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

Gerhard Hasslinger[a], Konstantinos Ntougias[b], Frank Hasslinger[c] and Oliver Hohlfeld[d]

[a] *Deutsche Telekom, Darmstadt, Germany, Email: gerhard.hasslinger@telekom.de*
[b] *Athens Information Technology, Athens,Greece, Email: kontou@ait.gr*
[c] *Darmstadt University of Technology, Darmstadt, Germany, Email: frank.hasslinger@stud.tu-darmstadt.de*
[d] *RWTH Aachen University, Aachen, Germany, Email: oliver@comsys.rwth-aachen.de*

Keywords
Web caching strategies
Least Recently Used (LRU)
Score Gated (SG-)LRU
Least Frequently Used (LFU)
Caching simulator
Zipf law request pattern
Zipf random variate generator
2nd order statistics control
Cache hit rate
Che approximation

Abstract

The topic of Internet content caching regained relevance over the last years due to the increasing and widespread use of content delivery infrastructures to meet 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 an arbitrarily predefined score function.

Caching efficiency is evaluated via simulations assuming Zipf distributed requests, which have been confirmed manifold in access pattern of popular web platforms for video streaming and various other content types. In this paper, we analyze the hit rate gain of alternative web caching strategies for the standard independent request model (IRM) within the complete relevant range of three basic system parameters. The results confirm absolute hit rate gains of 10%-20% over pure LRU as realistic estimation in general, as experienced in other trace-driven case studies for special caching strategies.

Moreover, we compare the performance for the independent request model with results for correlated dynamic request pattern over time, based on Wikipedia statistics that are available as daily top-1000 page requests. In this way, we show that the IRM analysis is valid for caches with a large user population, although high dynamics tends to reduce the achievable hit rate below the IRM result for smaller user communities.

1. Introduction: Content delivery and web caching

A. Relevance of web caching for growing Internet traffic

Web caching systems are widely deployed at global scale to improve downloads, video streaming, IP-TV and many other IP services. Today, a major portion of the IP traffic is transferred via content delivery networks (CDNs) and data centers in cloud architectures [1][17][22][28][33][36]. Their distributed nature yields substantial 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 [40], in home gateways and browsers on the user devices [8][15].

Caches are essential for increasing the Internet's capacity and enabling scaling to larger user bases. Figure 1 illustrates the growth of IP traffic since the year 2000 as reported by official statistics for Australia, Germany, and Hong Kong [3] [6][26] as well as annual reports by the router manufacturer Cisco [10]. In comparison to the fixed network and total traffic volume, we notice that mobile IP traffic still contributes less than 10%, but is catching up with currently higher growth rates. Consequently, caching in mobile networks is becoming more relevant, where the transmission over the air interface constitutes a main bottleneck. Therefore the usage and optimization of caches on mobile end devices is especially crucial, where limited storage capacity poses the challenge of efficient utilization of small caches. On the whole, caches of different size are present in various parts of IP networks ranging from the core via edge servers of clouds, CDNs and ISP networks to the user devices. In this way, they play an important role to overcome bandwidth bottlenecks and to reduce transfer delays [17][29][32][33].

The annual reports by Cisco [10] and Sandvine [34] include estimations of main IP traffic components, which reveal that video streaming and download applications generate most of the current IP traffic. In principle, this hasn't changed since the time when P2P traffic was dominant [16], which bypassed traditional web caches. In the last decade, the P2P traffic portion was decreasing, whereas the HTTP traffic portion became dominant again [34]. Today, the bulk of the network traffic is being delivered by global CDNs.

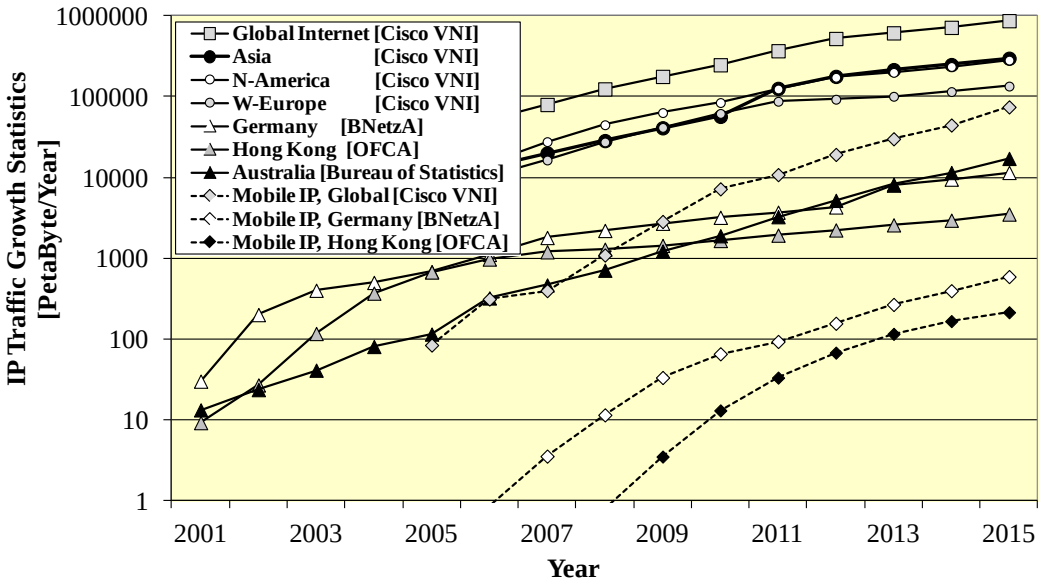

Figure 1: IP traffic growth in fixed and mobile networks

Cisco's annual reports include a global cloud index, which indicates that most IP traffic is flowing from data centers to the users and that there is a considerable amount of cross traffic within and between data centers [10].

B. Web caching strategies and criteria for efficiency

The efficiency of caching systems is driven by the replacement strategy for identifying and storing the most relevant objects. A basic and widely used caching scheme follows the least recently used (LRU) principle, employing simple and fast cache updates. However, LRU can't offer flexibility to prefer objects due to cost and benefit aspects, regarding the size, the transport path from the origin or other preferences from the content providers' or users' perspective [2][33].

A number of studies have evaluated LRU web caching efficiency in terms of the hit rate [14][27][39][41]. They partly recommend LRU, but on the other hand, alternatives are shown to achieve higher hit rates [18][21][23][25][30]. 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 by Megiddo and Modha [25], who propose an Adaptive Replacement Cache (ARC) algorithm for improved efficiency. ARC is shown to outperform LRU by 1.5-fold hit rate for a large set of measured and synthetic traces. However, ARC maintains two caching lists and involves more than half a dozen cases to be checked for an update per request, as depicted in Fig. 1 of the work [25].

