THE CACHE INFERENCE PROBLEM
and its Application to Content and Request Routing

NIKOLAOS LAOUTARIS[†‡] GEORGIOS ZERVAS[†] AZER BESTAVROS[†] GEORGE KOLLIOS[†]
nlaout@cs.bu.edu zg@cs.bu.edu best@cs.bu.edu gkollios@cs.bu.edu

***Abstract*— In many networked applications, independent caching agents cooperate by servicing each other's miss streams, without revealing the operational details of the caching mechanisms they employ. Inference of such details could be instrumental for many other processes. For example, it could be used for optimized forwarding (or routing) of one's own miss stream (or content) to available proxy caches, or for making cache-aware resource management decisions. In this paper, we introduce the *Cache Inference Problem* (CIP) as that of inferring the characteristics of a caching agent, given the miss stream of that agent. While CIP is insolvable in its most general form, there are special cases of practical importance in which it is, including when the request stream follows an Independent Reference Model (IRM) with generalized power-law (GPL) demand distribution. To that end, we design two basic "litmus" tests that are able to detect LFU and LRU replacement policies, the effective size of the cache and of the object universe, and the skewness of the GPL demand for objects. Using extensive experiments under synthetic as well as real traces, we show that our methods infer such characteristics accurately and quite efficiently, and that they remain robust even when the IRM/GPL assumptions do not hold, and even when the underlying replacement policies are not "pure" LFU or LRU. We exemplify the value of our inference framework by considering example applications.**

I. INTRODUCTION

Motivation: Caching is a fundamental building block of computing and networking systems and subsystems. In a given system, *multiple* caching agents may co-exist to provide a single service or functionality. A canonical example is the use of caching in multiple levels of a memory system to implement the abstraction of a "memory hierarchy".

For distributed systems, the use of caching is paramount: route caches are used to store recently-used IP routing entries and is consulted before routing tables are accessed; route caches are also used in peer-to-peer overlays to cache hub and hub cluster information as well as Globally Unique IDentifiers (GUIDs) to IP translations for content and query routing purposes [1]; DNS lookups are cached by DNS resolvers to reduce the load on upper-level DNS servers [2]; distributed host caching systems are used in support of P2P networks to allow peers to find and connect to one another [3]; web proxy and reverse proxy caches are used for efficient content distribution and delivery [4], [5], and the list goes on.

When multiple caching agents are used to support a single operation, be it a single end-host or a large CDN overlay network, the "design" of such agents is carefully coordinated to ensure efficient operation. For instance, the *size* and *replacement strategies* of the caching mechanisms used to implement traditional memory hierarchies are matched up carefully to maximize overall performance under a given *workload* (typically characterized by reference models or benchmarks). Similarily, the placement, sizing, and cache replacement schemes used for various proxies in a CDN are carefully coordinated to optimize the content delivery process subject to typically-skewed demand profile from various client populations.

Increasingly, however, a system may be forced to rely on one or more caching agents that are not necessarily part of its autonomous domain and as such may benefit from infering the characteristics of such agents. Next, we give a concrete example to motivate the work presented in this paper.

Application to Informed Caching Proxy Selection: Consider a web server (or a proxy thereof), which must decide which one of potentially many candidate reverse caching proxies (possibly belonging to competing ISPs) it should use to "front end" its popular content. Clearly, in such a setting, it cannot be assumed that details of the design or policies of such caching agents would be known to (let alone controlled or tuned by) the entity that wishes to use such agents. Yet, it is clear that such knowledge could be quite valuable. For example, given a choice, a web server would be better off selecting a reverse proxy cache with a larger cache size. Alternatively, if the web server needs to partition its content across multiple reverse proxies, it would be better off delegating content that exhibits popularity over long time scales to proxies employing an LFU-like replacement policy, while delegating content that exhibits popularity over shorter time scales to proxies employing an LRU-like replacement policy. Even if the web server has no choice in terms of the reverse caching proxy to use, it may be useful for the web server to ascertain the characteristics of the reference stream served by the caching proxy.

The above are examples of how knowledge of the characteristics of remote caching agents could empower a server to make judicious decisions. Later, in Section VIII, we show how the same information could also empower a client-side caching proxy to make judicious decisions regarding which other caching proxies to use to service its own miss stream. Both of these are examples of *Informed Caching Proxy Selection*, which has applications in any system that allows autonomous entities the flexibility of selecting from among (or routing content or requests across) multiple remote caching agents.

The Cache Inference Problem: In many of the distributed applications and networking systems (to which we have alluded above), a caching agent that receives a request which it cannot service from its local storage redirects (or routes) such requests

[†] Computer Science Dept, Boston University, Boston, Massachusetts, USA.
[‡] Dept of Informatics and Telecommunications, University of Athens, Greece.
[§] N. Laoutaris is supported by a Marie Curie Outgoing International Fellowship of the EU MOIF-CT-2005-007230. A. Bestavros and G. Kollios are supported in part by NSF CNS Cybertrust Award #0524477, CNS NeTS Award #0520166, CNS ITR Award #0205294, and EIA RI Award 0202067.

to other subsystems – *e.g.*, another caching agent, or an origin server. Such redirected requests constitute the "miss stream" of the caching agent. Clearly, such a miss stream carries in it information about the caching mechanisms employed by the caching agent. While the subsystems at the receiving end of such miss streams may not be privy to the "design" of the caching agent producing such a miss stream, it is natural to ask whether *"Is it possible to use the information contained in a miss stream to infer some of the characteristics of the remote cache that produced that stream?"*

Accordingly, the most general Cache Inference Problem (CIP) could be stated as follows: *"Given the miss stream of a caching agent, infer the replacement policy, the capacity, and the characteristics of the input request stream"*. The above question is too unconstrained to be solvable, as it allows arbitrary reference streams as inputs to the cache. Thus, we argue that a more realistic problem statement would be one in which assumptions about common characteristics of the cache reference stream (*i.e.*, the access pattern) are made. We single out two such assumptions, both of which are empirically justified, commonly made in the literature, and indeed leveraged in many existing systems. The first is that the reference stream follows the Independent Reference Model (IRM) [6], whereas the second is that the demand for objects follows a Generalized Power Law (GPL). As we show in Section II, CIP when subjected to the IRM and GPL assumptions is solvable.

As we hinted above, we underscore that the GPL and IRM assumptions on the request stream have substantial empirical justification, making the solution of the CIP problem of significant practical value to many systems. Moreover, as will be evident later in this paper, the solutions we devise for the CIP problem subject to GPL and IRM work well *even when the request stream does not satisfy the GPL and IRM assumptions*.

Beyond Inference of Caching Characteristics: So far, we have framed the scope of our work as that of "infering" the characteristics of an underlying caching agent. While that is the main technical challenge we consider in this paper (with many direct applications), we note that the techniques we develop have much wider applicability and value. In particular, we note that our techniques could be seen as "modeling" as opposed to "inference" tools. For instance, in this paper we provide analytical and numerical approaches that allow us to discern whether an underlying cache uses LFU or LRU replacement. In many instances, however, the underlying cache may be using one of many other more elaborate replacement strategies (*e.g.*, GreedyDual, LRFU, LRU(k), *etc.*) or replacement strategies that use specific domain knowledge (*e.g.*, incorporating variable miss penalties, or variable object sizes, or expiration times, *etc.*). The fact that an underlying cache may be using a replacement strategy other than LFU and LRU does not mean that our techniques cannot be used effectively to build a robust *model* of the underlying cache. By its nature, a model is a simplifcation or an abstraction of an underlying complex reality, which is useful as long as it exhibits verifiable predictive powers that could assist/inform the processes using such models – *e.g.*, to inform resource schedulers or resource allocation processes.

