# Understanding Sharded Caching Systems

Lorenzo Saino, Ioannis Psaras and George Pavlou
Department of Electronic and Electrical Engineering
University College London, London, UK
Email: {l.saino,i.psaras,g.pavlou}@ucl.ac.uk

***Abstract*—Sharding is a method for allocating data items to nodes of a distributed caching or storage system based on the result of a hash function computed on the item identifier. It is ubiquitously used in key-value stores, CDNs and many other applications. Despite considerable work has focused on the design and the implementation of such systems, there is limited understanding of their performance in realistic operational conditions from a theoretical standpoint. In this paper we fill this gap by providing a thorough modeling of sharded caching systems, focusing particularly on load balancing and caching performance aspects. Our analysis provides important insights that can be applied to optimize the design and configuration of sharded caching systems.**

## I. Introduction

*Sharding*[1] is a widely used technique to horizontally scale storage and caching systems and to address both processing and storage capacity bottlenecks. According to this technique, a large set of items is partitioned into a set of segments, named *shards*, based on the result of a hash function computed on the identifier of the item. Each shard is then mapped to a physical storage or caching device. This technique practically enables to partition data across members of a cluster and to identify the member of the cluster responsible for a given item by simply computing a hash function. Normally, sharding is used in conjunction with consistent hashing [3] to minimize the remapping of items as a result of cluster members joining or leaving the system.

Sharding is widely used in a variety of applications. It is for example ubiquitously implemented in database systems[2], Web caches in enterprise networks [2], [4], CDNs [3], [5] and key-value stores [6]. It is also used to partition large forwarding tables across network cards of a router [7] or across different routers of a network [8]. More recently, sharding has been applied to single-node key-value store implementations in order to partition items across memory regions and CPU cores [9].

However, despite a lot of work on the design and implementation of sharded systems, there is little theoretical understanding of sharding properties under realistic conditions. In this paper, we shed light on the performance of such systems. Our contribution focuses on two main aspects of sharded caching systems: load balancing and caching performance.

[1]The term *sharding* is sometimes referred to, in literature, as *hash partitioning* [1] or *hash routing* [2]. Throughout this paper, we use these terms interchangeably.

[2]To the best of our knowledge all major database implementations, both relational and NoSQL, currently support sharding.

With respect to load balancing, we first investigate how skewed distributions of item request frequency and heterogeneous item size affect imbalance. We then focus on understanding how item chunking and frontend caches can be used to mitigate it. Our findings show that the skewness of item popularity distribution considerably increases load imbalance. On the other hand both item chunking and frontend caches are effective solutions. We derive practical formulas for dimensioning frontend caches for load imbalance reduction.

With respect to caching performance, we analyze the behavior of a system of shards, each performing caching decisions independently. Our main finding is that under realistic operational conditions a system of sharded caches yields approximately the same cache hit ratio of a single cache as large as the cumulative size of all shards of the system. This result has important practical implications because it makes possible to model a sharded caching system as a single cache, hence making its analysis considerably more tractable.

The remainder of the paper is organized as follows. In Sec. II, we introduce the system model adopted throughout this paper. We analyze the performance of sharded systems with respect to load balancing in Sec. III and caching performance in Sec. IV. Finally, we draw our conclusions in Sec. V.

## II. System model

In this section we describe the system model adopted throughout this paper and explain assumptions and notation, which we report in Tab. I.

TABLE I: Summary of notation

| Symbol | Notation |
|---|---|
| $N$ | Number of items |
| $K$ | Number of shards |
| $C$ | Cache size |
| $L$ | Fraction of requests served by a shard |
| $X_i$ | Variable valued 1 if item $i$ assigned to shard, 0 otherwise |
| $\Lambda$ | IRM demand |
| $p_i$ | Probability of item $i$ being requested |
| $h_i$ | Cache hit ratio of item $i$ |
| $\alpha$ | Zipf exponent of item request frequency distribution |
| $H_N^{(\alpha)}$ | Generalized $N$-th order harmonic number |
| $T$ | Characteristic time of a cache |

Consistently with previous work, we assume that item requests conform to the Independent Reference Model (IRM) [10], which implies that the probability of each item being

978-1-4673-9953-1/16/$31.00 ©2016 IEEE

requested is stationary and independent of previous requests. Requests are issued for items $1, \ldots, N$ of a fixed catalog of size $N$ with probability $\Lambda = \{p_1, p_2, \ldots, p_N\}$.