From a more general perspective, the following set of basic criteria summarizes discussions on web caching efficiency in the literature regarding the design of caching strategies and their implementation:

A. Simple cache updates:
The implementation should be simple enabling updates at low constant update effort per request to cope with LRU.

B. High hit rate:
The hit rate should be high and should approach the optimum LFU hit rate when the independent request model (IRM) is relevant for the user request pattern.

C. Adaptability to dynamically changing popularity:
The influence (i.e. weight) of past requests on the decision about the cache content should decrease over time to enable response to shifts in popularity and for predictive prefetching methods.

D. Minimum upload traffic to the cache:
The traffic and effort for loading objects into the cache should be as small as possible. In distributed caching architectures the cross traffic between caches should also be minimized.

E. Flexible caching policy:
The caching strategy should be flexible and adaptable to optimize the cache content based on statistics about past requests, prediction methods or other policies considering content placement, transport costs and quality of service.

C. Score gated LRU (SG-LRU) web caching strategy

Since we couldn't find caching methods in the literature fulfilling all listed criteria, we propose score-gated LRU (SG-LRU) [18], which combines an LRU stack implementation with arbitrary scores being attributed to each object. The selection of objects to be put into the cache depends on the scores, which makes the method fully flexible according to the previous criterion (E).

LRU always puts the requested object on top of the cache, while the bottom object is evicted if cache storage is exhausted. As the only difference, SG-LRU admits a new external object to enter a full cache only if it has a higher score than the bottom object. Otherwise, the bottom object is put on top instead of the requested object.

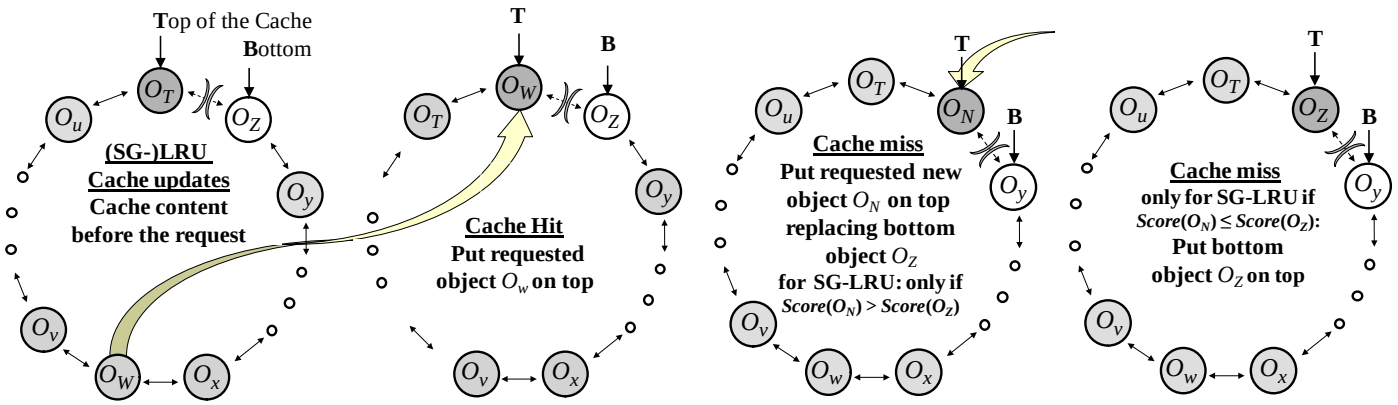

Figure 2: Updates for LRU and SG-LRU caches per request in a usual double linked cyclic list implementation

Figure 2 illustrates (SG-)LRU updates for the usual implementation as a cyclic double linked list. The update effort of SG-LRU on a double linked caching list implementation is almost the same as for LRU. In addition, object scores are updated for SG-LRU. Thus the score function has to be implemented such that updates are no more complex than the updates in the caching list, e.g., when only the score of the requested object or a few others is changed by a simple operation. A request count within a sliding window or with geometrically fading weight as explained in Section 3 can be updated at low constant effort per request.

On the other hand, LRU is loading objects into the web cache at the cache miss rate, i.e. a requested object is always put into the cache if it wasn't found. This leads to more LRU cache loading traffic and processing effort, which can be much higher than updating pointers in the LRU cache list. SG-LRU avoids most of those uploads, because the SG-LRU cache content usually becomes stable, after being filled with the objects of maximum score, provided that the ranking of objects due to scores is stabilizing over time. In this regard, SG-LRU undercuts the LRU update effort in usual web caching scenarios.

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 measurement based studies [21][23][25][30] by an exhaustive simulative evaluation of the tradeoff between LRU and the least frequently used (LFU) principle. Independent Zipf distributed requests are assumed in the first part, which have been confirmed as the prevalent access pattern on Internet platforms [4][9][21][31][37], 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. Even if LFU is not a practically valid web caching strategy because it cannot adapt to a changing popularity of objects, LFU achieves an optimum hit rate for the independent request model (IRM) and thus provides an upper bound of the caching efficiency. As the main contributions of this study, we perform

- a simulation study of the efficiency of SG-LRU caching strategies including advanced accuracy control estimation developed in [19], which is also used to check the Che approximation [9] 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.

We continue clarifying our main focus within a broad spectrum of caching evaluations in Section 2. Section 3 introduces the concept and implementation details of score gated LRU. Simulative performance evaluations are addressed in Section 4 including an efficient random generator for Zipf distributions and the control of the precision of hit rate estimates via 2nd order statistics. Section 5 presents SG-LRU 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 5. Section 6 extends the performance evaluations to dynamically changing object popularity with a case study of requests to Wikipedia pages. Finally, main results are summarized in the conclusions.

The article extends previous work [20] with more details on the implementation of the method and the caching framework.

2. Scenario Classification

Next, we address different preconditions and goals of caching applications to distinguish our main focus on web caching for a large user population from other scenarios.

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

Caching as used by IP-based services shares similarities with caching in computer and database systems, i.e., to provide faster access to a portion of frequently requested data. Internet workloads and their effects on costs and benefits of caches are entirely different from caches in local operation systems [2][5]. In computing and database applications, periodic sequences of requests are usual, whereas Internet workloads are dominated by random and Zipf distributed request sequences. Frequent uploads of data as done by LRU for each cache miss are much more expensive in terms of network traffic load and delay than for data in a local system. Therefore, results from this study on web caching strategies cannot be transferred to caches in computing systems and vice versa, although some technical discussions seem to mix up the different scenarios [27].

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, i.e. web pages that are only requested once over long time periods [5]. 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 IP-TV services is distributed from caches in CDN and cloud architectures. A recent update of caching options for HTTP by the IETF standardization provides guidelines on how to avoid inappropriate data in caches [12]. 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 performed on a per request basis, but they can partly be deferred to low traffic periods to reduce peak hour traffic. Cache providers often combine updates per request with daily updates [22][29][32][36]. 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 6.

