < c < 1$, then as $N \to \infty$, $u^{*,N}(x)$ converges pointwise to*

$$u^*(x) = \left[1 - \kappa x^{(\alpha-1)\beta}\right]^+, \tag{24}$$

where $[w]^+ = \max\{w, 0\}$ and the constant κ is such that the following constraint holds:

$$\int_0^1 u^*(x)dx = c. \tag{25}$$

The proof is given in [13], by distinguishing the two cases present in the previous discussion. In particular it is shown that if $c < c_{\alpha,\beta}$ we have

$$u^*(x) = \left[1 - \left(\frac{x}{x_0}\right)^{\beta(\alpha-1)}\right] \mathbf{1}_{\{x<x_0\}}; \tag{26}$$

on the other hand if $c \geq c_{\alpha,\beta}$ the result spreads over $[0, 1]$:

$$u^*(x) = 1 - (1-c)(1+\beta(\alpha-1))x^{\beta(\alpha-1)}. \tag{27}$$

In Figure 1 below we show the asymptotic optimal policy functions for fixed values of $\alpha = 2$, $\beta = 0.8$ ($c_{\alpha,\beta} \approx 0.44$), and increasing values of the cache size parameter c. For comparison, we also plot the corresponding optimal policy calculated numerically for a catalog of $N = 1000$ files, showing that the approximation is indeed good.

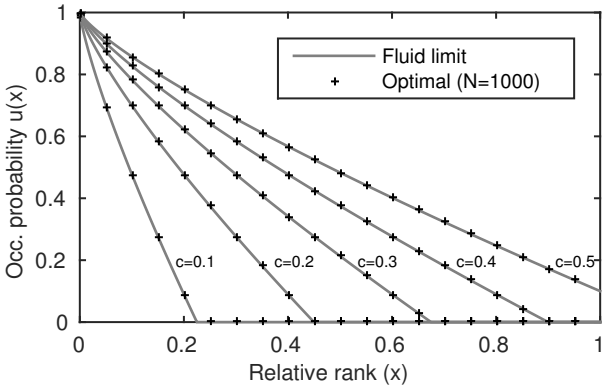

Figure 1: Asymptotic Optimal Policy

5.1.2 Asymptotic optimal cost

We are interested here in finding a limiting expression for large N for the optimal hit probability.

Here a new distinction appears in terms of popularity distribution. The most interesting case is when $\beta < 1$, i.e. the popularities are more heavy-tailed. In that case we will see that the optimal hit probability remains strictly below unity, and can be characterized asymptotically by integrating the optimal law described above. The proof of the following result can be found in [13].

THEOREM 4. *Suppose $\beta < 1$, and $C_N = cN$ for $c < 1$. Then as $N \to \infty$, the optimal hit probability converges to*

$$H^* = (1-\beta)\int_0^1 x^{-\beta}\left[1 - (1-u^*(x))^{\frac{\alpha}{\alpha-1}}\right] dx, \tag{28}$$

where $u^*(x)$ is the optimal policy from Theorem 3. In particular:

$$H^* = \begin{cases} \left(\frac{c}{c_{\alpha,\beta}}\right)^{1-\beta} \frac{\alpha\beta}{\alpha\beta+1-\beta}, & c < c_{\alpha,\beta} \\ 1-(1-\beta)(1-c)^{\frac{\alpha}{\alpha-1}}[1+\beta(\alpha-1)]^{\frac{1}{\alpha-1}}, & c \geqslant c_{\alpha,\beta} \end{cases} \tag{29}$$

Suppose we let $\beta \uparrow 1$ in the previous expressions. It is easily verified that we obtain $H^* \to 1$, indicating that the optimal policy becomes efficient when the popularity distribution has lighter tails. Indeed, we show below that this situation holds for $\beta \geqslant 1$, while maintaining the heavy tailed arrivals. Note that in this case the limiting H^* is no longer characterized by an integral of the continuum optimal policy, in particular the integral in (28) would become divergent.

PROPOSITION 2. *Consider a cache system as before with capacity $C_N = cN$, and $\lambda_n = \frac{1}{S_n(\beta)}\frac{1}{n^\beta}$ with $\beta \geqslant 1$. Then the optimal hit rate verifies $H^{*,N} \longrightarrow_N 1$ and moreover the static policy that stores only the C_N most popular contents is asymptotically optimal.*

PROOF. Since the static policy is feasible for Problem 2, we will have:

$$1 \geqslant H^{*,N} \geqslant \sum_{n=1}^{C_N} \lambda_n = \sum_{n=1}^{C_N} \frac{1}{S_N(\beta)}\frac{1}{n^\beta} = \frac{S_{C_N}(\beta)}{S_N(\beta)}.$$

If $\beta > 1$, then $S_N(\beta)$ has a finite limit $\zeta(\beta)$ as $N \to \infty$. Since C_N also goes to infinity we have:

$$\frac{S_{C_N}(\beta)}{S_N(\beta)} \underset{N}{\longrightarrow} 1,$$

which proves this case.