The inherent value of the techniques we devise in this paper for modeling purposes is an important point that we would like to emphasize at the outset. To drive this point home, we discuss below an example in which modeling an underlying cache could be used for informed resource management.

Application to Caching-Aware Resource Management: Consider the operation of a hosting service. Typically, such a service would employ a variety of mechanisms to improve performance and to provide proper allocation of resources across all hosted web sites. For instance, the hosting service may be employing traffic shapers to apportion its out-bound bandwidth across the various web sites to ensure fairness, while at the same time, it may be using a third-party caching network to front-end popular content. For simplicity, consider the case in which two web sites A and B are hosted under equal Service Level Agreements (SLAs). As such, a natural setting for the traffic shaper might be to assign equal out-bound bandwidth to both A and B. Now, assume that web site A features content with a highly-skewed popularity profile and which is accessed with high intensity, whereas web site B features content with a more uniform popularity and which is accessed with equally high intensity. Using the techniques we devise in this paper, by observing the miss streams of A and B, the hosting service would be able to construct a "model" of the *effective* caches for A and B. We emphasize "effective" because in reality, there is no physical separate caches for A and for B, but rather, the caching network is shared by A and B (and possibly many other web sites). In other words, using the techniques we devise in this paper would allow us to construct models of the "virtual" caches for A and B. Using such models, we may be able to estimate (say) the hit rates r_A and r_B that the caching network delivers for A and B. For instance, given the different cacheability of the workloads for A and B, we may get $r_{\mathrm{A}} \gg r_{\mathrm{B}}$. Given such knoweldge, the hosting service may opt to alter its traffic shaping decision so as to allocate a larger chunk of the out-bound bandwidth to B to compensate it for its inability to reap the benefits from the caching network. This is an example of *caching-aware resource management* which benefits from the ability to build effective models of underlying caching processes.[1]

Paper Overview and Contributions: In addition to concisely formulating the cache inference problem in Section II, this paper presents a general framework that allows us to solve these problems in Section III. In Sections IV and V, we develop analytical and numerical approaches that allow us to obtain quite efficient instantiations of our inference procedure for LFU and LRU. In Section VI, we generalize our instantiations to allow for the inference of the size of the underlying cache and that of the object universe. In Section VII, we present evidence of the robustness of our inference techniques, using extensive simulations, driven with both synthetic and real traces. In Section VIII, we present experimental results that illustrate the use of our inference techniques for informed request routing. We conclude the paper in Sections IX and X with a summary of related work and on-going research.

[1] The ability to infer the relative benefits from a shared cache in order to inform resource management decisions has been contemplated quite recently for CPU scheduling in emerging multi-core architectures [7], [8].

II. Problem Statement and Assumptions

Consider an object set $O = \{o_1, \ldots, o_N\}$, where o_i denotes the ith unit-sized object.[2] Now, assume that there exists a client that generates requests for the objects in O. Let X_n denote the nth request and let $X_n = i$ indicate that the nth request refers to object o_i. The requests constitute the input of a cache memory with space that accommodates up to C objects, operating under an unknown replacement policy *ALG*.[3] If the requested object is currently cached, then X_n leads to a *cache hit*, meaning that the requested object can be sent back to the client immediately. If o_i is not currently cached, then X_n leads to a *cache miss*, meaning that the object has to be fetched first before it can be forwarded to the client, and potentially cached for future use. Let Y_m denote the mth miss that appears on the "output" of the cache, where by output we mean the communication channel that connects the cache to a subsystem that is able to produce a copy of all requested objects, *e.g.*, an "origin server" that maintains permanent copies of all objects: $Y_m = X_n$ if the mth miss is due to the nth request, $n \geq m$. In both cases (hit or miss), a request affects the information maintained for the operation of *ALG*, and in the case of a miss, it also triggers the eviction of the object currently specified by *ALG*.

Definition 1: (CIP) Given: (i1) the miss-stream Y_m, $1 \leq m \leq m_{max}$, and (i2) the cache size C and the object universe size N, find: (o1) the input request stream X_n, $1 \leq n \leq n_{max}$, and (o2) the replacement policy *ALG*.

If one thinks of *ALG* as being a function $Y =$*ALG*(X), then CIP amounts to inverting the output Y and obtaining the input X and the function *ALG* itself.

Can the above general CIP be solved? The answer is no. To see that, it suffices to note that even if we were are also given *ALG*, we still could not infer the X_n's from the Y_m's and, in fact, we cannot even determine n_{max}. For instance, consider the case in which some very popular objects appear on the miss-stream only when they are requested for the first time and never again as all subsequent requests for them lead to hits. These objects are in effect "invisible" as they can affect n_{max} and X_n in arbitrary ways that leave no sign in the miss-stream and thus cannot be infered.[4]

Since the general CIP is too unconstrained to be solvable, we lower our ambition, and target more constrained versions that are indeed solvable. A first step in this direction is to impose the following assumption (constraint).

Assumption 1: Requests occur under the Independent Reference Model (IRM): requests are generated independently and follow identical stationary distributions, *i.e.*, $P[X_n = i | X_l, 1 \leq l < n] = P[X_n = i] = p_i$, where $p = \{p_1, \ldots, p_N\}$ is a Probability Mass Function (PMF) over the object set O.

[2] The unit-sized object assumption is a standard one in analytic studies of replacement algorithms [6], [9] to avoid adding 0/1-knapsack-type complexities to a problem that is already combinatorial. Practically, it is justified on the basis that in many caching systems the objects are much smaller than the available cache size, and that for many applications the objects are indeed of the same size (*e.g.*, entries in a routing table).

[3] We refer interested readers to [10] for a fairly recent survey of cache replacement strategies.

[4] Similar examples illustrating the insolvability of CIP can be given without having to resort to such pathological cases.

The IRM assumption [6] has long being used to characterize cache access patterns [11], [12] by abstracting out the impact of temporal correlations, which was shown in [13] to be minuscule, especially under typical, Zipf-like object popularity profiles. Another justification for making the IRM assumption is that prior work [14] has showed that temporal correlations decrease rapidly with the distance between any two references. Thus, as long as the cache size is not minuscule, temporal correlations do not impact fundamentally the i.i.d. assumption in IRM. Regarding the stationarity assumption of IRM, we note that previous cache modeling and analysis works have assumed that the request distribution is stationary over some long-enough time scale. Moreover, for many application domains, this assumption is well supported by measurement and characterization studies.[5]

The IRM assumption makes CIP simpler since rather than identifying the input stream X_n, it suffices to characterize X_n statistically by infering the p of the X_n's. Still, it is easy to see that CIP remains insolvable even in this form (see [15] for details). We, therefore, make one more assumption that makes CIP solvable.