The system comprises $K$ shards, with $K < N$, each equipped with the same amount of caching space of $C < \lfloor N/K \rfloor$ items. We assume that items are mapped uniformly to shards according to a hash function $f_H : [1 \ldots N] \rightarrow [1 \ldots K]$. Therefore, we can model the assignment of an item $i$ to a shard using a Bernoulli random variable $X_i$ such that:

$$f_{X_i}(x) = \begin{cases} \frac{1}{K}, & x = 1 \\ 1 - \frac{1}{K}, & x = 0 \end{cases}$$

## III. Load balancing

In this section we investigate how well the randomized assignment of items spreads load across shards. This analysis applies to both caching and storage systems.

This problem is an instance of a *heavily loaded single-choice weighted balls-in-bins game*. A single-choice balls-in-bins game consists in placing each of $N$ balls into one of $K$ bins chosen independently and uniformly at random (i.u.r.). In the weighted case, each ball has a different weight, which in our case is the probability of an item being requested. In the heavily loaded case, which is the case of our interest, the number of balls is considerably larger than the number of bins, *i.e.,* $N \gg K$.

The study of balls-in-bins games has received considerable attention in the past. Unfortunately, there is very little work investigating the specific case of weighted balls-in-bins games, and, to the best of our knowledge, there no previous work satisfactorily addressing the specific problem of our interest (*i.e.,* the heavily loaded weighted case). There is instead a large amount of work addressing the unweighted case. In this context, Raab and Steger [11] showed that if $N \geq K \log(K)$, the most loaded bin receives $N/K + \Theta(\sqrt{N \log(K)/K})$ balls with a probability not smaller than $1 - N^{-\theta}$ for an arbitrarily chosen constant $\theta \geq 1$. This is the tightest bound currently known.

Limiting our analysis to the assumption of uniform popularity distribution would not allow us to understand the load balancing dynamics of sharded systems under realistic operational conditions. In fact, as we show below, skewness in item popularity distribution strongly impacts load imbalance.

In addition to considering realistic item popularity distributions in our model, we also make novel contributions by shedding light on the impact of heterogeneous item size, item chunking and frontend caching on load imbalance.

In line with previous work [4], we quantify load imbalance by using the coefficient of variation of load across shards $c_v(L)$[3], where $L$ is a random variable corresponding to the fraction of requests processed by a single shard. Other studies (*e.g.,* [3], [6]), instead, quantify load imbalance as the ratio between the load of the most loaded shard and the average load of a shard $\mathrm{E}[L]$. The latter metric is normally adopted in experimental work [6], where it can be easily measured, or in theoretical work assuming uniform item popularity distribution [3], where it can be tightly bounded using standard Chernoff arguments. Unfortunately, in theoretical work assuming non-uniform item popularity distribution like ours, Chernoff bounds cannot be easily applied. As a consequence it is not possible to derive tight bounds on the load of the most loaded shard. For this reason we adopt the coefficient of variation as the metric to quantify load imbalance.

[3]The coefficient of variation of a random variable is the ratio between its standard deviation and its mean value: $c_v(L) = \sqrt{\mathrm{Var}(L)}/\mathrm{E}[L]$.

### A. Base analysis

We start analyzing load balancing performance without making any assumption regarding item popularity distribution.