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 caches in the browser of each single user can save around 20% of download traffic for repetitive requests [8]. The request pattern is different for each user and is also varying when a cache serves small communities. Our focus is on a large population with access to large web platforms, where the request pattern is known to be universal and characterized by a Zipf law [4].

E. Fixed versus variable size objects

Files and other objects representing cacheable web data have different sizes of up to several GByte in case of 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 files and video streams are nowadays segmenting data into small chunks in the kByte 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. From a more detailed perspective, Hefeeda and Saleh [21] suggest assigning 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.

3. Implementing score-gated LRU

The implementation of the basic (SG-)LRU cache as a double chained cyclic list is already shown in Figure 2 with a top and bottom pointer to enable updates at low effort per request.

In addition, a score value for each object has to be stored and pointers between the objects in the catalogue and their position in the cache are established as shown in Figure 3.

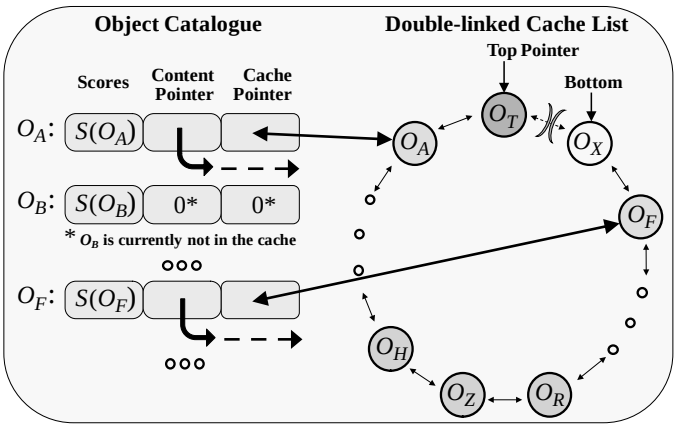

Figure 3: Data structure for simulation of SG-LRU caching

A count of the requests to each object can be achieved simply by incrementing the score of the requested object, corresponding to the least frequently used (LFU) policy. Since pure LFU can't adapt to changing popularity, approaches for restricted count statistics have been proposed and evaluated in literature [9][18][25]. A class of such strategies is attributed as LRFU spectrum [23][30], which includes LRU and LFU caching schemes as extreme cases. Two basic methods covering LRU and LFU in this way are

- Sliding Window (SW): An extension of LFU, which limits the request count per object within a window of the W most recent requests, and
- Geometrical Fading (GF): A decreasing weight factor ρ^k for the k^{th} recent request is introduced and the ranking is done according to the sum of weights per object for all requests in the past.

The survey [30] refers to a similar ρ-aging method, where the scores of all objects are reduced by a factor ρ not per request, but in longer time intervals, e.g. once an hour, thus reducing the effort per request.

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

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

Figure 4 illustrates how the object's position in the cache is determined for (SG-)LRU with the score-based Sliding Window and Geometric Fading schemes.

Both schemes behave similar when the window size equals the sum of weights: $W = 1/(1 - \rho)$. 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.

Regarding the web caching criteria (A-E) of section 1.A, it is obvious that Sliding Window and Geometric Fading are adaptive to changing popularity of objects according to (C).

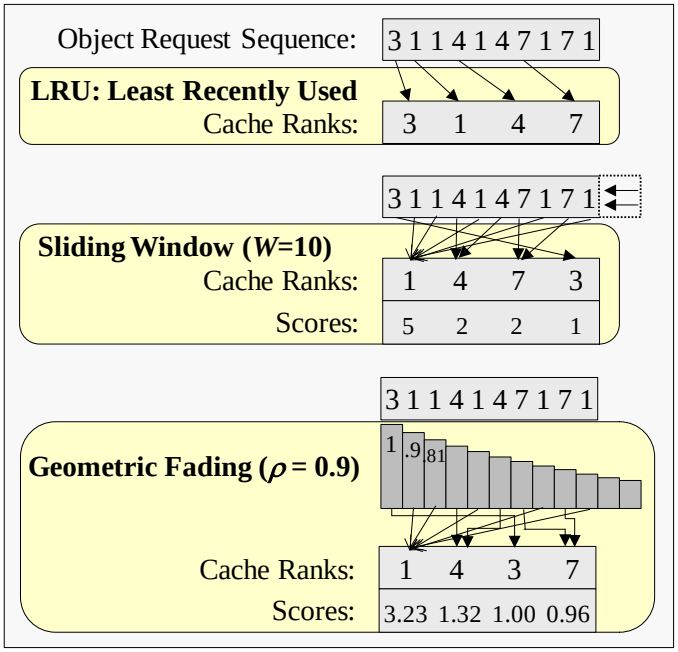

Figure 4: Comparison of LRU, Sliding Window, Geom. Fading

The parameters W and ρ determine the size of the involved memory for past requests, where a compromise has to be found in a trade-off with criterion (B), which is approached for large values $W \to \infty$, $\rho \to 1$.

We take a closer look at the implementation of sliding window and geometrical fading with regard to criterion (A) for fast updates per request.

For sliding window, we store the sequence of the W most recent requests r_{k-W+1} , …, r_k. Then a score update per request modifies the score of two objects: The score is incremented for the new requested object r_k and decremented for object r_{k-W+1}, whose request is falling out of the window and thus is overwritten by r_k. Those updates require low constant effort per request, but extra storage of the order $O(W)$ is needed to hold r_{k-W+1} , …, r_k. This puts a scalability restriction on W, which can be strict for user devices with small caches. On data center cache installations with TByte storage for the content, even a few GByte of extra storage for $W = 10^9$ seems negligible.

A direct implementation of the score function S^{GF} for geometrical fading would require an update of all objects by a factor ρ per request and an increment of the score of the requested object. Instead, as the update for the l^{th} request, we leave the weights for past requests unchanged and, add $(1/\rho)^l$ only to the score of requested object. In this way, we achieve the same ratio of weights for past requests as defined for S^{GF} in equation (1). The only drawback is that $(1/\rho)^l$ is steadily growing. If the updated score exceeds a threshold, we have to scale down the scores. Nonetheless, we prefer geometrical fading in the evaluations rather than sliding window, because

- mean update effort per request is lower for geometric fading,
- sliding window often needs to resolve equal scores of objects by a tie breaking decision.

For geometrical fading, equal scores are extremely seldom and thus have negligible effect. It can be proven that geometric fading scores of two objects, which have been requested at least once, are different, if ρ is a rational number, e.g., $\rho = 1/2$ or $\rho = 0.9999$. Nevertheless, the machine number format may assign the same score value, if the difference of two scores is below the significant precision threshold.