When $\beta = 1$, $S_N(1)$ diverges but we have the well known equivalent $S_N(1) \sim \log(N)$ which upon substitution yields:

$$\lim_N \frac{S_{C_N}(1)}{S_N(1)} = \lim_N \frac{\log(c)+\log(N)}{\log(N)} \to_N 1.$$

□

REMARK 2. *Note that in the case $\beta > 1$, since the tail of the popularities decays fast enough, the same result holds provided $C_N \to \infty$, regardless of the speed of convergence. Therefore, as long as sufficient capacity is allocated in the cache, the hit rate can be made arbitrarily high.*

5.1.3 Some limiting cases

By considering some extreme cases, we can further compare the optimal policy with the static (LFU) case.

REMARK 3. *Suppose the tail parameter of the interarrival times $\alpha \to \infty$. In this case, it is easy to see that $c_{\alpha,\beta} \to 1$, $x_0 \to c$ and the optimal policy converges to $u(x) = \mathbf{1}_{\{x \leqslant x_0\}}$, which amounts to the static most popular policy. In fact, in this case, it is easy to show that the inter-request times converge, as $n \to \infty$ to exponential random variables with parameter λ_n, thus recovering the result of Poisson arrivals.*

REMARK 4. *Suppose that $\beta \to 0$, i.e. the popularities become more and more equal. In this case $c_{\alpha,\beta} \to 0$ and the optimal timer policy from (29) satisfies:*

$$H^* \underset{\beta\to 0}{\longrightarrow} 1-(1-c)^{\frac{\alpha}{\alpha-1}}.$$

Note that $1-(1-c)^{\frac{\alpha}{\alpha-1}} > c$, which will be the hit rate of any static policy that stores a fraction c of the contents. Here the gain of TTL based policies is evident: under bursty arrivals, the timer policy is able to take advantage of the successive requests, enlarging the hit rate. The effect is greater as $\alpha \searrow 1$ (the heavier the tails of the interarrival times).

5.2 Capacity usage of the optimal policy

When using TTL policies, the memory constraint (5) imposes a limit of cache occupation only on average. One concern is that memory usage may overshoot the capacity constraint C by large amounts. We now show how to estimate memory usage and prove that in the asymptotic limit these overshoots are negligible.

Consider the total occupation process, given by $Z^N(t) := \sum_{n=1}^N Z_n(t)$. Under the optimal TTL policy, $E[Z(0)] = \sum_{n=1}^N u_n^{*,N} = C$ by construction. Moreover, since the occupation processes of every file are independent, we have:

$$\mathrm{Var}(Z^N(0)) = \sum_{n=1}^N \mathrm{Var}(Z_n(0)) = \sum_{n=1}^N u_n^{*,N}(1-u_n^{*,N}) \leqslant \frac{N}{4},$$

since each term comes from a Bernoulli random variable. By a direct application of Chebysev's inequality we have the following law of large numbers.

PROPOSITION 3. *Let the buffer size scale as $C_N = cN$ and let $N \to \infty$, then under the optimal TTL policy the occupation process verifies, for every $\delta > 0$:*

$$P\left(Z^N(t) > (1+\delta)C\right) \underset{N}{\to} 0.$$

Thus, the relative deviations from the average occupation are negligible in the limit.

By approximating the value of $\sum_{n=1}^N u_n^{*,N}(1-u_n^{*,N})$ by a Riemann integral as in the proof of Theorem 4, we can compute the asymptotic variability of the occupation process.

PROPOSITION 4. *Under the same scaling, the variance of the occupation process under the optimal TTL policy satisfies:*

$$\lim_N \frac{1}{N}\mathrm{Var}(Z^N(0)) = \int_0^1 u(x)(1-u(x))\,dx$$

where $u(x)$ is given by (26) for $c < c_{\alpha,\beta}$ or (27) for $c > c_{\alpha,\beta}$.

The above proposition shows that in fact the variance scales linearly with N. The constant can be readily calculated. For instance, a direct computation of the integral for the case $c < c_{\alpha,\beta}$ yields:

$$\frac{1}{N}\mathrm{Var}(Z^N(0)) \underset{N}{\longrightarrow} \frac{c}{2\beta(\alpha-1)+1}.$$

Analogously, one can solve for $c > c_{\alpha,\beta}$.

6. NUMERICAL EXPERIMENTS

In this Section, we will analyze the performance of the optimal TTL policy in several scenarios, and compare with other policies such as the static policy that stores the most popular files, and the LRU replacement algorithm. We also provide a new approximation for finite N of the optimal hit rate, and provide a variation of the TTL policy that ensures constant memory usage, yet maintains high performance.