Adopting the notation introduced above, we can express the fraction of requests that each shard receives as:

$$L = \sum_{i=1}^{N} X_i \, p_i \tag{1}$$

Since $X_1, \ldots, X_N$ are i.i.d. random variables, the expected value and variance of $L$ can be calculated as:

$$\mathrm{E}[L] = \mathrm{E}\left[\sum_{i=1}^{N} X_i \, p_i\right] = \sum_{i=1}^{N} \mathrm{E}[X_i] \, p_i = \frac{1}{K}$$

$$\mathrm{Var}(L) = \mathrm{Var}\left(\sum_{i=1}^{N} X_i \, p_i\right) = \sum_{i=1}^{N} \mathrm{Var}(X_i) \, p_i^2 = \frac{K-1}{K^2} \sum_{i=1}^{N} p_i^2$$

We can now derive $c_v(L)$ as:

$$c_v(L) = \frac{\sqrt{\mathrm{Var}(L)}}{\mathrm{E}[L]} = \sqrt{K-1} \sqrt{\sum_{i=1}^{N} p_i^2} \tag{2}$$

We can immediately observe from Eq. 2 that the load imbalance increases proportionally to the square root of the number of shards $K$ and to the skewness of item popularity (quantified by $\sum_{i=1}^{N} p_i^2$). Regarding the impact of item popularity skewness on load imbalance, the minimum value of $c_v(L)$ occurs when all items are equally probable, *i.e.,* $p_i = 1/N \quad \forall i \in [1 \ldots N]$, while the maximum is when one item is requested with probability 1 and all other items with probability 0. We formalize these considerations by deriving the following bounds for $c_v(L)$.

**Theorem 1.** *Let $L$ be the fraction of requests a shard receives in a system of $K$ shards subject to an arbitrary IRM demand over a catalog of $N$ items. Then:*

$$\sqrt{\frac{K-1}{N}} \leq c_v(L) \leq \sqrt{K-1} \tag{3}$$

*Proof.* The theorem can be proved immediately by jointly applying Hölder's inequality and Minkowski's inequality to bound $\sum_{i=1}^{N} p_i^2$:

$$\frac{1}{N} \left(\sum_{i=1}^{N} p_i\right)^2 \leq \sum_{i=1}^{N} p_i^2 \leq \left(\sum_{i=1}^{N} p_i\right)^2$$

Substituting $\sum_{i=1}^{N} p_i = 1$ (which holds since $p_i$ is the probability of an item $i \in [1 \dots N]$ being requested), we can rewrite the inequality as:

$$\frac{1}{N} \leq \sum_{i=1}^{N} p_i^2 \leq 1 \tag{4}$$

Substituting Eq. 2 into Eq. 4 and rearranging, we obtain Eq. 3 □

The above bounds, derived without assuming anything about item popularity, are however of little practical interest since they can span few orders of magnitude in realistic conditions (*i.e.,* $N \geq 10^6$). In the following section, we derive more useful results by making assumptions on the distribution of item popularity.

### B. Impact of item popularity distribution

We extend our analysis by assuming that item popularity follows a Zipf distribution, which is known to model very well the workloads of applications of our interest, such as Web content distribution [12] and key-value stores [6], [13]. According to this distribution, the probability of an item being requested is:

$$p_i = \frac{i^{-\alpha}}{\sum_{i=1}^{N} i^{-\alpha}} = \frac{i^{-\alpha}}{H_N^{(\alpha)}} \tag{5}$$

where $H_N^{(\alpha)}$ is the generalized $N^{th}$ order harmonic number and $\alpha > 0$ is a parameter affecting the skewness of the distribution. The greater $\alpha$, the greater is the skewness. The value of $\alpha$ which best fits real operational systems varies, but measurement studies report values in the interval $[0.6, 1.1]$.