Finally, we compare the SG-LRU method to the direct approach for score based caching, which maintains a sorted list of objects in the cache according to their scores. The latter requires logarithmic $O(\log(N))$ update effort for reinserting an object with modified score into a sorted list of N objects, e.g. via the heap-sort algorithm used in [23]. Other work on optimized cache policies attributes such update effort as "prohibitively high" and instead demands and proposes the ARC variant with constant update effort per request [25]. An implementation of LFU, with constant update effort per request is provided by [35], i.e. for objects being sorted by the number of past requests. SG-LRU avoids many case specific actions of the double LRU cache architecture of ARC. For geometrical fading scores, the only SG-LRU effort beyond pure LRU is for the l^{th} request:

- multiplying the weight factor $(1/\rho)^{l-1} \cdot (1/\rho) \to (1/\rho)^l$ and
- adding $(1/\rho)^l$ to the score of the requested object.

The SG-LRU mechanism doesn't strictly enforce to keep the highest scored objects in the cache as in a sorted list, but the probability to enter the cache is increasing with the score of an object and the number of request to an object. We compared both alternatives in a previous study for independent Zipf distributed requests [18] without finding significant differences in caching performance. Once all top-k objects have entered the cache, they will stay in the cache if the score ranking is stable over time. Therefore the SG-LRU cache content converges to the set of highest scored objects.

4. Simulation of Caching for Zipf Request Pattern

There are only few special analytical results known on cache performance, e.g. the hit rate for LFU in the independent request model (IRM) given in eq. (4), as well as some approximations, e.g. the Che approximation for the LRU hit rate. Therefore cache evaluations are mainly based on simulations and measurement based studies. 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 fixed size $M < N$ and a catalogue of N objects.

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 [4][38], for channels in IP-TV systems [7][31] or for peer-to-peer file sharing systems (BitTorrent, Gnutella) [21][37]. According to Zipf's law, a small fraction 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 $r \in (1, 2, \ldots, N)$:

$$z(r) = \alpha\, r^{-\beta} \text{ for } \alpha, \beta > 0; \quad \alpha = z(1) = 1/\sum_{r=1}^{N} r^{-\beta} \qquad (2)$$

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 [4][9][21]. The range $0.56 \le \beta \le 0.88$ is covering all cases in those studies. 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 the relevance of Zipf's law for Internet access, efficient random generators for Zipf ranks seem to be missing in literature. The Mathematica tool set [43] refers to an acceptance-rejection method for Zipf random variates proposed by Devroye [11], which only covers the range $\beta > 1$ for infinite support. 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 [19]

$$r = N\Big[1 - \frac{(1-R)(1-(1/2)^{1-\beta})}{1 - Z_{CDF}(N/2)}\Big]^{\frac{1}{1-\beta}}; \quad Z_{CDF}(k) = \sum_{n=1}^{\lceil k \rceil} z(n). \qquad (3)$$

In particular, the Zipf rank generator (3) is confirmed in [19] to return the correct rank or a neighbor rank, i.e. to deviate by no more than ±1 from the correct rank r for all $N \le 10^6$ 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 in order to control the Zipf rank generator. A fast random Zipf rank generator is a prerequisite for evaluating billions of requests in the simulations.

C. *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 set of cached objects. 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. During 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 the hit rate evaluations in order to converge to stable score ranks when the run time pertains sufficiently long.

Figure 5 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 $h_{LRU} \approx 10\%$, which is also confirmed by the Che approximation [9], whereas LFU achieves the maximum achievable hit rate under IRM assumptions:

$$h_{LFU} = z(1) + \ldots + z(M) \approx 20.68\% \qquad (4)$$

The SG-LRU results for different fading factor 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 $\rho \ge 1-10^{-6}$. A fading factor ρ has similar effect as a sliding window of the size $W = 1/(1-\rho)$. Therefore it is obvious that up to 10^6 past requests have significant influence on the scores and that the SG-LRU hit rates are increasing with K towards a saturation level. The results in Figure 5 demonstrate a transient phase of increasing hit rate in each curve, which finally reaches and stays at a maximum level after $K > 10 \cdot W = 10/(1-\rho)$ requests have been evaluated.

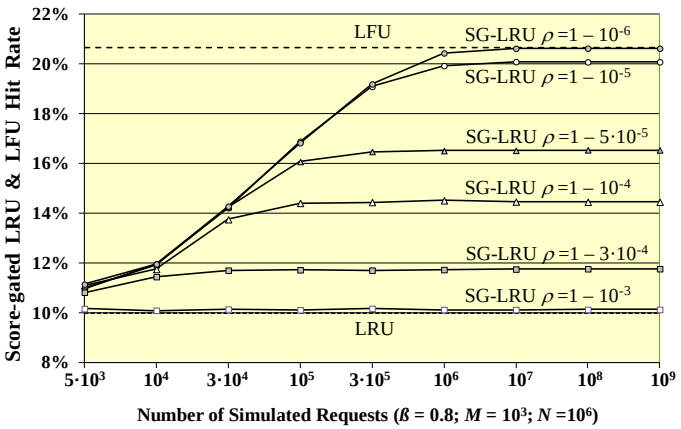

Figure 5: SG-LRU hit rate developing with simulation run time

The results confirm that SG-LRU with scores based on previous requests and backlog limitation 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 $h_{LFU} - \varepsilon$ in the IRM case, because the SG-LRU hit rate is monotonously growing with ρ or W.

D. *Control of the accuracy of hit rate simulation results using 2nd order statistics*

The simulative evaluations of cache hit rates are subject to variability that can be estimated either by confidence intervals

or by the 2nd order statistics of the simulation results [19]. Let $h(k) = 1$ if the k^{th} request is a cache hit, or otherwise $h(k) = 0$. We evaluate the second order statistics $\sigma(h_{(K)})$ as the standard deviation of a stochastic process for evaluations over request sequences of different length K, and thus for simulations of different run time. $\sigma(h_{(K)})$ 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 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. The simulation of LRU caches already enters a steady state as soon as the cache is filled, whereas SG-LRU caching behavior depends on the scores, which often require a long time span to establish stabilized ranking of objects [19].

We evaluate $\sigma(h_{(K)})$ during caching simulations based on 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 for the standard deviation:

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

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

The cache filling phase and a start phase of the first quarter of simulated requests are always excluded from the evaluation in order to avoid an impact of initial transient phases.

When we simulate independent requests (IRM), there are two alternatives to estimate the hit rate:

- by counting the hits $h(k)$ or
- by computing the cache content request probability $\pi(k)$ as the sum of request probabilities of all objects that are currently in the cache

$$\pi(k) = \Sigma_{\,j:\, Oj \text{ is in the cache after } k \text{ requests}}\, z(j)$$

where $z(j)$ is the request probability of object Oj. Again, we denote the standard deviations of the mean over sequences of K requests by $\sigma(\pi_{(K)})$.