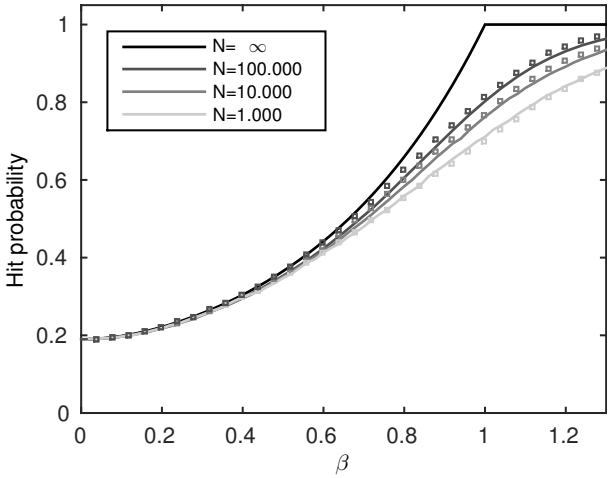

Figure 2: Hit probability as a function of β, asymptotic limit and simulations. $\alpha = 2$, $c = 0.1$.

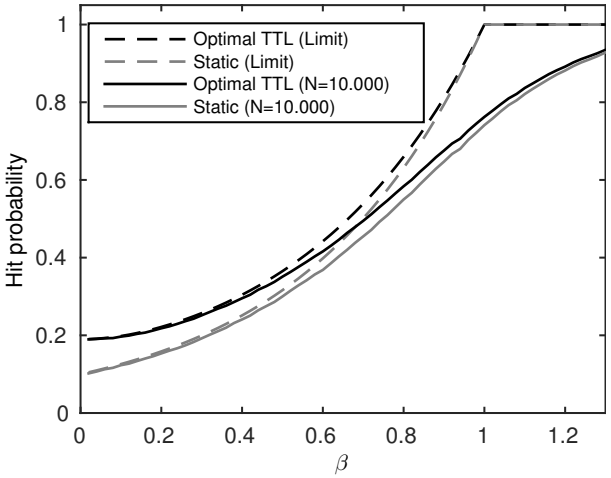

Figure 3: Optimal TTL policy and static policy performance as a function of β. $\alpha = 2$, $\beta = 0.1$.

To evaluate the algorithms, we implemented a discrete-event simulator on Java that emulates cache behavior under different policies (TTL, LRU, Static). The system is fed by general renewal arrival processes of intensity λ_n and total arrival rate $\lambda = 1$. Popularities are chosen as the Zipf(β) distribution as described before. Recall that a typical measured value for the popularity parameter is $\beta = 0.8$ [9].

To implement the optimal TTL policy described in Section 5 in the heavy-tailed case, we first solved the optimization Problem 2 in Matlab using the convex optimization library CVX [20]. Solving for the exact policy in large instances of this problem is computer intensive despite the convex structure. However, numerical trials show that, as long as the catalog size $N > 500$, the optimal occupation probabilities $u_n^{*,N}$ can be well approximated by the fluid limit function $u(x)$ from Theorem 3. Therefore, in our simulations, we simply derive the optimal occupation probabilities from the relative rank of the file and the fluid limit value, and set the timers accordingly. Note that this only requires knowledge of the tail parameters α, β of the distributions involved.

6.1 Convergence of the hit probability

We begin by analyzing the convergence to the limit H^* as the catalog size N grows. We simulated the system for different values of N with varying popularity parameter $\beta \in (0, 1.3)$. Inter-request times follow a Pareto distribution with $\alpha = 2$ and the storage capacity is fixed at $c = 0.1$ (i.e. 10% of the files stored on average). In Figure 2 we plot the simulation results as well as the limit given by equation (29) for $\beta < 1$. For $\beta \geqslant 1$ Proposition 2 shows that $H^* = 1$.

As N grows, the hit probability approaches the computed limit, although it does so slowly for critical values of β. This is not a limitation of approximating the optimal policy by the fluid limit $u(x)$, which indeed shows fast convergence; rather, the issue is the slow convergence of the Riemann sum to the integral in the proof of Theorem 4. Seeking a better estimate for finite N, we computed the difference between rectangular and trapezoidal approximations to the integral in question, and used a more precise approximation of $S_N(\beta)$. This yields the following correction for the hit rate with finite N:

$$H_N^* \approx \frac{(2N)^{1-\beta}}{(2N)^{(1-\beta)} - 1} H^* - \frac{(1+\beta).(2^{-\beta})}{(2N)^{(1-\beta)} - 1}. \qquad (30)$$

These approximations are marked in Fig. 2 as boxes, showing that indeed the approximation captures the hit probability for finite N and can be used to predict system performance.