In the remainder of this section we provide a closed-form approximation for $c_v(L)$ under a Zipf-distributed demand that we will use to derive the results presented in Sec. III-E

**Theorem 2.** *Let $L$ be the fraction of requests a shard receives in a system of $K$ shards subject to an IRM demand over a catalog of $N$ items and item request probability distributed following a Zipf distribution with parameter $\alpha$. Then:*

$$c_v(L) \approx \begin{cases} \sqrt{K-1}\dfrac{\sqrt{(N+1)^{1-2\alpha}-1}\cdot(1-\alpha)}{\sqrt{1-2\alpha}\cdot[(N+1)^{1-\alpha}-1]} & \alpha \neq \{\frac{1}{2}, 1\} \\ \dfrac{\sqrt{(K-1)\log(N+1)}}{2(\sqrt{N+1}-1)} & \alpha = \frac{1}{2} \\ \sqrt{\dfrac{N(K-1)}{(N+1)\log^2(N+1)}} & \alpha = 1 \end{cases} \tag{6}$$

*Proof.* We start by deriving an approximated closed-form expression of $H_N^{(\alpha)}$. We do so by approximating $H_N^{(\alpha)}$ with its integral expression evaluated over the interval $[1, N+1]$:

$$\sum_{i=1}^{N} \frac{1}{i^{\alpha}} = H_N^{(\alpha)} \approx \int_1^{N+1} \frac{dx}{x^{\alpha}} = \begin{cases} \frac{(N+1)^{1-\alpha}-1}{1-\alpha}, & \alpha \neq 1 \\ \log(N+1), & \alpha = 1 \end{cases} \tag{7}$$

We then use Eq. 7 to approximate $\sum_{i=1}^{N} p_i^2$ for the three cases $\alpha \notin \{\frac{1}{2}, 1\}$, $\alpha = \frac{1}{2}$ and $\alpha = 1$:

$$\sum_{i=1}^{N} p_i^2 \Big|_{\alpha \notin \{\frac{1}{2},1\}} = \sum_{i=1}^{N} \left( \frac{i^{-\alpha}}{\sum_{j=1}^{N} j^{-\alpha}} \right)^2 = \frac{H_N^{(2\alpha)}}{\left(H_N^{(\alpha)}\right)^2} \approx \frac{\left[(N+1)^{1-2\alpha}-1\right](1-\alpha)^2}{\left[(N+1)^{1-\alpha}-1\right]^2(1-2\alpha)} \tag{8}$$

$$\sum_{i=1}^{N} p_i^2 \Big|_{\alpha=\frac{1}{2}} = \sum_{i=1}^{N} \left( \frac{i^{-1}}{\sum_{j=1}^{N} j^{-1}} \right)^2 = \frac{H_N^{(1)}}{\left(H_N^{(\frac{1}{2})}\right)^2} \approx \frac{\log(N+1)}{4\left(\sqrt{N+1}-1\right)^2} \tag{9}$$

$$\sum_{i=1}^{N} p_i^2 \Big|_{\alpha=1} = \sum_{i=1}^{N} \left( \frac{i^{-1}}{\sum_{j=1}^{N} j^{-1}} \right)^2 = \frac{H_N^{(2)}}{\left(H_N^{(1)}\right)^2} \approx \frac{N}{(N+1)\log^2(N+1)} \tag{10}$$

It should be noted that Eq. 9 and Eq. 10 could be also obtained by deriving the limits of Eq. 8 for $\alpha \to \frac{1}{2}$ and $\alpha \to 1$ respectively applying L'Hôpital's rule.

Finally, applying the approximations of Eq. 8, Eq. 9 and Eq. 10 to Eq. 2, we obtain Eq. 6. □

To better understand the impact of number of shards and item popularity distribution on load imbalance we show in Fig. 1 the value of $c_v(L)$ calculated using Eq. 6 for various values of $\alpha$ and K.