Figure 6 shows excerpts $\pi(K+1), ..., \pi(K+1000)$ of the stochastic process $\pi(k)$ for an LRU caching simulation with Zipf distributed requests ($\beta = 0.8$; $N = 1000$ objects) in steady state IRM conditions when the cache is full. Four cases of different cache sizes $M = 13$, 47, 182 and 469 are considered, which are sufficient to achieve 10%, 25%, 50% and 75% cache hit rate.

Figure 7 shows a 2nd order analysis of a SG-LRU caching simulation with geometrical fading score function S^{GF} defined in equation (1) for the same scenario of independent Zipf requests as considered in Figure 6 ($\beta = 0.8$; $N = 1000$ objects; $\rho = 0.9999$). SG-LRU requires smaller cache sizes $M = 2$, 13, 87 and 342 to achieve 10%, 25%, 50% and 75% SG-LRU hit rate. 2nd order statistics are shown:

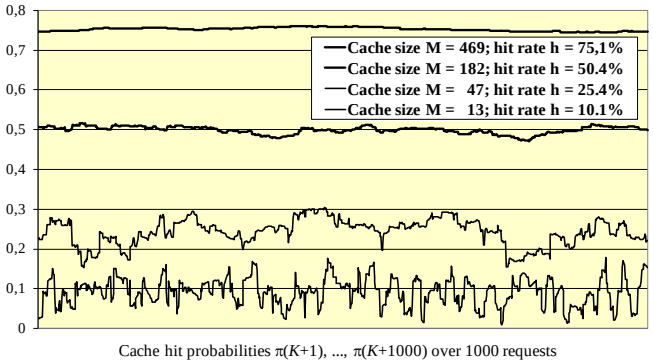

Figure 6: Hit probability $\pi(k)$ varying with the cache content

- for the hit count $\sigma(h_{(K)})$ and
- for the cache content request probability $\sigma(\pi_{(K)})$

with four curves for the different cache sizes in each case.

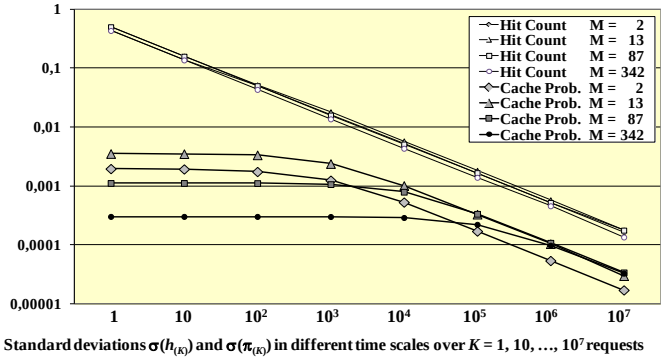

Figure 7: 2nd order statistics for SG-LRU caching simulations

The four curves for the hit count $\sigma(h_{(K)})$ are almost linear $\sigma(h_{(K)}) \approx \sigma(h_{(1)})/\sqrt{K}$, although cache hits are not an i.i.d. process. The four standard deviation curves for the cache content request probability $\sigma(\pi_{(K)})$ start at a low level $\sigma(\pi_{(1)}) < 0.005$ already for single request, are remaining almost constant over several time scales and finally also approach a linear decreasing slope. On all time scales we observe $\sigma(\pi_{(K)}) < \sigma(h_{(K)})$ and therefore prefer the cache content probability as hit rate estimator. The exhaustive comparison of pure LRU and SG-LRU hit rates in Section 5 involves the control by 2nd order statistics, confirming that 10^8 simulated requests are usually sufficient to bring the standard deviation of the hit rate estimator into the range of $10^{-4} - 10^{-5}$. Otherwise, in a few cases the 2nd order control indicates extended simulations to be required with up to 10^{10} requests.

5. Comparative (SG-)LRU and LFU Evaluation

A. *(SG-)LRU and LFU hit rate evaluation on the entire parameter range for independent Zipf distributed requests*

In several measurement driven case studies [21][23][25] alternative strategies are compared to LRU, but it remains open how the evaluated performance gains depend on the system parameters. We complement their measurement based experience by a

comprehensive evaluation of the gain of LFU compared to LRU for Zipf distributed independent requests (IRM). 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 shape parameter β of the Zipf distribution.

Figure 8 shows results from evaluations for $N = 10^6$ objects in the first graph and for $N = 10^4$ in the second. The hit rates for LRU and LFU are compared for varying cache sizes M and for $\beta = 0.5, 0.6, ..., 1$, actually 0.9999 instead of 1. The LFU optimum hit rate under IRM is given by the top-M request probabilities in eq. (4). In case of LRU, the Che approximation [9] can be applied, as studied in detail in Section 5.C. Nonetheless, we use SG-LRU simulations for cache hit rate evaluation in all cases and refer to eq. (4) and [9] only for an additional check.

Overall, we observe a 10% - 20% absolute hit rate gain of LFU over LRU for the entire relevant range, i.e., when LRU achieves hit rates between 10% and 50%. The relative LFU gain is large especially for small caches. When LRU hit rates are below 10%, then the LFU optimum is at least twice as high. Figure 8 also expresses efficiency of the caching strategies in terms of the cache sizes required to obtain a desired hit rate. In order to obtain 50% hit rate, LRU requires between 1.6-fold and 10-fold larger cache size than LFU; LRU caches for obtaining 10% hit rate are at least 10-fold, partly 100-fold larger.

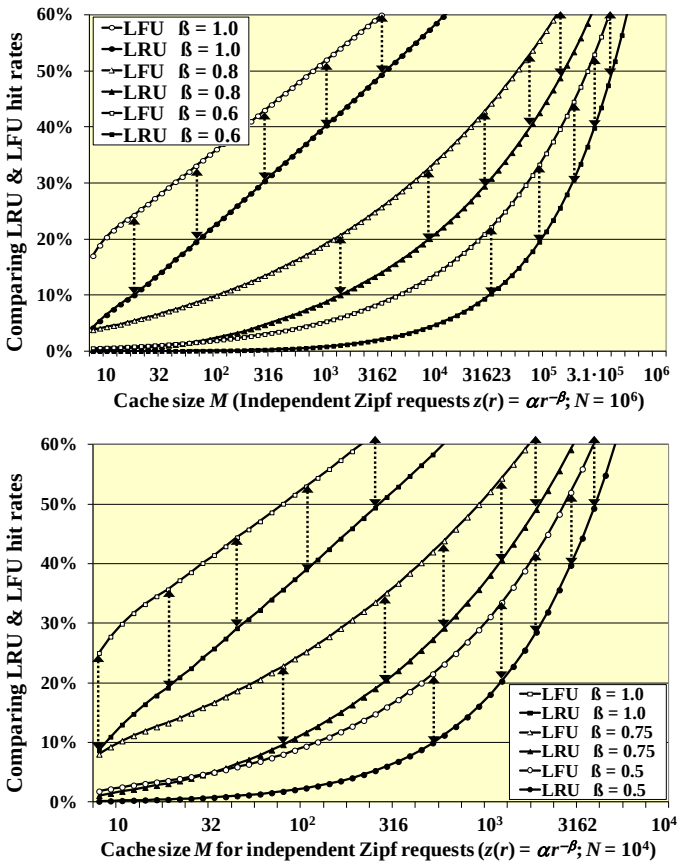

Figure 8: Comparing LFU/LRU hit rates for IRM Zipf requests

B. *Summary of the gain of LFU and SG-LRU performance over LRU for Zipf distributed IRM requests*

Finally, we evaluate the potential for improving LRU on a grid of 8·6·99 cases combining the three main parameters over their entire relevant range for

