Performance Evaluation for New Web Caching Strategies Combining LRU with Score Based Object Selection

Gerhard Hasslinger
Deutsche Telekom
Darmstadt, Germany
gerhard.hasslinger@telekom.de

Konstantinos Ntougias
Athens Information Technology
Athens, Greece
kontou@ait.gr

Frank Hasslinger
Darmstadt Univ. of Tech.
Darmstadt, Germany
frank.hasslinger@stud.tu-darmstadt.de

Oliver Hohlfeld
RWTH Aachen
Aachen, Germany
oliver@comsys.rwth-aachen.de

***Abstract* — The topic of Internet content caching regained relevance over the last years due to the extended use of data center infrastructures in CDNs, clouds and ISP networks to meet the capacity and delay demands of multimedia services. In this study, we evaluate the performance of web caching strategies in terms of the achievable hit rate for realistic scenarios of large user populations. We focus on a class of score gated least recently used (SG-LRU) strategies which combine the simple update effort of the LRU policy with the flexibility to keep the most important content in the cache according to a predefined score function.**

Caching efficiency is evaluated via simulations assuming Zipf request pattern, which have been confirmed manifold in the access to popular web platforms for video streaming and other types of content. We analyze the possible hit rate gain of alternative web caching strategies over pure LRU for the standard independent request model (IRM) within the complete relevant range of the three basic system parameters. The results confirm that the absolute hit rate gains of 10%-20% over LRU being observed in some case studies for special caching strategies are a realistic estimation in general.

Moreover, we compare IRM evaluations with results for dynamic request pattern over time using Wikipedia statistics, which recently have been made available as daily top-1000 page requests. Simulations are extended to show the impact of varying object popularity on the caching efficiency and to adapt a score-based caching strategy to increasing popularity dynamics.

***Keywords* — Web caching strategies, Zipf distributed requests, least recently used (LRU), score gated LRU, least frequently used (LFU), hit rate, simulation, Che approximation**

I. INTRODUCTION

A. *Web caching strategies for content delivery on the Internet*

Web caching systems are widely applied at global scale on the Internet to improve delivery of streaming, IPTV and many other IP services. Today, a major portion of IP traffic is transferred from content delivery networks (CDNs) and distributed data centers in cloud architectures [12][17][22][26], yielding essential traffic savings on expensive intercontinental and inter-domain links. The main benefits are reduced load and delays as well as higher throughput, when caches serve requests to popular data on shorter transport paths to the users. Caches are also present in local networks, as nano data centers [29], or in home gateways and browsers on the end devices [5][11].

In this regard, the efficiency of a cache is driven by the replacement strategy for identifying and storing the most relevant objects. A basic and widely used caching strategy follows the least recently used (LRU) principle, employing a simple cache update scheme. However, LRU can't offer flexibility to prioritize objects according to cost and benefit aspects, regarding the size, the transport path from the origin or other preferences from the content providers' or users' perspective [3].

A number of studies have evaluated LRU web caching efficiency in terms of the hit rate [10][21][28][30]. They partly recommend LRU, whereas on the other hand, alternatives are shown to achieve higher hit rates [3][13][16][18][20][24]. Thus, no clear picture can be concluded from literature whether LRU is appropriate for web caching in a tradeoff with more efficient but often more complex alternatives.

This tradeoff is discussed in detail for the ARC caching strategy proposed by Megiddo and Modha [20]. ARC is shown to outperform LRU by more than 1.5-fold hit rate in most of a large set of measured and synthetic traces. However, the update effort for ARC is high for maintaining two lists and checking half a dozen cases for each update as depicted in Fig. 1 in [20]. Consequently, the ARC method isn't widely adopted.

B. *Score gated LRU (SG-LRU) web caching strategy*

For cache optimization, we recently proposed a score-gated LRU (SG-LRU) approach [14], which combines an LRU stack implementation with arbitrary scores being attributed to each object. SG-LRU can even undercut the update effort of pure LRU web caching, when using an appropriate score function.

LRU always puts the requested object on top of the cache, while the bottom object of a full cache is replaced if cache storage is exhausted. SG-LRU admits a new external object to enter a full cache only if it has a higher score than the bottom object and otherwise puts the bottom object on top, instead of the requested object. Figure 1 illustrates (SG-)LRU updates for a usual implementation as a cyclic double linked list. If the score updates are simple, e.g., when only the score of the requested object is changing, SG-LRU requires slightly higher update effort in the caching list than pure LRU. However, LRU is loading an object into the web cache for each request to an external object, i.e. for each cache miss, whereas SG-LRU can avoid most uploads for seldom requested, low-scored objects. Thus, SG-LRU usually undercuts pure LRU update effort.

The detailed evaluation of the performance gain in SG-LRU hit rates as compared to pure LRU is a main goal of this study. We extend and complement experience from many case studies [3][13][16][18][20][24] based on measurement of specific web applications by an exhaustive simulative evaluation of the tradeoff between LRU and the least frequently used (LFU) principle for independent Zipf distributed requests. Zipf distri-

978-0-9883045-1-2/16 $31.00 © 2016 ITC

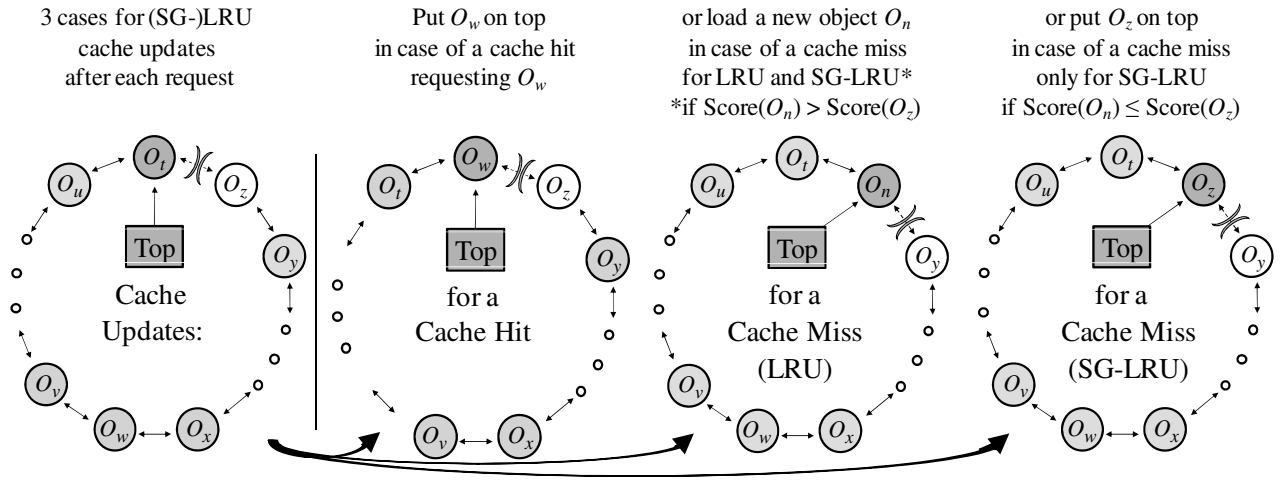

Figure 1: Updates for LRU and SG-LRU caches per request in a usual double linked cyclic list implementation

buted requests have been confirmed as the prevalent access pattern on Internet platforms [1][4][16][25][33] whenever a large user community has access to a large set of objects.