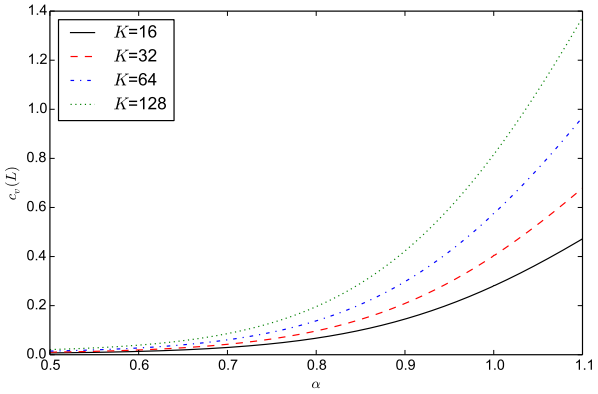



Fig. 1: $c_v(L)$ *vs.* $\alpha$ and $K$ ($N = 10^6$)

Analyzing Eq. 6 and Fig. 1, we can observe that the load imbalance is pretty close to 0 for low values of $\alpha$, increases monotonically as $\alpha$ and $K$ increase and decreases monotonically as $N$ increases. In particular, it should be noted that *item popularity skewness considerably affects the load imbalance*. This effect has been already widely experienced in operational systems [6], [14], [15]. Our findings demonstrate that unweighted balls-in-bins analyses investigating purely

the number of items assigned to each shard cannot capture accurately load imbalance dynamics under realistic demands.

### C. Impact of heterogeneous item size

If all items have the same size, the fraction of requests processed by a shard corresponds to the fraction of overall traffic it serves. However, if items have heterogeneous sizes, load imbalance in terms of number of requests and amount of traffic served do not coincide anymore. In this section we evaluate the load imbalance in terms of fraction of traffic served assuming that contents have heterogeneous size. Since previous work did not identify any significant correlation between item size and access frequency in the applications of our interest [12], [13], we assume in our analysis that item size and access frequency are independent.

**Theorem 3.** *Let $L_{(\mu,\sigma)}$ be the throughput served by a shard of a system of $K$ shards subject to demand $\Lambda = \{p_1, p_2, \ldots, p_N\}$, where the size of each item is a variable with mean $\mu$ and variance $\sigma^2$. Then:*

$$c_v\left(L_{(\mu,\sigma)}\right) = \sqrt{K}\sqrt{\left(1-\frac{1}{K}\right)+\frac{\sigma^2}{\mu^2}}\sqrt{\sum_{i=1}^{N} p_i^2} \tag{11}$$

*Proof.* Since we assume that item size and access frequency are independent, we can define $L_{(\mu,\sigma)}$ as:

$$L_{(\mu,\sigma)} = \lambda \sum_{i=1}^{N} X_i\, S_i\, p_i$$

where $S_i$ is an arbitrary random variable with mean $\mu$ and variance $\sigma^2$, representing the size of item $i$ and $\lambda$ is the overall rate of requests served by the system.

Since we assumed that $X_i$ and $S_i$ are independent we have:

$$\mathrm{E}[X_i \cdot S_i] = \mathrm{E}[X_i] \cdot \mathrm{E}[S_i] = \frac{\mu}{K}$$

$$\begin{aligned}\mathrm{Var}(X_i \cdot S_i) &= \mathrm{E}[X_i^2 \cdot S_i^2] - \mathrm{E}[X_i \cdot S_i]^2 \\ &= \frac{1}{K}\left[\left(1-\frac{1}{K}\right)\mu^2 + \sigma^2\right]\end{aligned}$$

Since $X_1S_1, \ldots, X_NS_N$ are i.i.d., expected value and variance of $L_{(\mu,\sigma)}$ can be calculated as:

$$\mathrm{E}[L_{(\mu,\sigma)}] = \mathrm{E}\left[\lambda \sum_{i=1}^{N} X_i S_i p_i\right] = \lambda \sum_{i=1}^{N} \mathrm{E}[X_i S_i]\, p_i = \frac{\lambda\mu}{K}$$

$$\begin{aligned}\mathrm{Var}(L_{(\mu,\sigma)}) &= \mathrm{Var}\left(\lambda \sum_{i=1}^{N} X_i S_i p_i\right) = \lambda^2 \sum_{i=1}^{N} \mathrm{Var}(X_i S_i)\, p_i^2 \\ &= \frac{\lambda^2}{K}\left[\left(1-\frac{1}{K}\right)\mu^2 + \sigma^2\right]\sum_{i=1}^{N} p_i^2\end{aligned}$$