- $\beta = 0.4, 0.5, ..., 1.1$ to determine the Zipf distribution,
- $N = 10^2, 10^3, ..., 10^7$ 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. In total, $8 \cdot 6 \cdot 99 = 4752$ simulations have been performed to cover the grid range. Each simulation runs at least 10^8 requests in the evaluation phase to keep the standard deviation of the hit rate results below $5 \cdot 10^{-5}$ as checked according to Section 4.D.

The results of the detailed simulation study are summarized in Figure 9. For each combination of β and N we first evaluate the minimum cache sizes $M_{1\%}, ..., M_{99\%}$ to obtain LRU hit rates of 1%, ..., 99% and then evaluate SG-LRU and LFU hit rates for the same cache size. Figure 9 shows the minimum, mean and maximum absolute LFU hit rate gain for all 8·6 combinations of β x N each time with a cache of the same size $M_{X\%}$ that is sufficient to achieve X% LRU hit rate.

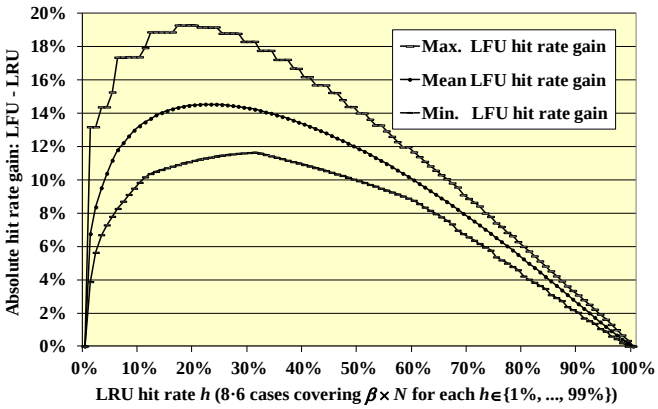

Figure 9: LFU gain over LRU for IRM Zipf requests: Result summary of 4752 cases on a grid on the relevant $\beta \times N \times M$ range

In fact, we can confirm at least 9.8% absolute hit rate gain whenever the LRU hit rate is in the range of 10% - 50%. In this range, the maximum gain is between 15% - 20% and the mean gain over all cases is 13.7%. 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 content catalogue $N = 10^7$. 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% and thus more than 2.5-fold hit rate.

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 [38] 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.

Finally, it isn't difficult to construct scenarios with larger deficits of the LRU cache hit rate. When we consider a cache of size M with IRM request pattern and probabilities

$$p_1 = p_2 = \ldots = p_M = 0.5/M >> p_{M+1} = p_{M+2} = \ldots = p_N = 0.5/(N\text{-}M)$$

then LFU achieves 50% hit rate by keeping O_1, ..., O_M in the cache, whereas the LRU hit rate is less than 25% for $N >> M$, because in the mean less than half of the objects O_1, ..., O_M are found in an LRU cache. For $p_1 = p_2 = \ldots = p_M \approx 0.58/M$ we observed a 28.9% absolute LFU hit rate gain over LRU, but we have no proof that this is the worst case of highest difference.

C. *Accuracy of the Che approximation on the entire range for Zipf distributed IRM requests*

Finally, we compare the Che approximation [9] with simulated LRU hit rates, confirming a surprisingly good match. High accuracy of the Che approximation is supported by mathematical arguments in [14] but without quantitative results in terms of concrete boundaries. Thus we checked the deviations between simulation results and the Che approximation again for all combinations of parameters on the grid $(\beta, N, M) \in$

$$\{0.4, 0.5, ..., 1.1\} \times \{10^2, 10^3 ..., 10^7\} \times \{M_{1\%}, M_{2\%}, ..., M_{99\%}\}$$

for Zipf distributed requests.

As the main result shown in Figure 10, we observe absolute differences between simulation and the approximation of less than 0.4%. The majority of evaluated differences were even below 0.02%. Each simulation result is again based on 10^8 or more requests until a standard deviation less than $5 \cdot 10^{-5}$ is confirmed [19]. 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 large β.

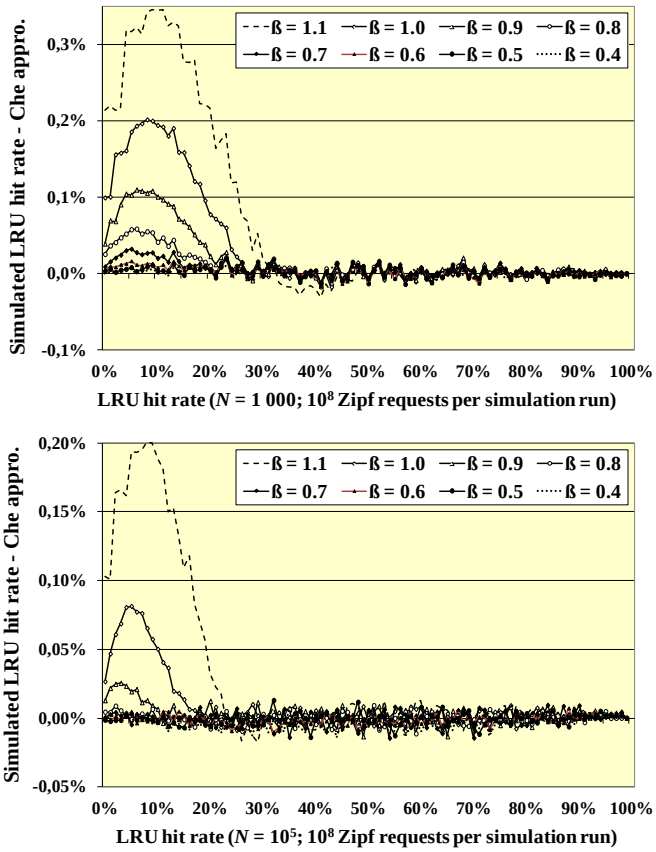

Figure 10: Che approximation: Deviations from simulations

6. Evaluation for Dynamic Popularity Based on Daily Wikipedia Page Request Statistics

A. *Dynamics in content popularity*

In contrast to the IRM assumption, changing popularity of Internet content is observed in measurement based studies

- for user-generated content, e.g., videos [13][24][38],
- especially for file sharing systems [37], and
- for IP-TV applications [7][31].