The LFU principle keeps those objects in the cache that have been most frequently requested in the past. Pure LFU is not a valid web caching strategy because it cannot adapt to a changing popularity of objects, but it achieves an optimum hit rate for the independent request model (IRM) and, thus, provides an upper bound of the caching efficiency. As the main new contributions of this study, we perform

- an extensive simulation study of the efficiency of SG-LRU caching strategies including advanced accuracy control estimation developed in [15], which is also used to check the Che approximation [6] for LRU hit rates,
- an exhaustive evaluation of the hit rate optimization potential left open by LRU compared to SG-LRU and LFU for the standard IRM model with Zipf distributed requests, and
- an extension of the study for dynamically changing popularity of objects applied to request pattern for Wikipedia pages.

II. Scenario Classification

Next, we address several preconditions in order to clarify our main focus within the broad spectrum of caching applications.

A. Web caching vs. caches in compuing & database systems

Caching for support of IP services has basic communalities with caching in computer and database systems, i.e. to get faster access to a part of the data that fits into a cache of limited size, but the workloads and the effects on costs and benefits are entirely different for both cases [2][3]. In computing and database systems, periodic sequences of requests are relevant whereas random and Zipf distributed request sequences are usual for web access. Frequent LRU data uploads from an original or higher layer web server to a cache are much more expensive than for data in a local system. Therefore, results from this study on web caching strategies cannot be transferred to caches in computing systems, although some technical discussions seem to mix up the different scenarios [21].

B. Cachable web content and HTTP caching guidelines

Caching is known to be inapplicable for a part of the web content, including highly dynamic content or one-timers [2], i.e. web pages that are only requested once over a long period. Moreover, a content owner or provider can mark objects as non-cacheable in network or in end system caches. Nonetheless, the major portion of IP traffic for video streaming and IPTV services is distributed from caches in CDN and cloud architectures. A recent update of caching options for HTTP [8] by the IETF standardization provides guidelines on how to avoid inappropriate data in caches. We always refer to the fraction of cacheable data in evaluations of caching efficiency.

C. Cache updates per request versus daily updates

Web caching strategies basically assume updates to be done per request, but they can partly be deferred to low traffic periods to reduce peak hour traffic. Cache providers often combine updates per requests with daily updates [17][23][26]. In this work, we focus on updates per request. The need for fast updates depends on the dynamics in the popularity of objects, as discussed in section VI.

D. Caches for large versus small populations

Caching systems are often organized hierarchically with central caches serving a large population and lower level caches for smaller regions. Even a cache in a browser for a single user can save 20% of download traffic for repetitive requests [5]. The request pattern is different for each user and varies for small communities. Our focus is on a large population with access to a large web platform, where the request pattern is known to be universal and characterized by a Zipf law [1].

E. Fixed versus variable size objects

Files and other objects representing cacheable web data have different size up to the GB range for videos. For small caches,

bin-packing problems can arise and therefore size has been considered as an important factor in studies which assume complete files as transport unit. However, coding schemes for video streams or files in client-server and peer-to-peer systems are nowadays segmenting data into small chunks in the kB range, while storage size is steadily increasing. Therefore, we simply assume objects of fixed size corresponding to data chunks. Files of different size can still be represented as sets of fixed size objects with equal score according to the file popularity. Moreover, Hefeeda and Saleh [16] suggest to assign linear decreasing scores to fixed size data chunks of the same video because users often stop video streams after some time, such that the first data chunks of a video are more relevant.

We continue in Section III with an overview of the concept and preconditions for simulative evaluation of (SG-)LRU web caching, a brief overview of a random generator for Zipf distributions (III.A-B), score functions surveying statistics of past requests (III.C) and the control of the precision of hit rate estimates via 2nd order statistics (III.D). Section IV presents hit rate results for the entire relevant parameter range for independent (IRM) Zipf distributed requests. The Che approximation is validated against simulation results in Section V. Section VI extends the performance evaluation for SG-LRU strategies to dynamically changing object popularity with a case study of requests to Wikipedia pages. Finally, main results are summarized in the conclusions.

III. Simulation of Caching for Zipf Request Pattern

A. Relevance of Zipf request pattern

Many studies have confirmed Zipf's law as an appropriate model for access pattern to content on the Internet including web shops and user-generated content such as videos hosted on YouTube [1][30], for channels in IP-TV systems [4][25] and for P2P file sharing systems such as Gnutella and BitTorrent [16][33]. According to Zipf's law, a small set of popular web objects attracts most user requests, which is favourable for the efficiency of small caches.

When a finite set of N objects is considered for web caching, Zipf's law assigns decreasing request probabilities $z(r)$ corresponding to the objects´ popularity ranks $r \in (1, 2, \ldots, N)$:

$$z(r) = \alpha r^{-\beta} \text{ for } \alpha, \beta > 0; \quad \alpha = z(1) = 1/\sum_{r=1}^{N} r^{-\beta} \qquad (1)$$

where β is an adaptive shape parameter and α is a normalization constant. Access probabilities are becoming more unbalanced for $\beta \to 1$. β has been determined for Zipf models that were adapted to different sets of request measurement traces in [1][6][16] resulting in the range $0.56 \le \beta \le 0.88$ covering all studied cases. Therefore we focus our caching simulations on Zipf distributed requests in the range $0.5 \le \beta \le 1$.

B. Inversion method for a random Zipf rank generator

Despite of Zipf's law relevance in Internet access, efficient random generators for Zipf ranks are missing in literature. The Mathematica tool set [32] refers to an acceptance-rejection method for Zipf random variates proposed by Devroye [7] which only covers the range $\beta > 1$ for infinite support. Instead, we derived the following inversion formula for selecting a Zipf rank r from a uniform random variate $R \in [0, 1]$ for finite sets of N objects [15]

$$r = N\Big[1 - \frac{(1-R)(1-(1/2)^{1-\beta})}{1 - Z_{CDF}(N/2)}\Big]^{\frac{1}{1-\beta}}; \; Z_{CDF}(k) = \sum_{n=1}^{\lceil k \rceil} z(n) \qquad (2)$$

In particular, the Zipf rank generator (2) is confirmed in [15] to return the correct rank or a neighbor rank, i.e. to deviate by no more than ±1 from the correct rank r for $N \le 10^6$ and $\beta = 0.1, 0.2, \ldots, 3.0$. The correctness of the rank r is verified by checking $Z_{CDF}(r-1) < R \le Z_{CDF}(r)$. The cumulative Zipf distribution $Z_{CDF}(r)$ is computed and stored for $r \in \{1, \ldots, N\}$ in the starting phase of a simulation. A fast random Zipf rank generator is a prerequisite for simulating billions of requests.