Assumption 2: The PMF p of the requests is a Generalized Power Law (GPL), *i.e.*, the ith most popular object, hereafter (without loss of generality) assumed to be object o_i is requested with probability $p_i = \Lambda / i^a$, where a is the skewness of the GPL and $\Lambda = (\sum_{i'=1}^{N} \frac{1}{i'^a})^{-1}$ is a normalization constant.

The GPL assumption allows for an exact specification of the input using only a single unknown — the skewness parameter a.[6] Combining the IRM and GPL assumptions leads to the following simpler version of CIP, which we call CIP2.

Definition 2: (CIP2) Given: (i1) the miss-stream Y_m, $1 \leq m \leq m_{max}$, and (i2) the cache size C and the object universe size N, find: (o1) the skewness parameter a of the IRM/GPL input, and (o2) the replacement policy ALG.

Before concluding this section, we should emphasize that the IRM/GPL assumptions were made to obtain a framework within which the CIP problem becomes solvable using sound analysis and/or numerical methods. However, as we will show in later sections of this paper, *the inference techniques we devise are quite robust even when one or both of the IRM and GPL assumptions do not hold*. This is an important point that we want to make sure is quite understood at this stage.

III. A General Inference Framework

Let q_i denote the appearance probability of the ith most popular object *in the miss stream* Y_m of CIP2 — let this be object $o_{g(i)}$. For a cache of size C, under LFU replacement $g(i) = C + i$, whereas under LRU replacement $g(i)$ may assume any value between i and $C + i$, depending on demand skewness and cache size (more details in Section V). In both cases, $q_i = (1/m_{max}) \sum_{m=1}^{m_{max}} I_{\{Y_m = g(i)\}}$, where $I_{\{\}}$ is the indicator function. Below, we present our general inference framework for solving CIP2. It consists of the following three steps:

[5] Obviously, if the demand is non-stationary and radically changing over shorter small time scales, then no analysis can be carried out.

[6] Without the GPL assumption the input introduces N unknowns, whereas without the IRM assumption it introduces n_{max} unknowns, that can be arbitrary many, and even exceed N.

1. Hypothesis: In this step we hypothesize that a known replacement policy *ALG* operates in the cache.
2. Prediction: Subject to the hypothesis above, we want to predict the PMF $\tilde{q}$ of the miss stream obtained when *ALG* operates on an IRM GPL input request stream. This requires the following: (i) obtain an estimate $\tilde{a}$ of the exact skewness a of the unknown GPL(a) input PMF p, and (ii) derive the steady-state hit probabilities for different objects under the hypothesized *ALG* and $\tilde{a}$, *i.e.*, $\tilde{\pi}_i$, $1 \leq i \leq N$. The $\tilde{\pi}_i$'s and the $\tilde{p}_i$'s (corresponding to a GPL($\tilde{a}$)) lead to our predicted miss-stream $\tilde{q}_i$ as follows:

$$\tilde{q}_i = \frac{\tilde{p}_{g(i)} \cdot (1 - \tilde{\pi}_{g(i)})}{\sum_{o_j \in O} \tilde{p}_{g(j)} \cdot (1 - \tilde{\pi}_{g(j)})} \tag{1}$$

3. Validation: In this step we compare our predicted PMF $\tilde{q}$ to the PMF q of the actual (observed) miss stream, and decide whether our hypothesis about *ALG* was correct. In order to define a similarity measure between $\tilde{q}$ and q, we view each PMF as a (high-dimensional) vector, and define the distance between them to be the L_p-norm distance between their corresponding vectors. Thus, the error between the predicted PMF and the observed one is given by $e(\tilde{q}, q) = (\sum_{i=1}^{N} (\tilde{q}_i - q_i)^p)^{\frac{1}{p}}$. In this work, unless otherwise stated, we use the L_2-norm. A positive validation implies a solution to CIP2 as we would have inferred both the unknown replacement algorithm and the PMF of the input stream, using only the observed miss stream.

In the above procedure, the prediction step must be customized for different *ALG*s assumed in the hypothesis step. In Sections IV and V, we do so for LFU and LRU, using a combination of analytical and fast numerical methods.

The above inference procedure has important advantages over an alternative naive "simulation-based" approach that assumes an $\tilde{a}$ and an *ALG*, performs the corresponding simulation, and then validates the assumptions by comparing the actual and the predicted PMFs. While the overall approach is similar, we take special care to employ analytic or fast numeric techniques where possible and thus avoid time-consuming simulations (mainly for the prediction step). Such advantages will be evident when we present our analytical method for deriving $\tilde{a}$ under LFU, and the corresponding analytical and fast numerical methods for obtaining the predicted miss streams for both LFU and LRU. Needless to say, fast inference is important if it is to be performed in an on-line fashion.

IV. The Prediction Step for LFU

The main challenge here is to obtain an estimate of a by using the observed miss stream q of an LFU cache. We start with some known techniques from the literature and then present our own SPAI analytical method. Having obtained the estimate $\tilde{a}$ and the corresponding $\tilde{p}$, it is straightforward to construct the predicted PMF $\tilde{q}$ of the actual miss stream PMF q. This is so because LFU acts as a cut-off filter on the C most-popular objects in the request stream, leading to the following relationship for $\tilde{q}$.

$$\tilde{q}_i = \frac{\tilde{p}_{C+i}}{1 - \sum_{j=1}^{C} \tilde{p}_j}, \qquad 1 \leq i \leq N - C \tag{2}$$

A. Existing Methods: MLE, OLS, and RAT

There are several methods for estimating the skewness of a power-law distribution through samples of its tail. In this section we will briefly present three popular ones: the Maximum Likelihood Estimator (MLE), the Linear Least Squares (OLS) estimator, and the RAT estimator. MLS and OLS are numeric methods of significant computational complexity, whereas RAT is analytic.

*The Maximum Likelihood Estimator (*MLE*):* Given a vector of N miss stream observations x we would like to determine the value of the exponent α which maximizes the probability (likelihood) of the sampled data [16]. Using the power law input demand, we can derive an equation that can be solved using standard iterative numerical methods and provide an unbiased estimate of α [15].

*An Alternative Analytical Estimator (*RAT*):* MLE is asymptotically optimal. In practice we may be interested in obtaining an estimate with a limited number of observations. A (poorer) estimator uses q_{C+1} and q_{C+2} to get α. Equating q_i to $i^{-\alpha} \div \sum_{j=C+1}^{U} j^{-\alpha}$ we get

$$\frac{q_{C+1}}{q_{C+2}} = \frac{(C+1)^{-\alpha}}{(C+2)^{-\alpha}} \tag{3}$$

$$\log \frac{q_{C+1}}{q_{C+2}} = -\alpha \log(C+1) + \alpha \log(C+2) \tag{4}$$

$$\alpha = \frac{\log(q_{C+1}/q_{C+2})}{\log(C+2) - \log(C+1)} \tag{5}$$

*A Linear Least-Squares Estimator (*OLS*):* Yet another method to estimate a is to use linear least-squares estimation on the $\log - \log$ plot of q_i, *i.e.*, the PDF of the miss stream. This graphical method is well documented and perhaps one of the most commonly used (*e.g.*, in [17]).