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 [7]. A study on the effect of dynamics in YouTube video traces on the cache efficiency [38] confirms that request correlations on time scales of a few hours can be ignored. Instead, the time scale of a few days up to 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 [37]. 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, -1000 and -10 000 Gnutella files has been observed in [37]. The measurement study [31] 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 shortly 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 much higher dynamics in the day by day Wikipedia request pattern than the previously referred studies on video, P2P and IP-TV platforms, see Section 6.D for more details. Nevertheless, Zipf distributed requests and the conclusions on caching efficiency of the previous sections are confirmed to be fully 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 [42]. Even if the data volumes are smaller than for popular video streaming platforms, Wikipedia operates a server and caching platform on several locations to improve the delays and loading times for the users and the availability of the platform. We evaluate daily statistics of the number of requests to the top-1000 pages, which are provided since August 2015 [42]. Since the number of requests to all Wikipedia web pages is about 10-fold higher, the top-1000 analysis can only give insight into the performance of small caches with focus on the most popular pages.

In a first step, we check whether Zipf distributions can be adapted to daily top-1000 request distributions. We determine the deviation between 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. (2), where $R_d(k)$ is the sum of requests to the top-k pages (k = 1, ..., 1000) at a day $d = 1, ..., 184$ in the half year 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 distributions $W_{d,\mathrm{CDF}}(k)$ and $Z_{\mathrm{CDF}}(k)$ achieve the same level

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

The parameter β of the adapted Zipf distribution 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 defined deviation criterion always in the range $0.5 < \beta < 0.6$ except for two days with larger β up to 0.75.

Figure 11 shows the minimum, maximum and mean absolute rank deviations of Zipf distributions adapted to each daily top-1000 request statistics. All rank deviations are in a 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.

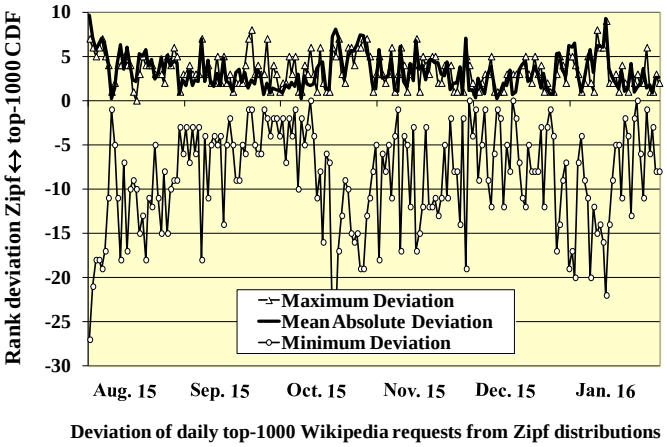

Figure 11: Zipf adaptations to daily Wikipedia page requests

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 add the top-1000 requests of several consecutive days, then we experience that such distributions for larger time frames are less variable and closer to a Zipf distribution.

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 12 shows the results on each day for SG-LRU caches with two different fading factors and LRU with cache sizes 25 and 200, respectively. Table 1 summarizes the minimum, mean and maximum values of the hit rates over all 184 days. Although the variability in the daily hit rates is high, the potential gain for SG-LRU is almost constant on each day, since SG-LRU can again almost fully exploit the maximum LFU hit rate under IRM conditions, where ρ = 0.9999 is sufficient for M = 25 and ρ = 0.99999 for M = 200, respectively. On the whole, a 10%-20% gain is confirmed similar to the results in Figure 9 also for a real request pattern due to Wikipedia statistics.

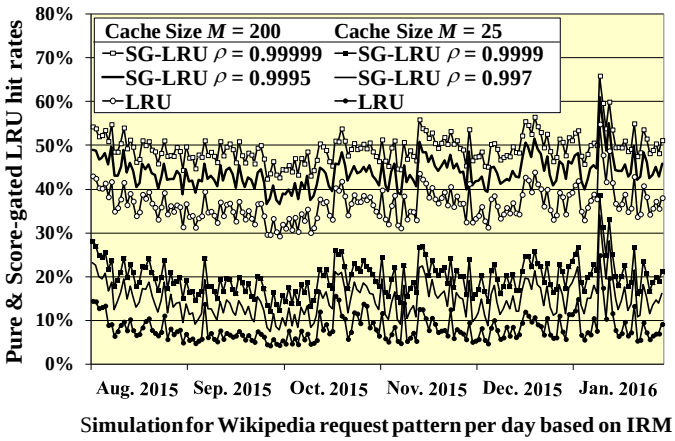

Figure 12: Cache hit evaluation for daily top-1000 requests

Table 1: Cache hit rates over the half year period

Cache Size	25			200		
	Min.	Mean	Max.	Min.	Mean	Max.
LRU hit rate	4.3%	8.3%	24.9%	29.3%	36.6%	54.9%
SG-LRU hit rate	12.2%	20.0%	38.5%	42.4%	49.4%	66.0%

D. *Cache simulation for daily changing request pattern compared to IRM hit rate results*

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 2 by the fraction of requests addressing top-k pages of the previous day.

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 76.1% 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.

Table 2: Fraction of requests to yesterday's top-k pages

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%
Y's Top-10^3 → Top-k	77.4%	78.2%	78.8%	79.2%	79.1%	76.1%

The fluctuation within the top-200 requests is captured day by day in Figure 13. A dark gray surface represents the requests to the previous day's top-200 and a light gray surface to previous top-1000 pages below a curve for the total number of requests to a steadily changing set of top-200 pages on each day.

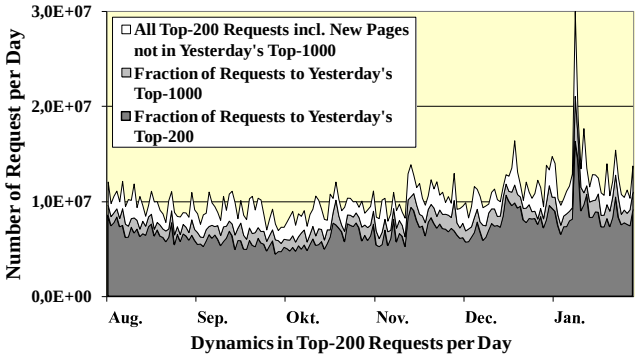

Figure 13: Changes in the daily top-200 request statistics

Finally, we simulate cache behavior over the half year time frame. On each day, we assume constant request probabilities $R_d^{(i)}/R_d$ to the pages, where $R_d^{(i)}$ is the number of requests to the page on rank i in the top-1000 statistics on day d. R_d denotes the total number of top-1000 requests on that day. For a new day, the cache starts with the content from the end of the previous day. Then the (SG-)LRU strategy has to adapt the cache content to the current request distribution in the starting phase of each day.