C. Score functions for SG-LRU web caching

In our web caching simulations, we perform cache updates for LRU and SG-LRU strategies, whereas uploading and delivery of objects is not included. Basic data structures are set up for a cache of size $M < N$ and a fixed set of N objects. The cache is implemented as a double chained cyclic list shown in Figure 1 with a top pointer to enable updates at low constant effort per request. Moreover, for each object we store $Z_{CDF}(r)$ according to eq. (1) in order to control the Zipf rank generator as well as a score value in case of SG-LRU.

In general, SG-LRU opens flexibility to prefer cache content according to any arbitrary score function. Usual demands for fast updates impose the restriction of only a few score modifications per request. Scores which represent the number of requests to each object following the LFU principle are simply obtained by incrementing the score of the requested object.

Since pure LFU can't adapt to changing popularity, other approaches with limited count statistics have been proposed and evaluated in literature [6][14][20]. A class of such strategies is attributed as LRFU spectrum [18], which includes LRU and LFU caching schemes as extreme cases. Two basic schemes covering the LRFU spectrum are

- sliding window, restricting LFU to request counts within a window of the W most recent requests, and
- geometrical fading, introducing a decreasing factor ρ^k for the k^{th} recent request and ranking an item according to the sum of weights, as specified in equation (3).

Let $\delta_{Oj}(k) = 1$, if the k^{th} recent request was addressing an object O_j of the set $O_1, \ldots, O_N$ and otherwise $\delta_{Oj}(k) = 0$, where $k = 1$ refers to the most recent request. Then we can define the score functions for sliding window S^{SW} and geometrical fading S^{GF}

$$S_{O_j}^{SW} = \sum_{k=1}^{W} \delta_{O_j}(k); \quad S_{O_j}^{GF} = \sum_{k} \delta_{O_j}(k)\rho^k; \quad 0 < \rho \le 1. \qquad (3)$$

Both schemes behave similar for $\rho = 1 - 1/W$. An unlimited LFU scheme is approached as one extreme for $W \to \infty$, $\rho \to 1$. On the other hand, geometric fading is equivalent to LRU for $\rho \le 0.5$, when the most recent request dominates the weights.

Two score updates are needed for sliding window per new request, where the score of the requested object is incremented and the score of another object, whose former request is falling out of the window, is decremented. Therefore, the sequence of the last W requested objects has to be stored.

For geometrical fading, a direct implementation of the score function S^{GF} requires an update of all objects by a factor ρ per request and an increment of the score of the requested object. Instead, we add $(1/\rho)^l$ only to the score of the l^{th} requested object. In this way, we achieve the same ratio of scores of different objects as for direct computation of S^{GF}. The only drawback is that $(1/\rho)^l$ is steadily growing. Therefore, we have to scale down all scores if the updated score exceed a threshold. Nonetheless, we prefer geometrical fading with fading factor $\rho = 1 - 1/W$ in the evaluations instead of sliding window, because geometrical fading has smaller mean update effort per request and sliding window often needs to resolve equal scores of objects by a tie breaking mechanism.

The usual approach for score based caching maintains a sorted list of objects in the cache according to their scores, which leads to prohibitively high update effort for reinserting objects with modified scores into a sorted list, e.g. via a heap-sort method used in [18]. Instead, we confirm SG-LRU to be sufficient for collecting high scored objects in the cache at low LRU update effort without the need for a strictly sorted list.

D. Evaluation of SG-LRU web caching

Next, we compare the achievable hit rates for LRU, SG-LRU and LFU in simulation studies. We start with a first evaluation example assuming independent and Zipf distributed requests. For caching evaluations of the independent request model (IRM) we consider three basic parameters:

- the size N of a fixed set of objects,
- the cache size M ($M < N$), and
- the shaping parameter β of the Zipf distribution, which determines the request probabilities $z(r) = z(1)\, r^{-\beta}$ to the objects.

We simulate how the cache content is developing for a sequence of K requests, starting from an empty cache. During a filling phase of the cache until M different objects have been addressed, the caching strategies behave identical. As soon as the cache is full, pure LRU already has entered steady state regarding the object set in the cache. Thus, it is sufficient to exclude the cache filling phase as a non-representative start phase for pure LRU simulations. For LFU and SG-LRU, the convergence to a steady state depends on stabilizing score ranks of the objects, which takes much longer than the cache filling phase. LFU scores count the requests to each object. In a sequence of k successive requests, an object in rank r gets a binomially distributed number of requests in [0, ..., k] with mean $k\, z(r)$. We exclude the first quarter of each simulation run from evaluations of the hit rate, in order to converge to stable score ranks when the run time pertains sufficiently long.

Figure 2 shows SG-LRU evaluation results for an example with $N = 10^6$ objects, a cache for $M = 1000$ objects and independent Zipf distributed requests with $z(r) = 0.01337 \cdot r^{-0.8}$. In this case, the LRU hit rate is $h_{\text{LRU}} \approx 10\%$, which is also confirmed by the Che approximation [6], whereas LFU achieves the maximum hit rate under IRM assumptions:

$$h_{\text{LFU}} = z(1) + \ldots + z(M) \approx 20.68\% \qquad (3)$$

The score gated SG-LRU results for different ρ fall between both extremes. For sufficiently long simulation runs with $K > 10^7$ requests, the hit rate is observed to stabilize, where SG-LRU stays close to h_{LRU} for $\rho \le 1-10^{-3}$ and comes close to h_{LFU} for $\rho \ge 1-10^{-6}$. For shorter simulations, the SG-LRU hit rate is still increasing with K towards a saturation level depending on ρ. Since a fading factor ρ has similar effect as sliding window with window size $W = 1/(1-\rho)$, it is obvious that the number of simulated requests is partly smaller than W. Then the scores are still evolving in a transient phase with increasing hit rates. On the other hand, the SG-LRU results in Figure 2 are close to their saturated maximum level as soon as $K > 10 \cdot W = 10/(1-\rho)$ requests are evaluated.

The results confirm that SG-LRU with scores based on previous requests limited by sliding window or geometrical fading can fully adapt to LRU as well as LFU hit rates as extreme cases based on a single parameter, ρ or W respectively. The parameter can be automatically tuned to approach the LFU hit rate up to $h_{\text{LFU}} - \varepsilon$ in the IRM case, because the SG-LRU hit rate is monotonously increasing with ρ and W.