Deriving $c_v(L_{(\mu,\sigma)}) = \sqrt{\mathrm{Var}(L_{(\mu,\sigma)})}/\mathrm{E}[L_{(\mu,\sigma)}]$ we obtain Eq. 11. □

We can observe from Eq. 11 that load imbalance still increases with $\sqrt{K}$ as for the homogeneous item size case. In addition and more importantly, *load imbalance also increases with the coefficient of variation of item size $c_v(S) = \sigma/\mu$.*

### D. Impact of chunking

The analysis carried out above shows the emergence of load imbalance as $\alpha$ and $K$ increase. One way to mitigate this is to split items into chunks and map each chunk to a shard independently. Splitting large items into independent chunks and handling each chunk independently is a common practice in many systems. In this section we quantify the benefit of chunking on load balancing.

**Theorem 4.** *Let $L_M$ be the load of a shard in a system where each item is split into $M$ chunks and each chunk is mapped to a shard independently. Chunking reduces the load imbalance by $\sqrt{M}$,* i.e.,

$$c_v\left(L_M\right) = \frac{c_v(L)}{\sqrt{M}} \tag{12}$$

*Proof.* We can reasonably assume that chunks of the same item are requested with the same probability and let $X_{i,j}$ be a Bernoulli random variable taking value 1 if chunk $j$ of item $i$ is mapped to the shard under analysis and 0 otherwise. The load at each shard can then be modeled as:

$$L_M = \sum_{i=1}^{N}\sum_{j=i}^{M} X_{i,j}\frac{p_i}{M} = \frac{1}{M}\sum_{i=1}^{N} Y_i\, p_i$$

where $Y_i = \sum_{j=1}^{M} X_{i,j} \sim B(M, \frac{1}{K})$ is a binomial random variable.

Since $Y_1, \ldots, Y_N$ are i.i.d., $c_v(L_M)$ can be easily calculated as:

$$\begin{aligned}c_v\left(L_M\right) &= \frac{\sqrt{\mathrm{Var}(L_M)}}{\mathrm{E}[L_M]} \\ &= \frac{\sqrt{\frac{1}{M^2}\sum_{i=1}^{N}\mathrm{Var}(Y_i)\, p_i^2}}{\frac{1}{M}\sum_{i=1}^{N}\mathrm{E}[Y_i]\, p_i} \\ &= \frac{\sqrt{(K-1)\sum_{i=1}^{N} p_i^2}}{\sqrt{M}}\end{aligned} \tag{13}$$

Substituting Eq. 2 into Eq. 13, we obtain Eq 12. □

Theorem 4 shows that *chunking is a very practical and effective solution to reduce load imbalance.*

### E. Impact of frontend cache

We conclude our analysis of load balancing aspects by investigating whether placing a frontend cache (or an array of frontend caches) in front of a sharded system can reduce the load imbalance experienced by shards.

Intuitively this seems to be the case, since, as we observed above, skewness in item popularity strongly increases load imbalance but at the same time, it can be addressed effectively by caching.

Fan *et al.* [16] already investigated this aspect concluding that a small frontend cache of $O(K \log K)$ items operating according to an LFU replacement policy is in fact sufficient to reduce load imbalance. However, their analysis only addresses the load imbalance caused by an adversarial workload with knowledge of the mapping between items and shards attempting to swamp a specific shard.

Our work, differently from [16], addresses the more common case of a non-adversarial workload with Zipf-distributed item popularity. We conclude that a frontend cache is in fact effective in reducing load imbalance as long as properly dimensioned.

For simplicity, the following analysis assumes a single frontend cache, depicted in Fig. 2. However, it can be trivially shown that the results derived here equally apply to the case of an array of equally sized frontend caches as long as the distribution of item request frequency is identical at all frontend nodes.