With a single cache for all requests, the phases adapting to daily changes are negligible within the total number $R_d \approx 2 \cdot 10^7$ of top-1000 requests per day, such that the IRM hit rate can still be closely approached for each day. Instead, we assume a number N_C of regional 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. Consequently, we simulate $\lceil R_d/N_C \rceil$ requests with constant request probabilities per day, and evaluate the caching efficiency depending on the number N_C of requests per cache per day as a characteristic factor. Figure 14 shows simulation results again for requests only to the top-1000 Wikipedia pages and for cache sizes of $M = 200$ with hit rate curves in the range > 30% as well as for $M = 25$ with hit rates < 30%, respectively. We still apply the SG-LRU caching strategy with a geometrical fading score function. The results shown in Figure 14 fall into three categories:

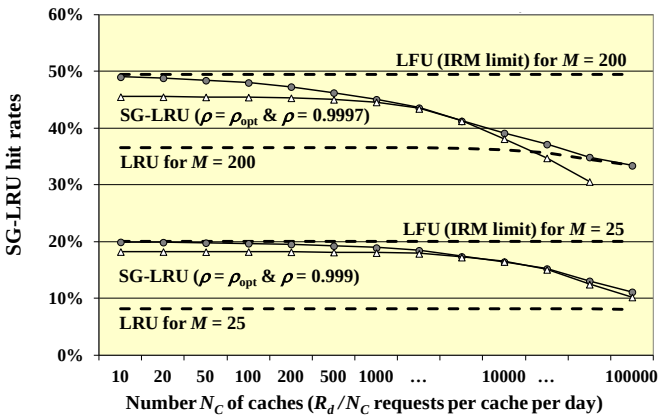

Figure 14: Cache hit rate for daily changing request pattern

- 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) per day is closely approached by SG-LRU with an 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 $N_C > 10\,000$, SG-LRU does not essentially improve pure LRU hit rates for geometrical fading scores.

The first category corresponds to $\lceil R_d/N_C \rceil > 2 \cdot 10^7/200 = 100\,000$ requests per day. In contrast to the static IRM case, SG-LRU hit rates for geometrically fading scores are now increasing up to an optimum parameter value ρ_{opt} and decreasing beyond ρ_{opt}. For example in the case $N_C = 10\,000$, $M = 25$, we observe SG-LRU hit rates going up to 16.6% for $\rho \approx 0.999$ and then reducing to 11.4% already for $\rho = 0.9999$. For larger N_C and thus smaller number of requests per cache per day, the optimum ρ_{opt} is decreasing towards $\rho_{opt} \to 0.5$, which means that SG-LRU is transforming towards pure LRU.

For further study, we consider score functions that better predict rising popularity of new pages, such that SG-LRU can improve over LRU also under higher popularity dynamics. In practice, Wikipedia and most other well known Internet platforms are supported by only a few caching locations with a high request load per cache per day. Therefore the independent request model (IRM) based on the daily request distribution is more relevant for the achievable cache hit rates.

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 daily changes in the requests to a page from R_d to R_{d+1} over several 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 [38] that dynamics in the time scale of days is more relevant.

Conclusions and Outlook

We perform extensive simulation studies to evaluate 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 [25].

In a second part, we include dynamic popularity changes based on Wikipedia page request statistics, which again exhibit Zipf-like request pattern. Although over 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 100 000 or more requests per day. The impact of dynamic request pattern on the performance becomes relevant for 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 analyzed LRU case, but can exploit the essentially higher LFU hit rate limit for Zipf distributed requests, which is simply determined by the sum of the request probabilities of the top-*M* most popular objects. Statistics about past requests is included in the proposed SG-LRU method and provides a simple extension to exploit the hit rate potential beyond pure LRU for IRM and for more realistic dynamic content popularity scenarios. We plan to proceed optimizing 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 in the EU SSICLOPS project <www.ssiclops.eu> under grant agreement No. 644866. 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] Akamai, State of the Internet, Quarterly Report Series (2017) <www.akamai.com>

[2] A. Araldo, D. Rossi, F. Martignon, Cost-aware caching: Caching more (costly items) for less (ISPs operational expenditures), IEEE Transactions on Parallel and Distributed Systems (TPDS), 27 (2016) 1316-1330

[3] Australian Bureau of Statistics, pages on Internet activity (2017) <http://abs.gov.au/ausstats/abs@.nsf/mf/8153.0>

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

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

[6] Bundesnetzagentur, Jahresbericht (2016) <www.bnetza.de> (in German)

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

[8] 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

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

[10] Cisco Systems, Visual networking index, forecast and methodology, White paper series (2017) <www.cisco.com>

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

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

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

[14] 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)

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

[16] G. Hasslinger, ISP platforms under a heavy peer-to-peer workload, Proc. P2P Systems and Applications, Eds.: R. Steinmetz and K. Wehrle, Springer LNCS 3485 (2005) 369-382

[17] 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

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

[19] 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

[20] G. Hasslinger, K. Ntougias, F. Hasslinger and O. Hohlfeld, Performance Evaluation for New Web Caching Strategies Combining LRU for Score Based Object Selection, Proc. Internat. Teletraffic Congress ITC'28, Würzburg, Germany (2016) 321-330

[21] 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

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

[23] 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

[24] 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)

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

[26] Hong Kong Office of the communications Authority (OFCA), Statistics of the Internet Traffic Volume (2017) <www.ofca.gov.hk>

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

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

[29] PeerApp Inc., <www.peerapp.com>

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

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

[32] Qwilt Inc., <www.quilt.com>

[33] M. Rabinovich and O. Spatcheck, Web caching and replication, Addison Wesley (2002)

[34] Sandvine Inc., Internet phenomena report <www.sandvine.com> (2017)

[35] K. Shah, A. Mitra and D. Matani, An O(1) algorithm for implementing the LFU cache eviction scheme (2010) see <dhruvbird.com/lfu.pdf>; <en.wikipedia.org/wiki/Least_frequently_used>

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

[37] D. Stutzbach, S. Zhao, and R. Rejaie, Characterizing files in the modern Gnutella network: A measurement study, Multimedia Systems 13/1 (2007) 35-50

[38] M. Tortelli, D. Rossi and E. Leonardi, ModelGraft: Accurate, Scalable, and Flexible Performance Evaluation of General Cache Networks, Proc. Internat. Teletraffic Congress ITC'28, Würzburg, Germany (2016)

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

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

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

[42] 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

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