B. SPAI: Our Single Point Analytic Inversion Method

As its name suggests, SPAI considers a single (measurement) "point" – the appearance frequency of the most popular object in the miss stream of an LFU cache, which is subject to GPL demand with skewness parameter a – and uses it to derive an estimate of a. A brief overview of SPAI goes as follows: We start with an equation that associates the measured frequency q_1 at the miss-stream with the request frequency p_{C+1} of the unknown input stream. We use our assumption (that the input is GPL) to substitute all p_i's by analytic expressions of i and a. We employ a number of algebraic approximations to transform our initial non-algebraic equation into a polynomial equation of our single unknown a. We then obtain a closed-form expression for a by solving a corresponding cubic equation obtained by an appropriate truncation of our original polynomial equation. The details are presented next.

$$q_1 = \frac{p_{C+1}}{\left(\sum_{i=1}^{N} p_i\right) - \left(\sum_{i=1}^{C} p_i\right)} = \frac{1/(C+1)^a}{\left(\sum_{i=1}^{N} 1/i^a\right) - \left(\sum_{i=1}^{C} 1/i^a\right)}$$
$$= \frac{1/(C+1)^a}{H_N^{(a)} - H_C^{(a)}}$$

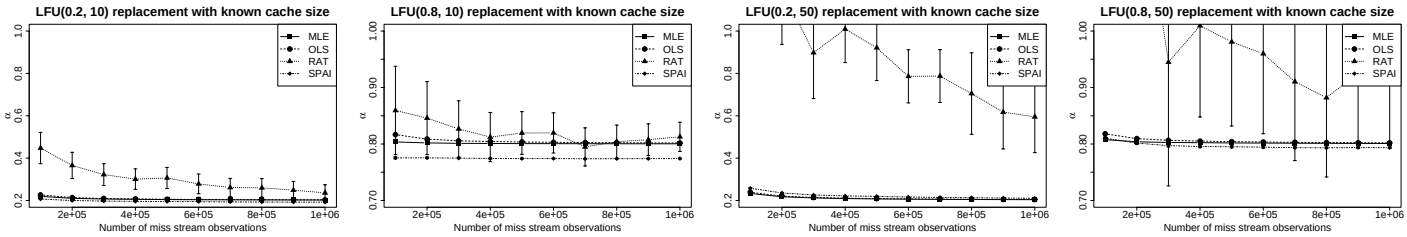

Fig. 1. SPAI vs MLE, OLS, and RAT on inferring the exponent of a GPL from samples of its tail (objects with rank $C+1$ and higher) shown with 95% confidence intervals.

Using the integral approximation $H_k^{(a)} \approx \frac{k^{1-a}-1}{1-a}$ for the kth generalized Harmonic number of order a, $H_k^{(a)}$, we can write:

$$q_1 \approx \frac{1/(C+1)^a}{\frac{N^{1-a}-1}{1-a} - \frac{C^{1-a}-1}{1-a}} = \frac{1-a}{(C+1)^a(N^{1-a}-C^{1-a})}$$

For large C we have $(C+1)^a \approx C^a$, leading to the following approximation.

$$q_1 \approx \frac{1-a}{C^a(N^{1-a}-C^{1-a})} = \frac{1-a}{C^aN^{1-a}-C} = \frac{(1-a)C^{-1}}{(N/C)^{1-a}-1}$$

After term re-arrangement, we re-write the last equation as

$$(N/C)^{1-a} - \frac{1}{q_1 C}(1-a) - 1 = 0 \tag{6}$$

or equivalently as

$$\beta^x - \gamma x - 1 = 0 \qquad \text{where:} \quad \begin{array}{l} x = 1-a < 1 \\ \beta = N/C > 1 \\ \gamma = \dfrac{1}{\tilde{p}_1 C} > 0 \end{array} \tag{7}$$

Expanding the exponential form β^x on x around point 0 we get $\beta^x = 1 + \sum_{k=1}^{\infty} \frac{(x\ln\beta)^k}{k!}$. By substituting in Eq. (7), we get the following *master equation*:

$$\left(\sum_{k=1}^{\infty} \frac{(x\ln\beta)^k}{k!}\right) - \gamma x = 0 \tag{8}$$

Equation (8) can be approximated through "truncation", *i.e.*, by limiting k to K instead of ∞. For $K=2$ we get a quadratic equation which has one non-zero real solution. For $K=3$ we get a cubic equation with two real solutions and we can choose the positive one. Finally, for $K=4$ we get:[7]

$$\frac{1}{24}x\left(x^3\log^4(\beta) + 4x^2\log^3(\beta) + 12x\log^2(\beta) + 24\log(\beta) - 24\gamma\right) = 0 \tag{9}$$

At least one of the roots of the cubic equation in the parentheses of Eq. (9) is real — we select the one in $[0,1]$ and use it to obtain the skewness parameter a through $a = 1-x$.

[7] Notice that for $K=4$ we actually have a cubic equation of x (inside the parentheses) after factoring out the common term x. We can therefore use the cubic formula [18] to obtain a closed-form solution for the unknown x. Theoretically we could go even further and consider $K=5$, which would put a quartic equation in the parentheses. The solution of the general quartic equation, however, involves very cumbersome formulas for the roots and is marginally valuable since the cubic equation already provides a close approximation as will be demonstrated later on. $K=5$ and the quartic equation is actually as far as we can go because for $K=6$ we would have a quintic equation in the parentheses for which there does not exist a general solution over the rationals in terms of radicals (the "Abel-Ruffini" theorem).

C. SPAI Versus Existing Approaches

We perform the following experiment in order to compare the performance of SPAI to existing approaches that could be used for the prediction step. We simulate an LFU cache of size C, driven by a $GPL(a)$ input request stream. In Figure 1 we plot the estimated skewness $\tilde{a}$ obtained from SPAI and the other methods (on the y-axis) for different numbers of miss-stream samples (on the x-axis). We use low ($a=0.2$) and high ($a=0.8$) skewness, and low $C/N=1\%$ and high ($C/N=5\%$) relative cache sizes. These results indicate that SPAI performs as well as (and most of the time better than) MLE and OLS, over which it has the additional advantage of being *an analytical method*, thus incurring *no* computational complexity. SPAI's rival analytical method, RAT, performs much worse.

V. The Prediction Step for LRU

The steady-state hit probabilities of LRU bear no simple characterization, and this has several consequences on the prediction step of our cache inference framework presented in Section III. First, even if we had $\tilde{a}$, it is not trivial to derive the $\tilde{\pi}_i$'s (and therefore get the $\tilde{q}_i$'s through Eq. (1)). Computing exact steady-state hit probabilities for LRU under IRM is a hard combinatorial problem, for which there exist mostly numerical approximation techniques [19], [20], [21]. Second, and most importantly, the steady-state hit probabilities depend on the skewness of the input through a function $\pi(a,C)$. Since we do not have a simple analytic expression for $\pi(a,C)$, it is impossible to invert Eq. (1) and obtain a closed-form expression for a, as with our SPAI method for LFU. Unfortunately, our analytical derivation of $\pi(a,C)$ [22] cannot be used for this purpose as it involves a complex non-algebraic function[8] that leads to a final equation for a (through Eq. (1)) that admits no simple closed-form solution.