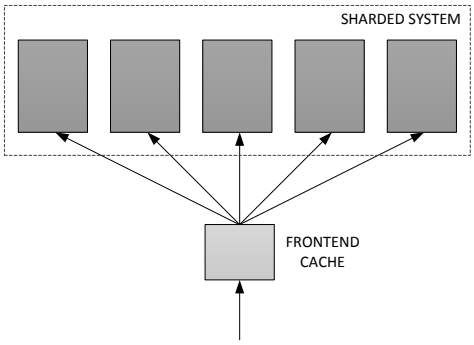



Fig. 2: Single frontend cache

Let $h_i$ be the hit probability of item $i$ at the frontend cache. The fraction of requests handled by a shard can then be written as:

$$L = \frac{\sum_{i=1}^{N} X_i \, p_i \, (1 - h_i)}{\sum_{i=1}^{N} p_i \, (1 - h_i)} \tag{14}$$

It immediately follows that its coefficient of variation is:

$$c_v(L) = \frac{\sqrt{\mathrm{Var}(L)}}{\mathrm{E}[L]} = \sqrt{K-1} \frac{\sqrt{\sum_{i=1}^{N} p_i^2 (1-h_i)^2}}{\sum_{i=1}^{N} p_i (1-h_i)} \tag{15}$$

We proceed in our analysis assuming a perfect frontend cache of size $C < N$ that always caches the $C$ most popular items. This corresponds to an LFU replacement policy since we assume an IRM workload. We reserve the extension of this analysis to include alternative replacement policies for future work.

Assuming that $p_1 \leq p_2 \leq \ldots \leq p_N$, then a perfect frontend cache permanently stores items $1, \ldots, C$. Denoting $L_C$ as the load after filtering from a frontend cache of $C$ items, we can write $c_v(L_C)$ as:

$$c_v(L_C) = \sqrt{K-1} \frac{\sqrt{\sum_{i=C+1}^{N} p_i^2}}{\sum_{i=C+1}^{N} p_i} \tag{16}$$

We further assume that item popularity follows a Zipf distribution with parameter $\alpha > 0$ and apply the approximation of Eq. 7 to derive the following closed-form expression for $c_v(L_C)$.

$$c_v(L_C) \approx \begin{cases} \sqrt{K-1} \dfrac{\sqrt{\frac{(N+1)^{1-2\alpha} - (C+1)^{1-2\alpha}}{1-2\alpha}}}{\frac{(N+1)^{1-\alpha} - (C+1)^{1-\alpha}}{1-\alpha}} & \text{if } \alpha \notin \{\frac{1}{2}, 1\} \\ \sqrt{K-1} \dfrac{\sqrt{\log(N+1) - \log(C+1)}}{2(\sqrt{N+1} - \sqrt{C+1})} & \text{if } \alpha = \frac{1}{2} \\ \sqrt{K-1} \dfrac{\sqrt{\frac{N-C}{(N+1)(C+1)}}}{\log(N+1) - \log(C+1)} & \text{if } \alpha = 1 \end{cases} \tag{17}$$

We can now present our main theorem and discuss its implications.

**Theorem 5.** *Let $L_C$ be the load at a shard subject to a Zipf-distributed IRM demand with skewness parameter $\alpha$ pre-filtered by a frontend perfect cache of size $C$, with $C < N$. Then $c_v(L_C)$ is a convex function with respect to $C$ and has a global minimum for:*

$$C^* = \underset{C}{\operatorname{argmin}} \, c_v(L_C) = \gamma(N+1) - 1 \tag{18}$$

*where, if $\alpha \notin \{\frac{1}{2}, 1\}$, $\gamma$ is the unique solution of:*

$$2(1-\alpha)\gamma^{2\alpha-1} - (1-2\alpha)\gamma^{\alpha-1} - 1 = 0 \tag{19}$$