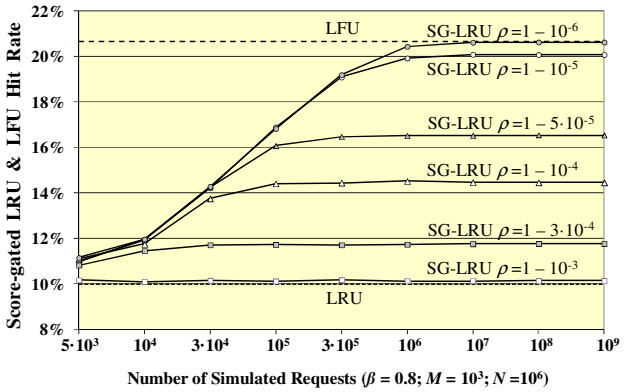

Figure 2: SG-LRU hit rate simulations for different run times

E. Hit rate estimator and variance of the simulation results

The evaluation of the hit rate from simulations is subject to variability that can be indicated by confidence intervals or by the standard deviation based on a number of simulation runs. Let $h(k) = 1$ if and only if the k^{th} request is a cache hit. In order to characterize the variability in hit rate simulations, we evaluate the second order statistics $\sigma(h_{(K)})$ that indicates the standard deviation of a stochastic process over request sequences of different length K. $\sigma(h_{(K)})$ is defined and computed from the mean values over K successive requests of the process $h(k)$:

$$h_{(K)}(j) = \sum_{k=(j-1)K+1}^{jK} h(k) \,/\, K \quad \Rightarrow$$

$$\sigma(h_{(K)}) = \sqrt{E(h_{(K)}^2(j)) - \mu^2(h)}; \quad \mu(h) = E(h_{(K)}(j)) = \Pr(h(k) = 1).$$

Note, that the mean $\mu(h)$ equals the hit rate in all time scales K for a process in steady state, whereas $\sigma(h_{(K)})$ is expected to decrease with K, e.g. $\sigma(h_{(K)}) = \sigma(h_{(1)})/\sqrt{K}$ for a process of i.i.d. random values. Note also, that the cache filling phase and a start phase of a quarter of the requests is not included in the evaluations. In order to evaluate $\sigma(h_{(K)})$ during caching simulations, we consider successive request sequences of length $K = 10, 10^2, ..., 10^R$ of a simulation run over 10^{R+1} requests and take the usual estimate of the standard deviation:

$$\sigma(h_{(K)}) = \sqrt{\sum_{j=1}^{10^{R+1}/K} h_{(K)}^2(j) - \mu^2(h)} \Big/ \left(10^{R+1}/K - 1\right);$$

$$\mu(h) = \sum_{k=1}^{10^{R+1}} h(k)/10^{R+1}.$$

For simulations based on the independent request model we can estimate the hit rate in two ways [15]: (i) by counting the hits or (ii) by the sum of the request probabilities of all objects in the cache over the considered request sequence. We denote the standard deviations of both estimators by (i) $\sigma(h_{(K)})$ and (ii) $\sigma(\pi_{(K)})$, respectively. Figure 3 shows a 2nd order analysis of both hit rate estimators during a simulation for SG-LRU caching with geometrical fading scores for independent Zipf requests ($\beta = 0.8$; $N = 1000$ objects; $\rho = 0.9999$). Four different cache sizes ($M = 2$, 13, 87 and 342) are considered, which achieve about 10%, 25%, 50% and 75% SG-LRU hit rate. The four curves for the standard deviation of the hit count $\sigma(h_{(K)})$ are almost linear $\sigma(h_{(K)}) = \sigma(h_{(1)})/\sqrt{K}$. The four standard deviation curves for the sum of cache probabilites $\sigma(\pi_{(K)})$ as hit rate estimator start at a low level $\sigma(\pi_{(1)}) < 0.005$ already for snapshots of a single request, are remaining almost constant over several time scales and finally are also decreasing. On all time scales we observe that $\sigma(\pi_{(K)}) < \sigma(h_{(K)})$ and therefore prefer the sum of probabilities of cached objects as hit rate estimator.

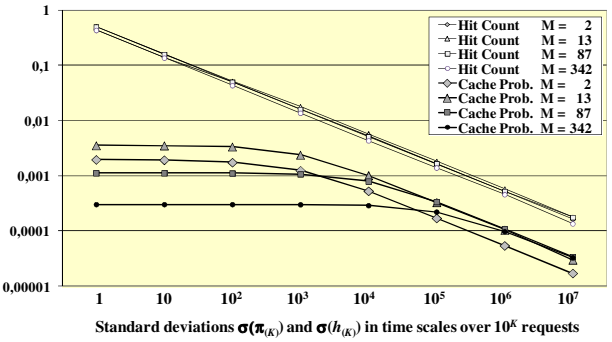

Figure 3: 2nd order statistics for SG-LRU caching simulations

IV. Exhaustive LFU / LRU Hit Rate Evaluations for Zipf Distributed Independent Requests (IRM)

In discussions whether LRU could be sufficient for web caching, the gain of alternative strategies is evaluated in measurement driven case studies [13][16][18][20] but it remains open how the performance depends on system parameters. We complement the experience based on measurement by a generalized simulative evaluation of the gain of LFU over LRU for the IRM model with Zipf distributed requests.

Therefore we extend our web caching simulations over the entire relevant range of the three characteristic parameters: (i) the number of objects N, (ii) the cache size M ($M < N$), and (iii) the shaping parameter β of the Zipf distribution.

Figure 4 shows results from extended evaluations for different size $N = 10^6$ and $N = 10^4$ of the set of objects. The achievable hit rate is compared for LRU and LFU for varying cache sizes M and for $\beta = 0.5, 0.6, ..., 1$, actually with 0.9999 instead of 1. Beyond simulations, the LFU optimum hit rate under IRM equals the sum of request probabilities $h_{\text{LFU}} = z(1) + ... + z(M)$. In case of LRU, the Che approximation can be applied and is confirmed to provide accurate results as discussed in Section V. We also checked that SG-LRU closely approaches the LRU and LFU results as extreme cases.

Regarding the potential advantage of alternatives over LRU for independent Zipf distributed requests, we observe a 10%-15% absolute hit rate gain of LFU over LRU for the entire relevant range, i.e. whenever LRU achieves a hit rate in the range 10% - 50%. The relative gain is largest especially for small caches. When LRU hit rates are below 10%, then the LFU optimum is at least twice as high. Moreover, the caching efficiency can be expressed in terms of the LFU versus LRU cache size required to obtain a demanded hit rate. In order to achieve 20% hit rate, LRU requires 3-fold to 8-fold larger cache size than LFU.