6.2 Comparison with the static policy

In the case of decreasing hazard-rates, it was shown that the optimal TTL policy outperforms the optimal static policy, which is also the limit of the least-frequently-used (LFU) policy in steady state. We now quantify this for the case analyzed in Section 5. Note that, under the same scaling $C_N = cN$, the asymptotic hit probability of the static policy can be computed for $\beta < 1$ as:

$$\lim_N \frac{1}{S_N(\beta)} \sum_{n=1}^{C_N} \frac{1}{n^\beta} = \lim_N \frac{S_{C_N}(\beta)}{S_N(\beta)} = \lim_N \frac{(cN)^{1-\beta}}{N^{1-\beta}} = c^{1-\beta}.$$

where we have used the equivalent in (15). For $\beta \geqslant 1$, Proposition 2 shows that the static policy also achieves maximal hit probability asymptotically.

In Figure 3 we plot the the asymptotic hit rates (dashed lines) as well as simulation results for the case of $\alpha = 2$, varying β. Here and in the remainder of the section we will use for simulations a catalog size $N = 10000$ and a cache size $C = 1000$, ($c = 0.1$). We remark that variations in c were explored as well, but they do not affect the qualitative conclusions which we draw in each case.

The main observation is that, when the arriving requests satisfy the decreasing hazard rate property, the static policy (and hence LFU) is no longer optimal. Indeed, because of its accommodation for traffic burstiness, the TTL policy performs better than the static policy across all ranges of β, with a stronger advantage in the case of more uniform popularities (lower values of β).

6.3 The effect of the hazard-rate on LRU

We now analyze the performance of the LRU policy, which is a popular policy within cache implementations due to its

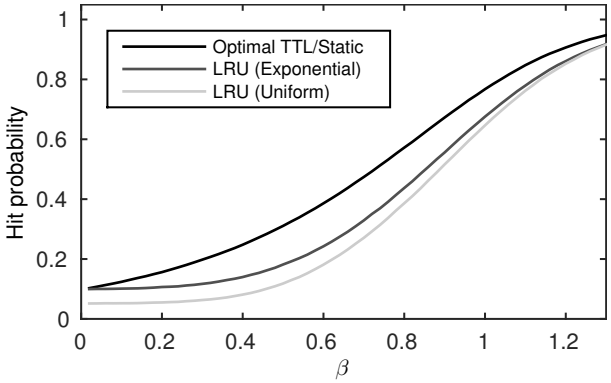

Figure 4: Optimal TTL policy vs. LRU in the Poisson and increasing hazard rate case. $c = 0.1$, $N = 10000$.

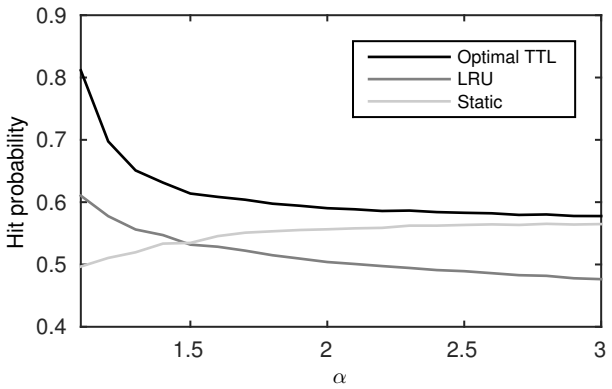

Figure 6: Hit rate of the optimal TTL, LRU and static policies for Pareto inter-request times with varying α. $\beta = 0.8$, $c = 0.1$, $N = 10000$.

simplicity, since it does not require previous knowledge of the request distribution and popularities. It turns out that our analysis based on hazard rates sheds some light on *when* LRU can provide near-optimal performance.

The main point is that LRU can be well approximated by an *homogeneous timer* policy: this is the key idea from Che et al. [10] and recently extended in [29] for general inter-request distributions. The eviction time T_C in LRU is approximately constant and homogeneous across files in a large system, and can be approximated by the solution of eq. (6) as discussed in Section 3. Therefore, in the situations when the optimal TTL policy assigns homogeneous timers, LRU performance should be near-optimal.

As a first example, we consider the constant (Poisson) together with the increasing hazard rate (IHR) case; in both the optimal TTL policy coincides with the static policy. For IHR we used inter-request times following a Uniform distribution in $[0, 2/\lambda_n]$.

Results are shown in Figure 4. In this case LRU is clearly suboptimal, and its deficiency becomes more severe in the IHR case. Note that in both cases, the optimal TTL solution is highly inhomogeneous due to the convexity of the objective function: some files are statically stored and the other ones never are (an extremal point). As in this case requests

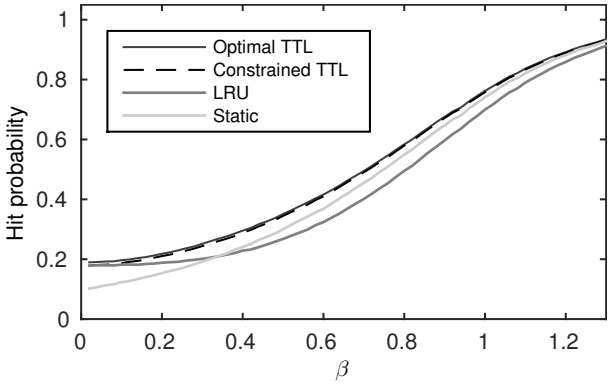

Figure 5: Optimal TTL policy, LRU and static policy for Pareto inter-request times with $\alpha = 2$, $c = 0.1$, $N = 10000$.

are more periodical in time, LRU is unable to quickly evict the less popular files once they get stored.