In light of the discussion above, our approach for the prediction step for LRU replacement is the following. We resort to a numeric search technique for obtaining $\tilde{a}$, *i.e.*, we start with an arbitrary initial value $a_0 \in [0,1]$, compute the corresponding steady-state hit probabilities $\tilde{\pi}(a_0)$ using either the numeric technique of Dan and Towsley [19], or our own analytical one from [22], and compute the miss probability for each object of the input. Next, we sort these in decreasing popularity to obtain the predicted miss stream $\tilde{q}(a_0)$ and, finally, the error $e(\tilde{q}(a_0), q)$. We then use a local search approach to find the $\tilde{a}$ that minimizes the error e.

[8] The unknown a appearing in complex polynomial and exponential forms over multiple different bases.

	C=20		C=40		C=60		C=80		C=100		C=120		C=140		C=160		C=180		C=200	
$\alpha = 0.2$	0.20	20	0.20	41	0.20	49	0.20	75	0.19	66	0.18	66	0.17	75	0.15	51	0.16	78	0.15	78
$\alpha = 0.4$	0.41	24	0.40	41	0.39	51	0.39	66	0.39	88	0.38	97	0.38	118	0.36	107	0.36	134	0.37	158
$\alpha = 0.6$	0.60	21	0.62	46	0.60	63	0.60	82	0.58	89	0.56	96	0.54	100	0.56	136	0.54	141	0.55	160
$\alpha = 0.8$	0.80	21	0.81	42	0.81	61	0.79	76	0.79	93	0.78	109	0.80	135	0.78	147	0.78	165	0.80	200
$\alpha = 1.0$	1.04	24	1.01	41	1.01	63	1.00	82	0.99	97	1.01	127	0.96	125	1.00	159	1.00	179	0.96	180

TABLE I

INFERENCE USING OUR $1-$DIMENSIONAL SEARCH METHOD ON THE MISS-STREAM OF AN LFU(α, C) CACHE.

	C=20		C=40		C=60		C=80		C=100		C=120		C=140		C=160		C=180		C=200	
$\alpha = 0.2$	0.31	250	0.31	250	0.27	225	0.28	250	0.27	225	0.30	250	0.24	188	0.22	183	0.23	225	0.27	250
$\alpha = 0.4$	0.40	27	0.41	75	0.41	75	0.40	101	0.42	135	0.42	135	0.41	154	0.43	197	0.43	209	0.39	187
$\alpha = 0.6$	0.65	41	0.61	51	0.58	58	0.58	88	0.64	119	0.60	124	0.64	161	0.61	161	0.63	192	0.60	202
$\alpha = 0.8$	0.80	25	0.87	45	0.77	58	0.86	101	0.83	119	0.79	119	0.80	144	0.84	174	0.81	187	0.85	207
$\alpha = 1.0$	0.93	14	0.93	38	0.97	58	0.91	57	0.94	107	0.93	106	0.98	139	0.98	156	0.91	166	0.93	194

TABLE II

INFERENCE USING OUR $2-$DIMENSIONAL SEARCH METHOD ON THE MISS-STREAM OF AN LRU(α, C) CACHE.

VI. INFERRING SIZE OF CACHE AND OBJECT UNIVERSE

In this section we show how to extend the prediction techniques of the previous sections to make them useful under unknown cache size C and object universe size N.

A. Handling LFU Using One-dimensional Search

Let U denote the number of unique distinct objects that appear on the miss stream Y_m of an LFU cache. Assume also that C_{max} is an upper bound on the (unknown) cache size. We can pick an arbitrary cache size $\tilde{C} \leq C_{max}$. This leads directly to an object universe size $\tilde{N} = U + \tilde{C}$. Our approach will be to perform a numeric search by setting our "free" variable $\tilde{C}$, obtain the corresponding $\tilde{N}$, applying our LFU test using $(\tilde{C}, \tilde{N})$, computing the corresponding error $e(\tilde{q}(\tilde{C}, \tilde{N}), q)$, and changing $\tilde{C}$ in the direction of minimizing the aforementioned error between the observed and predicted PMFs.

B. Handling LRU Using Two-dimensional Search

A similar approach can be used for LRU. In this case, however, there is a non-zero probability that any object may appear on the miss stream, and thus $U = N$. This means that the numeric search over $\tilde{C}$ doesn't effect N. However, since the standard prediction step for LRU already includes a numeric search over the skewness $\tilde{a}$, it follows that the resulting numeric search when N and C are not known will have to be over a two-dimensional free variable $(\tilde{a}, \tilde{C})$. We use a general non-convex optimization method, namely the differential evolution method [23], to find the best values for $\tilde{a}$ and $\tilde{C}$.

VII. PERFORMANCE EVALUATION

In this section, we verify the quality of the results obtained through the application of our inference framework (for both LRU and LFU), when N and C are unknown. We will start by demonstrating the very high accuracy of our inference techniques in a conformant setting, in which the assumptions used for designing our methods are satisfied. Next, we demonstrate the robustness of our methods in non-conformant settings, in which these assumptions do not hold. We do so by using real datasets, which are neither strictly GPL, nor IRM, and by applying our inference techniques when a replacement policy other than "pure" LFU or LRU is in use.

A. Results in Conformant Settings

Our goal is to show that 1D and 2D searches for LFU and LRU under unknown N, C are possible as: (1) when we have a perfect PMF of a miss-stream of an exact GPL input, then the error is convex and thus such searches find an almost exact solution; (2) even when the miss-stream PMF is not perfect (not produced from an infinite number of samples), meaning it contains some sampling noise, still our methods detect a, C quite accurately.

Table I presents the results of our $1-$dimensional search method on the miss-stream of an LFU(α, C) cache. The input demand is synthetically generated by sampling a $GPL(\alpha)$ distribution. Our inference method is applied on the miss-stream once 10^6 misses have been observed. Similarly, Table II presents the results of our $2-$dimensional search method on the miss-stream of an LRU(α, C) cache. Each cell in these tables includes the inferred $(\tilde{a}, \tilde{C})$ pair corresponding to the actual (a, C) pair defined by the coordinates of the cell. As it may be seen from these results, the inferred values are quite accurate in most cases. Generally, a is easier to be inferred with high accuracy than C. A possible reason for this is that since the a characterizes the entire distribution, it suffices to look at the correct place for getting a robust estimate of it. The correct place in this case is the initial most popular objects of the miss-stream. These objects stabilize quickly around their expected appearance probabilities after relatively few samples thus revealing the skewness of the GPL; SPAI is largely build around this observation. Regarding C, we see that getting a robust estimation on it becomes increasingly difficult with decreasing skewness and increasing cache size. Our explanation for this is that in these cases, the objects that are around the border of the cache tend to have similar, very low request probabilities and thus it becomes increasingly harder to estimate the exact cache size.

B. Results under Non-Conformant Input

To check the robustness of our techniques, we conducted experiments using the following "real" datasets, for which the pefect GPL and IRM assumptions do not hold.

- *Stocks:* A stock quote datastream [24] collected on April 4, 2006, comprising 937,565 requests for its top 400 most-popular objects

- *UCB:* UC Berkeley Home IP Web Traces [25] of 2,575,144 references to over 5,000 objects, spanning the period from November 1, 1996 to November 19, 1996.