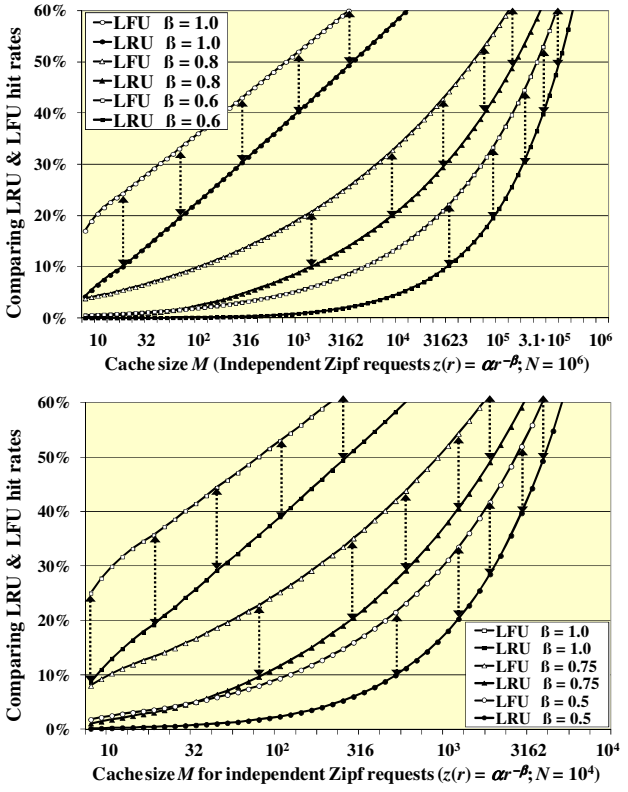

Figure 4: Comparing LFU/LRU hit rates for IRM Zipf requests

Finally, we evaluate the LFU improvement potential beyond LRU for each grid point combining the main parameters:

- $\beta = 0.4, 0.5, ..., 1.1$ for Zipf distributed requests due to IRM,
- $N = 10^2, 10^3, ..., 10^7$ for the number of objects, and
- $M = M_{1\%}, M_{2\%}, ..., M_{99\%}$ for the cache size,

where $M_{x\%}$ is the minimum cache size required to obtain an LRU hit rate of $x\%$ depending on β and N. On the whole, $8 \cdot 6 \cdot 99 = 4752$ simulations have been performed to cover this range. Each simulation runs at least 10^8 requests in the evaluation phase in order to reduce the standard deviation of the hit rate results below $5 \cdot 10^{-5}$ as checked according to Section III.E.

The results of the detailed simulation study are summarized in Figure 5. It is already visible from Figure 4 that the difference between LRU and LFU hit rates is primarily depending on the achieved LRU hit rate. For each combination of β and N we first evaluate the minimum cache sizes $M_{1\%, ..., }M_{99\%}$ to obtain 1%, ..., 99% LRU hit rate and then compute the LFU hit rate for the same cache size. Figure 5 shows the minimum, mean and maximum absolute LFU gain over all cases with the same LRU hit rate of $x\%$.

In fact, we can confirm at least 9.8% potential gain of LFU whenever the LRU hit rate is in the range of 10% - 50%. The mean LFU gain in this range is 13.7% and the maximum goes up to 15% - 20%. The maximum is most often reached for the case $\beta = 1.1$, $N = 100$. Then, the LRU and LFU cache sizes required for $x\%$ hit rate are often unchanged for several values of $x\%$. The minimum gain is due to different cases, most of which are in the largest set of items $N = 10^7$. Again, the relative differences in LFU versus LRU caching efficiency are largest for small caches. When the LRU hit rate is, e.g., 5% then an LFU cache of equal size achieves at least 12.8% hit rate and 15% in the mean over all 8·6 cases of $N \times \beta$ combinations.

The additional storage required to compensate for 10%-20% LRU hit rate deficit ranges from about twice the capacity for $N = 1000$ objects up to 10-fold and more storage required in examples with $N = 10^7$ for large content platforms. Measurement based studies [28] show similar curves for moderate LRU hit rates up to 25%, which indicate even 10 - 100-fold higher storage requirement to obtain 10%-20% more LRU hit rate.

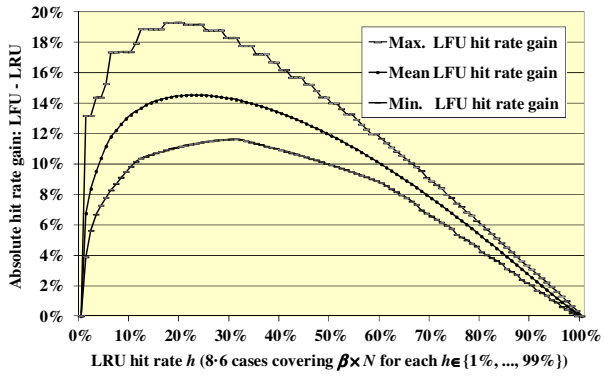

Figure 5: LFU gain over LRU for IRM Zipf requests: Result summary of 4752 cases as a grid on the relevant $\beta \times N \times M$ range

V. Accuracy of the Che Approx. Checked by Simulation

Finally, we compare the Che approximation [6] with simulated LRU hit rates, confirming a surprisingly good match as also experienced in [10]. Note, that high accuracy is supported by mathematical arguments in [10] but without quantitative analysis in terms of concrete boundaries. Thus we checked the deviations between simulation results and the Che approximation for all combinations of parameters in a grid range $(\beta, N, M) \in \{0.4, 0.5, ..., 1.1\} \times \{10^2, 10^3 ..., 10^7\} \times \{M_{1\%}, M_{2\%}, ..., M_{99\%}\}$ for Zipf distributed requests. As the main result, we observe a maximum absolute difference of 0.4% while the majority of evaluated differences is below 0.02%. Each simulation result is again based on 10^8 or more requests and is expected to be subject to a standard deviation of less than $5 \cdot 10^{-5}$ [15]. Partly, longer simulations runs are needed because the precision of the Che approximation is often in the same range. The largest deviations are encountered for small cache sizes M and for $\beta \to 1$.

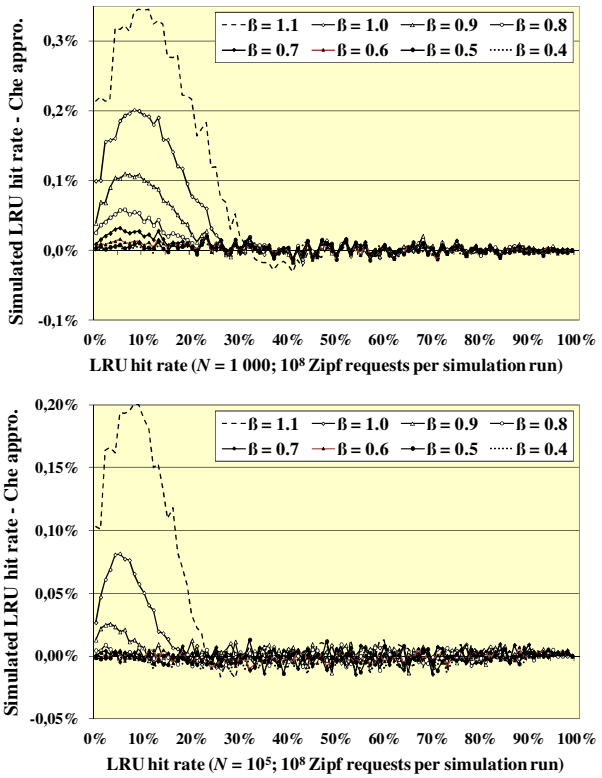

Figure 6: Che approximation: Deviations from simulated results

VI. Evaluation for Dynamic Popularity based on Daily Wikipedia Page Request Statistics

A. Content Popularity Dynamics over Time

In contrast to the IRM assumption, many measurement studies indicate how the popularity of Internet content varies over time

- for user-generated content, e.g., videos [9][19][27][28],
- especially for file sharing systems [33], and
- for IP-TV applications [4][25].