The situation is more interesting when inter-request times have the decreasing hazard rate property. In Figure 5 we plot the simulation results for Pareto ($\alpha = 2$) inter-arrival times, comparing the hit rates of LRU, the static policy, and the optimal TTL.

Note that in this case LRU may outperform the static policy. In fact, for lower values of β, LRU achieves near-optimal performance as compared to the TTL policy. An explanation for this lies in the fact that, in the DHR case, the optimal solution tends to equalize the hazard rates of the stored files: as $\beta \searrow 0$, popularities become similar and hazard rates become homogeneous. Therefore, the optimal timer policy has more homogeneous timers, and the LRU policy with its near fixed timer provides near optimal performance. This increase in the performance of LRU over the static policy was already observed in [29] for hyper-exponential distributions (which satisfy the DHR property) in some situations. Note also that in the limit case of Poisson arrivals, the LRU policy also approaches the optimal as $\beta \searrow 0$ in Figure 4.

One could wonder if the TTL's gains over LRU are based on the fact that capacity is imposed only on average, allowing the buffer occupation to go temporarily beyond C. To test this, we implemented also a variation of the optimal policy which we called *constrained TTL*. This is a similar to the Practical TTL policy described in [16] to maintain constant buffer occupation[2]: upon arrival of a request (with the buffer full), the object with the least time remaining in its timer is evicted immediately. If a timer expires, the object is not evicted until a request not present in the buffer arrives, and in that case the object with longest overdue timer is evicted. Thus, the constrained TTL policy maintains constant buffer occupation, and as shown in Figure 5 it achieves close-to-optimal performance.

As a final example, we analyze the performance of LRU against the optimal TTL and static policies with varying tail parameter α and fixed $\beta = 0.8$. As we can see in Fig. 6, as the burstiness of traffic increases ($\alpha \searrow 1$), LRU also outperforms the static policy, by storing temporarily less popular contents but which may be requested again soon,

[2]The main difference with [16] is that here timers are computed following the optimal TTL policy from Theorem 3.

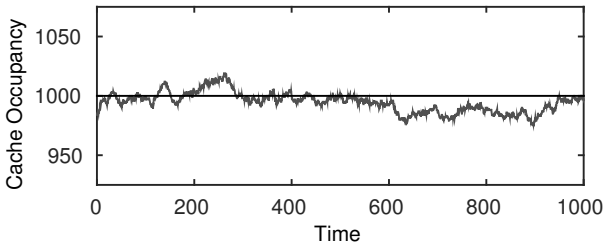

Figure 7: Capacity usage as a function of time for $N = 10000$ and $C = 1000$. Pareto arrivals with $\alpha = 2$ and $\beta = 0.8$.

due to the DHR property. As α grows, we approach the Poisson case as discussed in Section 5.1, and the static policy becomes optimal.

6.4 Capacity usage

Finally, we analyze the steady state memory usage of the optimal TTL policy. In Figure 7, we plot the number of stored objects under the optimal TTL policy as a function of time, for Pareto arrivals with $\alpha = 2$ and popularity parameter $\beta = 0.8$. The catalog and buffer size are the same as before. As expected from Proposition 3, the capacity usage has low overshoot from the average occupation. This also justifies why the Constrained TTL policy performs almost as well as the optimal. Using Proposition 4, one can estimate the occupation variance in this case as $Var(C(t)) \approx 40$, which is in accordance with the simulation.

7. GENERALIZATIONS

In this section we explore two generalizations of the optimal timer based policy characterized by Problem 1. We first analyze the case of exponentially distributed timers, first studied in detail in [16]. We show how to cast the optimal timer policy by writing an optimization problem by means of Laplace transforms. Then we proceed to analyze lines of cache networks, using both exponential and deterministic timers, and show also how to derive a timer based policy with desirable optimality properties.

7.1 Exponential timers

In Section 3 we assumed that timers associated to each file were deterministic. Random timers with general distribution are discussed in [16], in particular the case when $T_k^{(n)} \sim \exp(\mu_n)$. Under this policy, upon arrival of request k for content n, a random timer T_k^n is chosen independently for each arrival, and the file is evicted upon timer expiration as before. The authors of [16] derive the hit probability and occupation probability, which we reproduce here using our notation.

The hit probability for file n in steady state is simply:

$$P_0^{(n)}\left(X_0^{(n)} < T_0^{(n)}\right) = \int_0^\infty e^{-\mu_n x} dF_n(x) = F_n^*(\mu_n),$$

where $F_n^*(s)$ is the Laplace transform of the inter-request time X_0, i.e. $E_0^{(n)}\left[e^{-sX_0}\right]$.