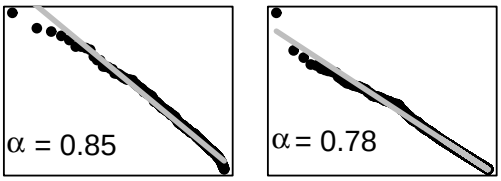

Fig. 2. Stocks (left) and UCB (right) datasets used in our experiments (and GPL(α) approximation thereof).

We subject each input data stream to LFU replacement with a cache size C and we infer the value of α by using SPAI. Given this estimate of the power-law exponent, $\hat{\alpha}$, we attempt to minimize the error between the observed miss-stream and the miss-stream that would result from performing LFU replacement with a cache size $\hat{C}$ on a GPL($\hat{\alpha}$) distributed input. The value $\hat{C}$ which minimizes the L_2 error between the two vectors is our guess for C. The results are presented in Tables III and IV. Despite the fact that these datasets are not strictly GPL in popularity profile, the obtained estimation is quite accurate.

Unlike LFU, LRU is sensitive to temporal correlations of the input stream and these datasets include a substantial amount of such correlation [26]. To quantify the effect of temporal correlations on our method we produced a random permutation of the Stocks dataset and subjected it to LRU replacement for the same range of cache sizes as the original. As can be seen in Table IV the effect of this shuffling is a greater miss rate, which implies more information for our algorithm and hence greater accuracy in our results. Having said that we note that even when the IRM assumptions breaks, our results are fairly accurate and can be further improved by collecting a larger number of observations.

ALG	C	20	40	60	80	100
LFU	$\hat{C}$	46	70	85	93	87
	$\hat{\alpha}$	0.89	0.9	0.88	0.85	0.8
LRU	$\hat{C}$	25	53	70	91	110
	$\hat{\alpha}$	0.73	0.76	0.78	0.81	0.82

TABLE III
ESTIMATING α AND C FOR THE UCB DATASET

C. Results under Non-Conformant Replacement

Reference locality is the main criterion that caching subsystems capitalize on. Various replacement policies exploit different manifestations of reference locality in the request stream. The two canonical examples being LFU, which exploits frequency, and LRU, which exploits recency. More nuanced replacement policies attempt to exploit both of these basic manifestations of reference locality (as well as others). In this section, we show that our inference techniques are able to robustly discern whether an underlying replacement operates "more like" LFU than LRU (or vice versa).

To that end, we implemented a hybrid replacement strategy Hybrid(λ), which splits the available storage space C in two parts of size $(1-\lambda)C$ and λC respectively, where $0 \leq \lambda \leq 1$. The first part is devoted to LFU replacement, effectively storing the most popular objects in the input stream. The second part is used to perform LRU replacement on the rest of the objects in our universe. By varying λ, we are able to make Hybrid(λ) behave "more like" either LFU or LRU, with Hybrid(0)=LRU and Hybrid(1)=LFU. In this section, we consider the case in which Hybrid(λ) is the unknown replacement algorithm operating in the cache.[9]

ALG	C	10	20	30	40	50
LFU	$\hat{C}$	20	33	34	49	43
	$\hat{\alpha}$	0.99	1.01	0.93	0.96	0.86
LRU	$\hat{C}$	23	32	41	48	58
	$\hat{\alpha}$	0.73	0.73	0.73	0.72	0.73
	Miss-rate	46.30%	40.10%	35.79%	32.44%	29.66%
LRU (shuffled)	$\hat{C}$	14	22	32	44	56
	$\hat{\alpha}$	0.82	0.85	0.88	0.93	0.96
	Miss-rate	90.99%	83.27%	76.55%	70.74%	65.59%

TABLE IV
ESTIMATING α AND C FOR THE STOCKS DATASET. BOTTOM ROW SHOWS RESULTS WHEN REQUEST STREAM IS SHUFFLED TO SATISFY IRM.

Let $\delta(\lambda, \text{LRU})$ denote the *real* L_2 distance between the PMFs of the miss streams of a Hybrid(λ) cache and an LRU cache *driven by the same GPL(a) request stream* assuming we know the probability of occurrence of every object in the miss-stream, including the objects we never see, *i.e.*, those whose probability is zero. We similarly define $\delta(\lambda, \text{LFU})$. For large λ, Hybrid(λ) is closer to LFU and, thus, $\delta(\lambda, \text{LRU}) - \delta(\lambda, \text{LFU})$ will be positive. Inversely, for small λ, Hybrid(λ) is closer to LRU and, thus, $\delta(\lambda, \text{LRU}) - \delta(\lambda, \text{LFU})$ will be negative.

The normalized difference between $\delta(\lambda, \text{LRU})$ and $\delta(\lambda, \text{LFU})$ — namely $(\delta(\lambda, \text{LRU}) - \delta(\lambda, \text{LFU}))/(\delta(\lambda, \text{LRU}) + \delta(\lambda, \text{LFU}))$ could be used as an indicator function ($\bar{I}$). A more positive (closer to $+1$) value of $\bar{I}$ indicates that the underlying replacement is "more like" LFU, whereas a more negative (closer to -1) value of $\bar{I}$ indicates that the underlying replacement is "more like" LRU. Notice that an oracle that also knows the input rank of the objects would be able to obtain an even more authoritative indicator function, which we call I.

Computing $\bar{I}$ (not to mention I) is not possible, if all what we are given is the miss stream from a cache with unknown characteristics. In particular, we are not given the GPL(a) request stream nor are we privy to the replacement strategy or its parameters (λ in our case), and thus we cannot compute I. For that we must rely on an estimate of the indicator function. We do so next, using the methods of Sections IV and V.

Let $e(\lambda, \text{LRU})$ be the error between the PMF of the observed miss stream and the predicted PMF for that stream, obtained by executing our LRU test from Section V on the underlying Hybrid(λ) replacement algorithm. Similarly, let $e(\lambda, \text{LFU})$ denote the corresponding error from applying the LFU test from Section IV.[10] The normalized difference between $e(\lambda, \text{LRU})$ and $e(\lambda, \text{LFU})$ could be use as an estimate $\hat{I}$ of the indicator function I. Figure 3 shows how $\hat{I}$ tracks I and $\bar{I}$, and thus provides for an efficient, robust discrimination between different flavors of Hybrid(λ).

[9] We note that Hybrid(λ) resembles the Least Recently/Frequently Used (LRFU) replacement policy of Lee *et al.* [27]. LRFU has a parameter λ, $0 \leq \lambda \leq 1$, which allows LRFU to subsume the entire spectrum of policies between in-cache LFU (as opposed to ideal LFU) and LRU. LRFU degenerates to in-cache LFU for $\lambda = 0$ and to LRU for $\lambda = 1$, whereas it becomes a joint frequency/recency based replacement policy for intermediate values of λ.

[10] In both cases we assume that objects are sorted by their miss-stream popularity as their actual input rank is in most cases unknown.

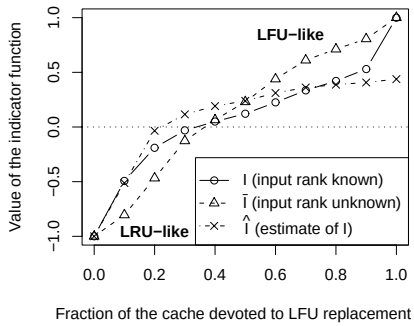

Fig. 3. Computed indicator function $\hat{I}$ versus the real indicator function $\bar{I}$ and the ideal I indicator function.