Some interesting conclusions can be drawn from these papers:

- The temporal dynamics in object popularity are noticed on the timescales of days, weeks, or months [4]. A study on the effect of dynamics for caching based on YouTube video traces [28] concludes that request correlations on time scales of a few hours can be ignored. Instead the time scale of a few days/weeks is experienced as most important.
- The popularity evolution of each object is characterized by a rapid growth towards maximum popularity followed by a phase of slow decrease [33]. When the uploaded content is young, significant rank changes are encountered mainly from low initial popularity, stabilizing at the maximum.
- Studies on P2P and IP-TV systems indicate content popularity dynamics to be relatively low. Only 1-3% daily drift in the popularity of the top 100, 1 000 and 10 000 Gnutella files has been observed in [33]. The measurement study [25] reports that the cosine similarity value between the popularity of individual TV channels over 3 days is about 0.97, corresponding to a fairly stable ranking of the content.

In principle, unpredicted changes in content popularity make caching less efficient, if such dynamics is high enough to render content useless only a few requests after being loaded into the cache. The effect of popularity dynamics on cache hits mainly depends on the user population served by the cache and its request frequency, which can count into millions per day.

In fact, we experience higher dynamics in the day by day Wikipedia request pattern than the previously referred studies on video, P2P and IP-TV platforms, see Section VI.D for more details. Nevertheless, Zipf distributed requests and the conclusions on caching efficiency of the previous sections are confirmed to be still relevant.

B. Daily Wikipedia top-1000 statistics & Zipf request pattern

For realistic access pattern of a popular web platform, we refer to statistics being published by Wikipedia [31]. Even if the data volumes are smaller than for popular video streaming platforms, we expect similar dynamic Wikipedia page request pattern. We evaluate daily statistics of the number of requests to the top-1000 pages, which are provided since August 2015.

In a first step, we adapt Zipf distributions to daily top-1000 request distributions. We measure the deviation of the Wikipedia top-1000 requests for the cumulative distribution function $W_{d,\mathrm{CDF}}(k) = R_d(k)/R_d(1000)$ and a Zipf adaptation $Z_{\mathrm{CDF}}(k)$ due to eq. (1-2), where $R_d(k)$ is the sum of requests to the top-k pages (k = 1, ..., 1000) at day d = 1, ..., 184 for the 6 month period from Aug. 2015 - Jan. 2016. Our measure of deviation $\Delta_{Wd\leftrightarrow Z}(k)$ is defined by the difference in the ranks, for which both CDF distributions achieve the same level, i.e. we define

$$\Delta_{Wd\leftrightarrow Z}(k) = j_k - k \quad \text{where} \quad Z_{\mathrm{CDF}}(j_k) \le W_{d,\mathrm{CDF}}(k) < Z_{\mathrm{CDF}}(j_k + 1).$$

Therefore, the Zipf parameter β is determined in order to minimize the mean absolute rank deviation $(\Sigma_k\, |\Delta_{Wd\leftrightarrow Z}(k)|)/1000$. We found the minimum according to the considered deviation criterion always in the range $0.5 < \beta < 0.6$ except for two days in the time frame with larger β up to 0.75.

Figure 7 shows the minimum, maximum and mean absolute rank deviations of Zipf distributions adapted to each daily top-1000 request statistics for optimum β. All rank deviations are in the range [–27, 9] and mean absolute rank deviations are always less than 10. On 4 out of 184 days, the Wikipedia top-1000 requests can be perfectly matched by a Zipf distribution with rank deviations of only –1, 0 or 1 on all 1000 ranks.

The largest deviations from Zipf distributions are experienced in the top-10, whereas the tail of the top-1000 distributions can be closely fitted. This is due to a large statistical variation of requests especially for the top-1 page, ranging from $3 \cdot 10^5$ to $6 \cdot 10^6$ requests, while the total number of daily top-1000 request is less variable between $2.4 \cdot 10^7$-$3.4 \cdot 10^7$. When we consider the distribution given by the mean number of top-k requests over increasing time periods, then we experience a convergence very close to a Zipf distribution.

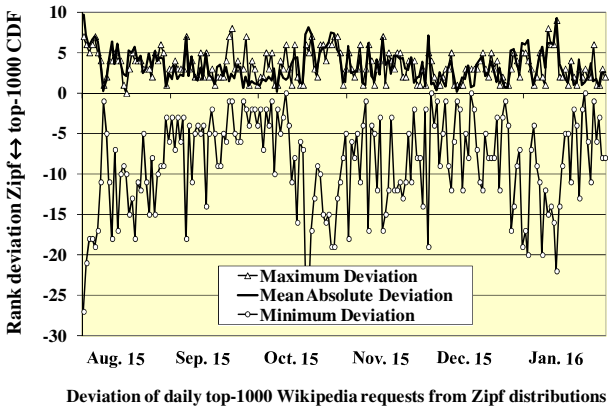

Figure 7: Zipf adaptations to daily Wikipedia page requests

C. Daily Wikipedia top-1000 statistics & IRM cache hit rates

In a first evaluation of caching efficiency based on the Wikipedia data, we assume independent requests per day with request probabilities according to the top-1000 request frequencies and separated simulations for each day.

Figure 8 shows SG-LRU results for several fading factors ρ and for cache sizes 25 and 200, respectively. Although the variability in the daily hit rates is high, from 4.3%-24.9% for LRU with M = 25 and from 29.3%-54.9% with M = 200, the potential gain for SG-LRU is almost constant each day close to the maximum LFU hit rate under IRM conditions. The mean LRU hit rates over 184 days are 8.2% for M = 25 and 36.5% for M = 200, as compared to mean SG-LRU hit rates of 20% (M = 25, ρ = 0.9999) and 49.4% (M = 200, ρ = 0.99999). This confirms the 10%-20% gain similar to the results in Figure 5 also for a real request pattern due to Wikipedia statistics. For those evaluations, all requests are assumed to address top-1000 web pages of a day. With regard to a 10-fold higher number of requests to all Wikipedia web pages, larger cache sizes are required in order to achieve the same hit rates.