By the same reasoning of Section 3, the occupation probability u_n can be computed as:

$$\begin{aligned} u_n = E[Z_n(0)] &= \lambda_n E_0^{(n)}\left[\min\{X_0^{(n)}, T_0^{(n)}\}\right] \\ &= \lambda_n \int_0^\infty e^{-\mu_n x}(1 - F_n(x))dx \\ &= \int_0^\infty e^{-\mu_n x} d\hat{F}_n(x) =: \hat{F}_n^*(\mu_n), \end{aligned} \quad (31)$$

where in the second equality we used the distribution function of the minimum of two independent random variables, and $\hat{F}_n^*(s)$ is the Laplace transform of the residual lifetime distribution $\hat{F}_n$.

Using these results, the optimal timer policy amounts to choosing the eviction frequencies $\{\mu_n\}$ according to:

PROBLEM 3 (OPTIMAL EXPONENTIAL TTL POLICY).

$$\max_{\mu_n \geqslant 0} \sum_{n=1}^N \lambda_n F_n^*(\mu_n)$$
$$s.t.: \sum_n \hat{F}_n^*(\mu_n) \leqslant C.$$

Problem 3 is different from Problem 1 with deterministic timers, but has a similar structure: instead of the inter-request distributions and their associated residual lifetimes, the corresponding Laplace transforms appear, evaluated at the eviction frequencies instead of the eviction times.

The Laplace transforms F_n^* and $\hat{F}_n^*$ are convex and decreasing functions over the positive real line. This means Problem 3 is seeking to *maximize* a convex function over a convex domain, not a convex optimization program. As argued before, the result should be at the extreme points of the convex set, but in this case this set can be the whole boundary $\sum_n \hat{F}_n^*(\mu_n) = C$.

As in Section 3, one can rewrite this problem in terms of the occupation probabilities, by performing the change of variables $u_n = (\hat{F}_n^*)^{-1}(\mu_n)$, obtaining the analogue of Problem 2:

PROBLEM 4.

$$\max_{u_n \in [0,1]} \sum_{n=1}^N \lambda_n F_n^*((\hat{F}_n^*)^{-1}(u_n)),$$
$$s.t.: \sum_n u_n \leqslant C.$$

In the case of Poisson arrivals, $F_n = \hat{F}_n$ and thus problem reduces to the linear program of Section 4.1, so the solution again degenerates to the static policy, as expected due to the PASTA property. Obtaining more general convexity conditions for general inter-request distributions under the same change of variables proves harder in this case, and will be pursued in future work.

7.2 Infinite line of TTL caches

We now extend our analysis of optimal TTL caching beyond the case of a single cache. Consider instead a line of TTL caches, with requests arriving at one end, and the following management policy: only one copy of each content is stored at some cache k in the line, with $k = 0$ representing the cache that receives external requests. When content n is requested, the object is moved to cache 0 and a timer is started. Upon timer expiration, it is evicted and moved to the next cache in line, where the process is repeated.

This "move to front" rule tends to keep popular objects in the first caches of the line, while less requested content drifts away and is stored further away from the users. Note again that one copy of the file is kept at any point in time.

We first analyze this policy in the case of Poisson arrivals with intensity λ_n and exponentially distributed timers of frequency μ_n. Since each file behaves independently, we begin by modeling the position of each file in the line. For simplicity, we will assume that the line is infinite, but the same analysis can be carried out for finite lines.

Let $L_n(t)$ denote the position of content n in the line over time. It is easy to see that, under the memoryless assumptions above, $L_n(t)$ is a continuous-time Markov chain with state space $k \in \{0, 1, 2, \ldots\}$ and transition rates:

$$q_{k,k+1} = \mu_n, \quad q_{k,0} = \lambda_n.$$

This Markov chain is always stable, and by solving its global balance equations it can be shown that its invariant distribution is geometric:

$$\pi_n(k) = P(L_n = k) = \left(\frac{\lambda_n}{\lambda_n + \mu_n}\right)\left(\frac{\mu_n}{\lambda_n + \mu_n}\right)^k.$$

Therefore, the expected search cost of file n, measured by the number of hops needed to retrieve the content, is given in steady state by:

$$E[L_n] = \frac{\frac{\mu_n}{\lambda_n+\mu_n}}{1 - \frac{\mu_n}{\lambda_n+\mu_n}} = \frac{\mu_n}{\lambda_n}. \tag{32}$$

Assume now that caches have an average capacity C. The average occupation of cache k in the line is given by $\sum_n \pi_n(k)$. Since π_n is decreasing in k for each n, the most restrictive cache is the first in the line. Therefore, we can formulate an optimization problem to determine the eviction frequencies that minimize the average search cost:

Problem 5.

$$\min_{\{\mu_n \geqslant 0\}} \sum_{n=1}^{N} \lambda_n E[L_n] = \sum_{n=1}^{N} \mu_n,$$
$$s.t.: \sum_{n=1}^{N} \pi_n(0) = \sum_{n=1}^{N} \frac{\lambda_n}{\lambda_n + \mu_n} \leqslant C.$$

The above is a convex optimization program that can be readily solved to yield the optimal eviction frequencies.