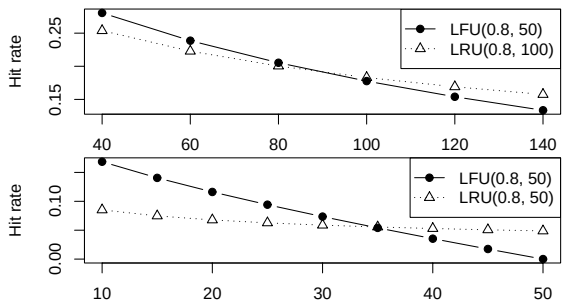

Fig. 4. Hit rates resulting from routing the miss stream of an LRU (top) and LFU (bottom) caching gateway to an LFU versus LRU peer cache.

VIII. Application to Informed Request Routing

In this section, we revisit one of the examples presented in Section I to motivate the CIP problem. We present results that quantify the benefits to a client-side caching proxy P_0 (the gateway) from being able to route its miss stream to one of many candidate peer proxy caches P_i, $1 \leq i \leq n$, based on knoweldge acquired using our inference techniques about the characteristics of the caches at these proxies.

In the experiments that follow, we assume that the input request streams for the set of cooperating caching agents P_i, $0 \leq i \leq n$, follow GPL profiles with common rank, but with different skew over the object universe O. Such an assumption is consistent with a setting in which the caches cater to different local populations, with consistent popularity ranking of objects, but with different absolute preferences.[11] Also, we note that as is typical in caching networks [4], [5], requests routed to a proxy cache from a remote peer do not affect the local state of the proxy cache (as opposed to a hierarchical caching system in which a lower level cache operate as an L2 cache). Thus the gateway's routing of its miss stream to a proxy P_i does not change the state of P_i's cache.[12]

Let a_i denote the skewness of demand presented to P_i, where $0 \leq i \leq n$. Also, let C_i, ALG_i, and π_i denote the storage capacity, the replacement algorithm, and the steady-state hit vector of proxy P_i, respectively. Suppose that the gateway P_0 uses our cache inference technique to effectively infer (a_i, C_i, ALG_i) for each one of its peer proxies P_i, $1 \leq i \leq n$. Then depending on ALG_i, P_0 can employ an analytical or numerical technique to derive π_i (a_i,C_i, ALG_i), *i.e.*, the per-object hit probabilities at P_i. In such a case, the gateway's routing of its miss stream to P_i would yield a hit ratio h_i and a service cost c_i, where h_i=$q_0(a_0)$*π_i(a_i,C_i, ALG_i), and $c_i = d(P_0,P_i) \cdot h_i$, respectively.[13] Thus, the gateway P_0 can choose an optimal proxy $P_{i^\star}$ so as to obtain $c_{i^\star}$, a minimal service cost for its miss-stream.

To quantify the benefit from using the above informed request routing, we present results from two sets of experiments in which a gateway P_0 uses our cache inference techniques to inform its decision regarding which one of two peer caching proxies (P_1 and P_2) to select as targets of its own miss stream.

[11] In other words, while object o_i is the i^{th} most popular object for all P_i, $0 \leq i \leq n$, its access probability p_i is different across all P_i, $0 \leq i \leq n$.

[12] In [28] we showed that cache pollution phenomena may arise if such decoupling of cache state is not in place.

[13] By "*" we denote the inner product operator for vectors, whereas $d(P_0,P_i)$ denotes the distance (say, delay) between P_0 and P_i).

In the first experiment, the gateway P_0 operates as an $LRU(0.6, C)$ — *i.e.*, it employs an LRU cache of size C, driven by a local reference stream with skew 0.6 — over a $1,000$-object universe. Furthermore, the gateway P_0 may direct its miss stream to proxy P_1, which operates as an $LFU(0.8, 50)$, or to proxy P_2, which operates as an $LRU(0.8, 100)$. Figure 4 (top) shows the hit rates (on the y-axis) that the LRU gateway's miss stream is able to secure at each one of the two proxies, P_1 and P_2, while varying the gateway's cache size C (on the x-axis). The results of this experiment show for small local cache size, the gateway is better off routing its miss stream to the LFU proxy P_1, whereas for large local cache sizes, the gateway is better off routing its miss stream to the LRU proxy P_2.

In the second experiment, the gateway P_0 operates as an $LFU(0.6, C)$ cache, whereas proxies P_1 and P_2 operate as $LFU(0.8, 50)$ and $LRU(0.8, 50)$, respectively, over a $1,000$-object universe. Figure 4 (bottom) shows the hit rates (on the y-axis) that the LFU gateway's miss stream is able to secure at each one of the two proxies, P_1 and P_2, while varying the gateway's cache size C (on the x-axis). These results show a trend similar to that observed for an LFU gateway, albeit with a much more pronounced difference in hit rates for small and large local cache sizes, suggesting that the benefit from employing our inference techniques could be significant (in terms of the relative hit rates achievable at P_1 versus P_2).

IX. Related Work

The only previous work on the cache inference problem that we are aware of is due to Burnett et al. [29]. This work presents the design and performance evaluation of the *Dust* software used for inferring the buffer management policy of popular operating systems. *Dust* is shown to be able to infer both the available buffer capacity and the employed replacement strategy. It achieves its goal by using long streams of probe requests to bring the underlying buffer into a controlled initial state and then inferring the replacement strategy by observing the transitions out of this state with additional probe requests. The fundamental difference with our work is that the inference technique employed by *Dust* is *active*, in the sense that it relies heavily on the use of the aforementioned probes. Long probe streams are feasible since the *Dust* software and the unknown buffer management sub-system reside within the same physical host and, thus, there's no communication cost to be paid for the probes. Our methods on the other hand are *passive*, in the sense that they rely solely on the miss-stream caused by

a cache's own input stream (which is also unknown). Such passive operation is more suitable in distributed environments, in which bringing the target caching system to a "controlled state" (through active probing) is not feasible (not to mention that it could be construed as adversarial behavior!)

Vanichpun and Makowski [30], [31], [32] have presented a series of papers on the characterization of the miss stream of a cache under different input request models (independent reference model, higher-order Markov chain model, partial Markov chain model, LRU stack model). These works focus on the effect of locality of reference, either due to long term popularity or due to short term temporal correlation, on the miss-stream and the cache miss ratio and, therefore, examine caching in the inverse direction as compared to what we do; they are trying to characterize the output given the input, whereas we try to characterize the input (and what is in the box) given the output.

The analysis of replacement strategies is a decades-old problem, see for example [19], [20], [21], [22], [28] and references therein. These works assume a given model for requests and derive numerical or analytical approximate per-object hit probabilities under given replacement strategies. We use these methods as stubs in our work when testing hypotheses regarding the unknown replacement algorithm that we try to infer.

X. Summary and Conclusion

This paper provides a succint definition of the cache inference problem, and presents effective and robust procedures for the inference of the attributes of a remote caching agent, as well as the characteristics of the request stream to which it is subjected, using only passive observation of the cache miss stream. In addition to demonstrating the robustness of our procedures, we presented results that illustrate the potential utility of our techniques to informed request/content routing.