D. Cache simulation under daily changing request pattern

From the daily top-1000 page request statistics we can also gain insights into the dynamics of page popularity. The rate of change in the top-k pages is expressed in Table 1 by the fraction of requests addressing top-k pages of the previous day.

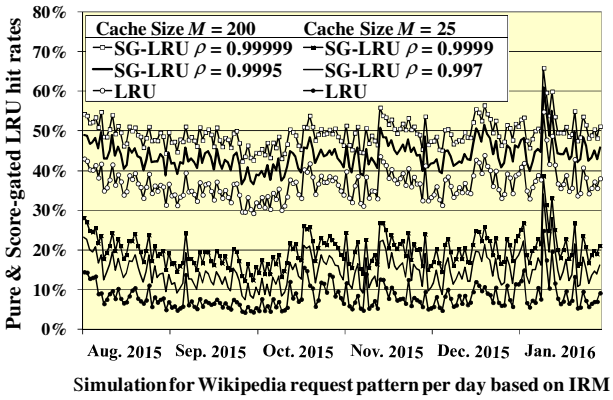

Figure 8: Cache hit evaluation for top-1000 daily requests

The evaluation is again based on 184 days from Aug. 2015 - Jan. 2016. At least more than half of the requests for $k = 25$ and even 75% for $k = 1000$ are referring to yesterday's top-k pages. On the other hand, 20% - 24% of the requests are for new pages which didn't appear among yesterday's top-1000.

Top-k pages: k =	25	50	100	200	500	1000
Y's Top-k → Top-k	54.7%	58.8%	62.4%	66.7%	71.9%	76.1%
1 – (Y's Top-1000 → Top-k)	22.6%	21.8%	21.2%	20.8%	20.9%	23.9%

Table 1: Fraction of requests to yesterday's top-k pages

The fluctuation within the top-200 is captured day by day in Figure 9, with an upper curve for the total number of requests to a steadily changing set of top-200 pages for each day. A dark gray surface represents the requests to the previous day's top-200 and a light gray surface to previous top-1000 pages.

We simulate cache evaluations for the complete time frame shown in Figure 10 based on the number R_d of requests to the top-1000 Wikipedia pages per day ($R_d \approx 2 \cdot 10^7$). A day is simulated with constant request probabilities to those 1000 pages, which are proportional to their daily request count. For a new day, the cache starts with the content from the end of the previous day and it is left to the (SG-)LRU strategy to adapt the cache content to the current request distribution.

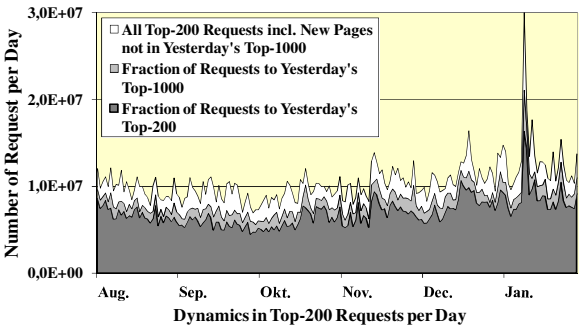

Figure 9: Changes in the daily top-200 request statistics

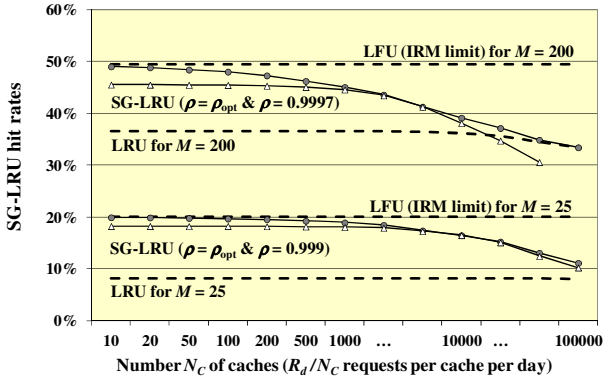

Figure 10: Cache hit rate for daily changing request pattern

Figure 10 shows results of the simulations, again for requests only to the top-1000 Wikipedia pages and for cache sizes of $M = 200$ with hit rate curves in the range $>25\%$ as well as for $M = 25$ with hit rates $<25\%$, respectively. If we assume a single cache for all requests, then the phases adapting to daily change are negligible within the total number $R_d \approx 2 \cdot 10^7$ of top-1000 requests per day, such that the results are equal to the IRM case. Instead, we assume a number N_C of caches, each of which is serving a fraction $1/N_C$ of the user population and thus a fraction $\lceil R_d/N_C \rceil$ of the daily Wikipedia requests. In fact, the considered population and the generated number of requests $\lceil R_d/N_C \rceil$ per cache per day is a main characteristics for the efficiency. When $\lceil R_d/N_C \rceil$ is smaller, then the dynamics due to daily changes is becoming more relevant.

We still apply the SG-LRU caching strategy with geometrical fading score function. The results shown in Figure 10 fall into three different categories depending on N_C:

- In the range $N_C \leq 200$ (for $M = 25$ even up to $N_C \leq 1000$), the LFU hit rate limit for independent requests (IRM) is closely approached by SG-LRU with appropriate fading factor ρ.
- In a range $500 \leq N_C \leq 10\,000$, the SG-LRU hit rate doesn't exploit the LFU limit under IRM conditions, but still significantly improves over pure LRU.
- In a range $N_C \geq 20\,000$, SG-LRU does not essentially improve pure LRU hit rates for geometrical fading scores.

Note that ranges for N_C can also be expressed by the number of requests per cache per day via $\lceil R_d/N_C \rceil \approx 2 \cdot 10^7/N_C$. As a difference compared to the static IRM case, SG-LRU hit rates are now increasing up to an optimum ρ_{opt} and decreasing beyond ρ_{opt}. In the case $N_C = 10\,000$ and $M = 25$, we observe SG-LRU hit rates going up to 16.6% for $\rho \approx 0.999$ and then falling down to 11.4% already for $\rho = 0.9999$. With dynamically changing request pattern, we experience SG-LRU hit rates $h(\rho)$ generally as a function of ρ starting to increase from LRU for $\rho < 0.5$ towards an optimum ρ_{opt} with maximum hit rate h_{max} and then falling for $\rho > \rho_{opt}$ down to hit rates even below the LRU hit rate. This behaviour allows to automatically determine ρ_{opt} and the maximum hit rate. For further study, we are testing more sophisticated and further optimized score functions, trying to predict rising popularity of new pages.

For dynamics on the time scale of hours, the Wikipedia statistics has no data available per page. Therefore we leave an analysis in smaller time scales open for future work based on more detailed alternative traces. If we interpolate a daily change in the requests to a page from R_d to R_{d+1} over the hours as a monotonous trend, then the dynamics is smoother, leading to improved caching efficiency as compared to a hard shift from R_d to R_{d+1}. From our current experience, we agree to [28] that dynamics in the time scale of days/weeks is more relevant.