We now extend the above result to the case of deterministic timers. The key property is to show that the steady state distribution of $L_n(t)$ is again geometric, by using a method of stages. Namely, to model a deterministic timer distribution with value T_n, we divide the level k in S consecutive stages $k_1, \ldots, k_S$, each with an exponential timer of parameter $S\mu_n$ and $\mu_n = T_n^{-1}$. With this change, the time in each stage follows an Erlang distribution of parameters S and $S\mu_n$. As the number of stages S grows large, this approximates the deterministic distribution concentrated at T_n, due to the law of large numbers.

This procedure yields a modified Markov chain $\tilde{L}_n(t)$, the state space is now $l \in \{0, 1, 2, \ldots\}$ and the set $\{kS \leqslant l < (k+1)S\}$ corresponds to the event $\{L_n = k\}$, i.e. the object is cached in level k of the line. The transition rates for $\tilde{L}_n$ are given by:

$$\tilde{q}_{l,l+1} = S\mu_n, \quad \tilde{q}_{l,0} = \lambda_n,$$

and therefore by the same reasoning as before:

$$\tilde{\pi}_n(l) = P(\tilde{L}_n = l) = \left(\frac{\lambda_n}{\lambda_n + S\mu_n}\right)\left(\frac{S\mu_n}{\lambda_n + S\mu_n}\right)^l.$$

Accumulating this distribution over $\{kS \leqslant l < (k+1)S\}$, the stages of level k, yields again a geometric distribution:

$$\pi_n(k) = P(L_n(t) = k) = P\left(\tilde{L}_n \in \{kS, \ldots, (k+1)S - 1\}\right)$$
$$= \left(\frac{\lambda_n}{\lambda_n + S\mu_n}\right)^S \left[\left(\frac{S\mu_n}{\lambda_n + S\mu_n}\right)^S\right]^k.$$

As the number of stages $S \to \infty$, we have the limit:

$$\left(\frac{S\mu_n}{\lambda_n + S\mu_n}\right)^S \underset{S\to\infty}{\longrightarrow} e^{-\frac{\lambda_n}{\mu_n}} = e^{-\lambda_n T_n}.$$

We have thus proved:

Proposition 5. *Let $L_n(t)$ denote the process tracking the level at which a file is stored in a line of TTL caches, where requests arrive as a Poisson process of intensity λ_n, and the file is evicted to the next cache in line after a deterministic time T_n. Then, L_n in steady state verifies:*

$$P(L_n(t) = k) = \left(1 - e^{-\lambda_n T_n}\right)\left(e^{-\lambda_n T_n}\right)^k, \tag{33}$$

and thus the average search cost is given by:

$$E[L_n] = \frac{e^{-\lambda_n T_n}}{1 - e^{-\lambda_n T_n}} = \frac{1}{e^{\lambda_n T_n} - 1}.$$

We can use this result to formulate the optimal TTL problem equivalent to Problem 5. However, while the search cost $E[L_n]$ for file n is convex in T_n, the occupation probability at cache $k = 0$ is given by $u_n = 1 - e^{-\lambda_n T_n}$, a concave function. Nevertheless, we can use the idea of Section 4, changing to variables u_n, posing the optimal TTL problem as:

Problem 6.

$$\min_{\{u_n \in [0,1]\}} \sum_{n=1}^{N} \lambda_n E[L_n] = \sum_{n=1}^{N} \lambda_n \frac{1 - u_n}{u_n},$$
$$s.t.: \sum_{n=1}^{N} \pi_n(0) = \sum_{n=1}^{N} u_n \leqslant C.$$

By solving this convex optimization, the optimal timers are obtained from $T_n = -\frac{1}{\lambda_n}\log(1 - u_n^*)$.

Remark 5. *Note that we could also have recast Problem 5 in terms of the occupation probabilities, and obtained the same convex program as Problem 6. This implies that the optimal performance is the same for exponential or deterministic timers. The reason behind it is that both yield a geometric distribution for the stage probabilities, so in essence we are optimizing over the parameter of this distribution, indirectly through the choice of timers or eviction frequencies.*

8. CONCLUSIONS

In this paper, we analyzed from a theoretical perspective the performance of caching systems operating under timer based eviction policies. Building upon previous results for the relevant performance metrics, we formulated an optimization problem that characterizes the optimal choice of timers that maximizes the hit probability. This enables us

to understand the structural properties of the system. In particular, we proved that the convexity of this problem is tightly related to the monotonicity of the hazard rate of the inter-request times, and provided a detailed analysis of the case of heavy tailed inter-request distributions and popularities, which are relevant in practice. Through simulation and numerical evaluation, we also analyzed the performance of alternative cache management algorithms. We also pursued some generalizations to random timers and networks of caches which will be explored in future work.