Our current work is focusing on the use of our inference techniques for *modeling* purposes. In doing so, our goal is not to discover the specifics of the caching solution deployed at a remote caching agent or service (which could be quite elaborate), but rather our goal is to obtain a *model* of the remote cache – a model that could be used to predict the implications from local resource management decisions. In that regard, we are investigating the possible integration of our techniques into content and request routing applications that could benefit from such information.

Appendix

The input and the output are connected through the following expression:

$$q_i = \frac{p_i \cdot (1 - \pi_i(p, ALG))}{\sum_{o_j \in O} p_j \cdot (1 - \pi_j(p, ALG))} \qquad (10)$$

where $\pi_i(p, ALG)$ denotes the steady-state probability of finding object o_i in the cache under the IRM with mass function p, and *ALG*. One could think that if we could write the steady-state probability as a linear function of the input, i.e., $\pi(p, ALG) = b_{ALG} \cdot p + c_{ALG}$, where b_{ALG}, c_{ALG} are vectors with N constants, capturing the operation of the replacement algorithm *ALG*, then Eq. (10) would produce a system of N equations (one for each object o_i) on N unknowns p_i that, assuming we knew *ALG*, could be solved for finding the unknowns. One could then assume some replacement algorithm *ALG* for which it has b_{ALG}, c_{ALG}, solve the system to obtain p, get q from Eq. (10), and then compare the obtained q with the actual observed miss-stream to validate whether the assumption regarding *ALG* was correct or not[14] . This would give a way of obtaining p and *ALG* under IRM and arbitrary p.

The previous approach, however, is very unlikely to apply due to the fact that it seems impossible to capture the operation of a replacement algorithm by a linear function. For replacement algorithms like LRU and FIFO: (1) exact mathematical representations of $\pi(p, ALG)$ involve non-linearities in the form of products of multiple unknowns p_i's[15] , and (2) approximate representations are numeric and thus cannot help in writing $\pi(p, ALG)$ as a function of the unknowns p (the only closed-form result that we are aware of is our recent work on LRU, but even this one involves a $\pi(p, LRU)$ that is a non-linear function of p, and applies only to p that have a GPL probability mass).

Regarding LFU replacement, then Eq. 10 does not help because it fails to capture the initial C most popular objects. These objects have a steady-state probability of 1, thus disappear from the formula and reduce the number of available equations to $N - C$. We then cannot obtain the p_i's for these objects.

References

[1] "Gnutella2 developer network: Known hub cache and hub cluster cache & node route cache and addressed packet forwarding," http://www.gnutella2.com/index.php.

[2] "Domain names - implementation and specification," http://tools.ietf.org/html/rfc1035.

[3] "The gnutella web caching system," http://www.gnucleus.com/gwebcache/.

[4] "The ircache project," http://www.ircache.net/.

[5] "The squid web proxy cache," info at http://www.squid-cache.org/.

[6] E. G. Coffman and P. J. Denning, *Operating systems theory*. Prentice-Hall, 1973.

[7] A. Fedorova, M. Seltzer, C. Small, and D. Nussbaum, "Performance Of Multithreaded Chip Multiprocessors And Implications For Operating System Design," in *Proceedings of USENIX 2005 Annual Technical Conference*, April 2005.

[8] A. Fedorova, M. Seltzer, and M. D. Smith, "A Non-Work-Conserving Operating System Scheduler for SMT Processors," in *Proceedings of the ISCA Workshop on the Interaction between Operating Systems and Computer Architecture (WIOSCA)*, June 2006.

[9] P. R. Jalenkovic and A. Radovanovic, "Asymptotic insensitivity of least-recently-used caching to statistical dependency," in *Proc. of IEEE Infocom*, San Francisco, CA, 2003.

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

[11] M. F. Arlitt and C. L. Williamson, "Web server workload characterization: the search for invariants," in *Proc. of ACM SIGMETRICS '96*, Philadelphia, Pennsylvania, United States, 1996, pp. 126–137.

[12] L. Breslau, P. Cao, L. Fan, G. Philips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *Proc. IEEE Infocom*, New York, 1999.

[13] S. Jin and A. Bestavros, "Sources and characteristics of web temporal locality," in *Proc. IEEE/ACM Mascots '00*, San Fransisco, CA, 2000.

[14] More details on this solution method in Sect. XXX

[15] In fact such formulas are not practical even for producing numeric results since they have computational complexities that are exponentially related to N, C (see the related work sections in [21], [22] and references therein)

[14] K. Psounis, A. Zhu, B. Prabhakar, and R. Motwani, "Modeling correlations in web traces and implications for designing replacement policies," *Computer Networks*, vol. 45, 2004.
[15] N. Laoutaris, G. Zervas, A. Bestavros, and G. Kollios, "The Cache Inference Problem and its Application to Content and Request Routing (Extended Version)," CS Department, Boston University, Tech. Rep. BUCS-TR-2006-017, July 2006.
[16] J. Aldrich, "R.A. Fisher and the making of maximum likelihood 1912-1922," *Statistical Science*, vol. 12, no. 3, pp. 162–176, 1997.
[17] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 835–846, 1997.
[18] R. Nickalls, "A new approach to solving the cubic: Cardan's solution revealed," *The Mathematical Gazette*, vol. 77, pp. 354–359, 1993.
[19] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proceedings of ACM SIGMETRICS '90*, 1990, pp. 143–152.
[20] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, 2002.
[21] N. Laoutaris, H. Che, and I. Stavrakakis, "The LCD interconnection of LRU caches and its analysis," *Performance Evaluation*, vol. 63, no. 7, pp. 609–634, 2006.
[22] N. Laoutaris, "A closed-form method for LRU replacement under generalized power-law demand," *submitted in Performance Evaluation*, 2006.
[23] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.
[24] "Inet ats, inc. http://www.inetats.com/."
[25] S. D. Gribble, "Uc berkeley home ip http traces," July 1997, available at http://www.acm.org/sigcomm/ITA/.
[26] F. Li, C. Chang, G. Kollios, and A. Bestavros, "Characterizing and exploiting reference locality in data stream applications," *icde*, vol. 0, p. 81, 2006.
[27] D. Lee, J. Choi, J. Kim, S. Noh, S. Min, Y. Cho, and C. Kim, "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Transactions on Computers*, vol. 50, no. 12, pp. 1352–1361, 2001.
[28] N. Laoutaris, G. Smaragdakis, A. Bestavros, and I. Stavrakakis, "Mistreatment in distributed caching groups: Causes and implications," in *Proc. of IEEE Infocom '06*, Barcelona, Spain, 2006.
[29] N. Burnett, J. Bent, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "Exploiting gray-box knowledge of buffer-cache management," in *Proc. of USENIX '02*, Monterey, California, USA, 2002, pp. 29–44.
[30] S. Vanichpun and A. M. Makowski, "Comparing strength of locality of reference popularity, majorization, and some folk theorems," in *Proc. of IEEE Infocom '04*, Hong Kong, PRC, Apr. 2004.
[31] ——, "The output of a cache under the independent reference model: where did the locality of reference go?" in *Proc. of ACM SIGMETRICS '04*, New York, NY, USA, 2004, pp. 295–306.
[32] ——, "Modeling locality of reference via notions of positive dependence - some mixed news," in *Proc. of IEEE Infocom '06*, Barcelona, Spain, 2006.