CONCLUSIONS AND OUTLOOK

We devoted detailed simulation studies to the efficiency of caching for usually observed Zipf distributed request pattern. The least recently used (LRU) caching strategy is compared to score gated SG-LRU methods, combining low LRU update effort with flexible score-based selection of the cache content.

In a first part on the standard model of independent (IRM) Zipf distributed requests, we confirm that LRU hit rates in the range 10-50% generally leave further 10-20% absolute hit rate potential unused compared to SG-LRU, as also shown in several measurement studies for more complex caching strategies [20].

In a second part, we include dynamic popularity changes based on Wikipedia page request statistics, which again exhibit Zipf-like request pattern. Although >20% new pages appear among the top-1000 pages every day, the cache hit rates for Wikipedia pages are still close to IRM conditions for the Zipf-like daily request distribution when a cache serves a large population, i.e. when a cache handles at least 50 000 requests per day. Dynamic request pattern have more impact on the performance of smaller caches, leading to less homogeneous results also depending on local environments and user preferences.

On the whole, the results indicate that caching efficiency is not restricted to the prevalently considered LRU case, but can go up to the essentially higher LFU hit rate limit for Zipf distributed requests. Including statistics about past requests by the proposed SG-LRU method provides a simple extension to exploit the hit rate potential beyond pure LRU for IRM and for more realistic dynamic content popularity scenarios. For future work, we plan to optimize SG-LRU score functions for request pattern based on an extended set of web access measurements.

ACKNOWLEDGEMENTS

This work has received funding from the European Union´s Horizon 2020 research and innovation programme 2014-2018 under grant agreement No. 644866 and within the EU COST ACROSS activity 1304. This work reflects only the authors´ views and the European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] L. Breslau et al., Web caching and Zipf-like distributions: Evidence and implications, Proc. IEEE Infocom (1999)

[2] M. Busari, and C. Williamson, ProWGen: A synthetic workload generation tool for simulation evaluation of web proxy caches, Computer Networks 38 (2002) 779-794

[3] P. Cao and S. Irani, Cost-aware WWW proxy caching, Proc. USENIX Symposium, Monterey, CA, USA (1997)

[4] M. Cha et al., Watching TV over an IP network, Proc. 8th SIGCOMM Conf. on Internet Measurement (IMC), Athens, Greece (2008) 71-84

[5] J. Charzinski, Traffic properties, client side cachability and CDN usage of popular web sites, Proc. 15th MMB conference, Essen, Germany, Springer LNCS 5987 (2010) 182-194

[6] H. Che, Y. Tung, and Z. Wang, Hierarchic web caching systems: modeling, design and experimental results, IEEE JSAC 20(7) (2002) 1305-14

[7] L. Devroye, Non-uniform random variate generation, Springer (1986)

[8] R. Fielding, M. Nottingham and J. Reschke, Hypertext transfer protocol HTTP/1.1: Caching, IETF standardization, RFC 7234 (2014)

[9] F. Figueiredo et al., TrendLearner: Early prediction of popularity trends of user generated content (2014) <http://arxiv.org/abs/1402.2351>

[10] C. Fricker, P. Robert and J. Roberts, A versatile and accurate approximation for LRU cache performance, IEEE Proc. 24th International Teletraffic Congress, Kraków, Poland (2012)

[11] R.G. Garoppo et al., The greening potential of content delivery in residential community networks, Computer Networks (2014) 256-267

[12] G. Hasslinger and F. Hartleb, Content delivery and caching from a network provider's perspective, Special Issue on Internet based Content Delivery, Computer Networks 55 (Dec. 2011) 3991-4006

[13] G. Hasslinger and O. Hohlfeld, Efficiency of caches for content distribution on the Internet, Proc. ITC-22, Amsterdam, The Netherlands (2010)

[14] G. Hasslinger, K. Ntougias and F. Hasslinger, A new class of web caching strategies for content delivery, Proc. Networks Symposium, Funchal, Madeira, Portugal (2014)

[15] G. Hasslinger, K. Ntougias and F. Hasslinger, Performance and Precision of Web Caching Simulations Including a Random Generator for Zipf Request Pattern, Proc. 18th MMB Conf., Münster, Germany, Springer LNCS 9629 (2016) 60-76

[16] M. Hefeeda and O. Saleh, Traffic modeling and proportional partial caching for peer-to-peer systems, IEEE/ACM Trans. on Networking 16/6 (2008) 1447-1460

[17] N. Kamiyama et al., ISP-operated CDN, 14th NETWORKS Telecom. Network Strategy & Planning Symposium, Warszawa, Poland (2010)

[18] D. Lee et al., LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies, IEEE Transactions on Computers 50/12 (2001) 1352-1361

[19] H. Li, et al., On popularity prediction of videos shared in online social networks, Proc. 22nd ACM International Conference on Information & Knowledge Management (CIKM), San Francisco, CA, USA (2013)

[20] N. Megiddo and S. Modha, Outperforming LRU with an adaptive replacement cache algorithm, IEEE Computer (Apr. 2004) 4-11

[21] P. Panchekha, Caching in theory and practice, Dropbox TechBlog (2012) <tech.dropbox.com/2012/10/caching-in-theory-and-practice>

[22] M. Pathan, R.K. Sitaraman and D. Robinson, Advanced content delivery, streaming and cloud services, Wiley (2014)

[23] PeerApp Inc. <www.peerapp.com>

[24] S. Podlipnik and L. Böszörmenyi, A survey of web cache replacement strategies, ACM Computer Surveys (2003) 374-398

[25] T. Qiu et al., Modeling channel popularity dynamics in a large IPTV system, Proc. 11th ACM SIGMETRICS, Seattle, WA, USA (2009)

[26] R.K. Sitaraman et al., Overlay networks: An Akamai perspective, Chapter 16 in Advanced content delivery, streaming and cloud services, Wiley (2014) 307-328

[27] G. Szabo, and B.A. Huberman, Predicting the popularity of online content, ACM Communications 53/8 (2010) 80-88

[28] S. Traverso et al., Unravelling the impact of temporal and geographical locality in content caching systems, IEEE Trans. on Multimedia 17 (2015) 1839-1854

[29] C. Valancius et al., Greening the Internet with nano data centers, Proc. ACM CoNEXT Workshop, Rome, Italy (2009)

[30] D. Wessels, Squid: The definitive guide, O'Reilly (2004)

[31] Wikipedia statistics https://wikitech.wikimedia.org/wiki/Pageviews_API, top-1000 request statistics for English Wikipedia pages, wikimedia.org/api/rest_v1/metrics/pageviews/top/en.wikipedia/all-access

[32] Wolfram Research, Wolfram language tutorial (2015) https://reference.wolfram.com/language/tutorial/RandomNumberGeneration.html

[33] S. Zhao, D. Stutzbach and R. Rejaie, Characterizing files in the modern Gnutella network: A measurement study, SPIE/ACM Proc. Multimedia Computing and Networking (2006)