9. ACKNOWLEDGMENTS

The authors would like to thank D. Sadoc-Menasche and R. Srikant for their inputs during useful discussions. The authors were partially supported by AFOSR US under grant FA_9550_15_1_0183.

10. REFERENCES

[1] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[2] F. Baccelli and P. Brémaud. *Elements of Queueing Theory*. Springer-Verlag, Berlin, 2013.

[3] O. Bahat and A. M. Makowski. Measuring consistency in TTL-based caches. *Performance Evaluation*, 62(1):439–455, 2005.

[4] J. Barrera and J. Fontbona. The limiting move-to-front search-cost in law of large numbers asymptotic regimes. *Annals of Applied Probability*, 20(2):722–752, 2010.

[5] D. S. Berger, P. Gland, S. Singla, and F. Ciucu. Exact analysis of TTL cache networks. *Performance Evaluation*, 79:2–23, 2014.

[6] D. S. Berger, S. Henningsen, F. Ciucu, and J. B. Schmitt. Maximizing cache hit ratios by variance reduction. *ACM SIGMETRICS Performance Evaluation Review*, 43(2):57–59, 2015.

[7] G. Bianchi, A. Detti, A. Caponi, and N. Blefari Melazzi. Check before storing: What is the performance price of content integrity verification in LRU caching? *ACM SIGCOMM Computer Communication Review*, 43(3):59–67, 2013.

[8] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: Evidence and implications. In *Proc. of IEEE/Infocom 1999*, pages 126–134.

[10] H. Che, Y. Tung, and Z. Wang. Hierarchical web caching systems: Modeling, design and experimental results. *IEEE Journal on Selected Areas in Communications*, 20(7):1305–1314, 2002.

[11] A. Dan and D. Towsley. An approximate analysis of the LRU and FIFO buffer replacement schemes. In *Proc. of ACM/SIGMETRICS 1990*, pages 143–152.

[12] M. Dehghan, L. Massoulie, D. Towsley, D. Menasche, and Y. C. Tay. A utility optimization approach to network cache design. In *Proc. of IEEE/Infocom 2016*, page to appear.

[13] A. Ferragut, I. Rodríguez, and F. Paganini. Optimizing TTL caches under heavy-tailed demands (extended version). Available at http://fi.ort.edu.uy/2243/17/publications.html.

[14] J. A. Fill. Limits and rates of convergence for the distribution of search cost under the move-to-front rule. *Theoretical Computer Science*, 164(1):185–206, 1996.

[15] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley. Analysis of TTL-based cache networks. In *Proc. of Intl. Conf on Performance Evaluation Methodologies and Tools (VALUETOOLS)*, pages 1–10, 2012.

[16] N. C. Fofack, P. Nain, G. Neglia, and D. Towsley. Performance evaluation of hierarchical TTL-based cache networks. *Computer Networks*, 65:212–231, 2014.

[17] C. Fricker, P. Robert, and J. Roberts. A versatile and accurate approximation for LRU cache performance. In *Proc. of the 24th International Teletraffic Congress*, pages 57–64, 2012.

[18] N. Gast and B. V. Houdt. Transient and steady-state regime of a family of list-based cache replacement algorithms. In *Proc. of ACM/SIGMETRICS 2015*, pages 123–136.

[19] E. Gelenbe. A unified approach to the evaluation of a class of replacement algorithms. *IEEE Trans. on Computers*, 100(6):611–618, 1973.

[20] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, Mar. 2014.

[21] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proc. of the ACM/Conext*, pages 1–12, 2009.

[22] P. Jelenković and A. Radovanović. Asymptotic insensitivity of least-recently-used caching to statistical dependency. In *Proc. of IEEE/Infocom 2003*, pages 438–447.

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

[24] P. R. Jelenković and A. Radovanović. Least-recently-used caching with dependent requests. *Theoretical computer science*, 326(1):293–327, 2004.

[25] P. R. Jelenković and A. Radovanović. The persistent-access-caching algorithm. *Random Structures & Algorithms*, 33(2):219–251, 2008.

[26] P. R. Jelenković, A. Radovanović, and M. S. Squillante. Critical sizing of LRU caches with dependent requests. *Journal of Applied Probability*, 43(4):1013–1027, 2006.

[27] J. Jung, A. W. Berger, and H. Balakrishnan. Modeling TTL-based internet caches. In *Proc. of IEEE/Infocom 2003*, pages 417–426.

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

[29] V. Martina, M. Garetto, and E. Leonardi. A unified approach to the performance analysis of caching systems. In *Proc. of IEEE/Infocom 2014*, pages 2040–2048.

[30] E. J. Rosensweig, J. Kurose, and D. Towsley. Approximate models for general cache networks. In *Proc. of IEEE/Infocom 2010*, pages 1–9.