*in the open interval $\gamma \in (0, 1)$, otherwise:*

$$\gamma = \begin{cases} -\frac{1}{4} \, W_{-1}\left(-\frac{1}{2} e^{-\frac{1}{2}}\right)^{-2} \approx 0.08 & \text{if } \alpha = \frac{1}{2} \\ -\frac{1}{2} \cdot W_0\left(-2e^{-2}\right) \approx 0.2032 & \text{if } \alpha = 1 \end{cases} \tag{20}$$

*where $W_0$ and $W_{-1}$ denote respectively the main and lower branches of the Lambert W function.*

*Proof.* We start deriving the partial derivative of $c_v(L_C)$ with respect to $C$ for the three cases $\alpha \notin \{\frac{1}{2}, 1\}$, $\alpha = \frac{1}{2}$ and $\alpha = 1$, which we report in Eq. 21.

First, we analyze the case $\alpha \notin \{\frac{1}{2}, 1\}$. Solving the inequality $\partial c_v(L_C)/\partial C \geq 0$ and substituting $\gamma = \frac{C+1}{N+1}$ yields:

$$\frac{1-\alpha}{1-2\alpha} \left[2(1-\alpha)\gamma^{2\alpha-1} - (1-2\alpha)\gamma^{\alpha-1} - 1\right] \geq 0$$

Now, we analyze the behaviour of $\partial c_v(L_C)/\partial C$ to demonstrate that $c_v(L)$ has a unique global minimum for $C = C^*$.

Since by definition $0 \leq C < N$, we can immediately observe that $\gamma \in (0, 1)$. More specifically, $\gamma \to 0^+$ when $C \to 0$ and $N \to +\infty$, while $\gamma \to 1^-$ when $C \to N$. The values of $\partial c_v(L_C)/\partial C$ at the boundaries of the $\gamma$ interval are:

$$\lim_{\gamma \to 0^+} \frac{\partial c_v(L_C)}{\partial C} = \begin{cases} -\frac{1-\alpha}{1-2\alpha} & \text{if } \alpha > 1 \\ -\infty & \text{if } \alpha \in \left(0, \frac{1}{2}\right) \cup \left(\frac{1}{2}, 1\right) \end{cases}$$

$$\lim_{\gamma \to 1^-} \frac{\partial c_v(L_C)}{\partial C} = 0 \quad \forall \alpha \in \mathbb{R}^+ \backslash \left\{\frac{1}{2}, 1\right\}$$

$$\frac{\partial c_v(L_C)}{\partial C} \approx \begin{cases} \dfrac{\sqrt{K-1}(1-\alpha)\left[2\frac{1-\alpha}{1-2\alpha}\left((N+1)^{1-2\alpha}-(C+1)^{1-2\alpha}\right)-(C+1)^{-\alpha}\left((N+1)^{1-\alpha}-(C+1)^{1-\alpha}\right)\right]}{2\sqrt{\frac{(N+1)^{1-2\alpha}-(C+1)^{1-2\alpha}}{1-2\alpha}}(C+1)^{\alpha}\left[(N+1)^{1-\alpha}-(C+1)^{1-\alpha}\right]^2} & \text{if } \alpha \notin \{\frac{1}{2},1\} \\ \dfrac{\sqrt{K-1}\left[\log\left(\frac{N+1}{C+1}\right)-\sqrt{\frac{N+1}{C+1}}+1\right]}{4\sqrt{C+1}\left(\sqrt{N+1}-\sqrt{C+1}\right)^2\sqrt{\log\left(\frac{N+1}{C+1}\right)}} & \text{if } \alpha = \frac{1}{2} \\ \dfrac{\sqrt{K-1}\left[(N-C)-\frac{1}{2}(N+1)\log\left(\frac{N+1}{C+1}\right)\right]}{(C+1)^2(N+1)\log^2\left(\frac{N+1}{C+1}\right)\sqrt{\frac{N-C}{(N+1)(C+1)}}} & \text{if } \alpha = 1 \end{cases} \tag{21}$$

Since by definition $\alpha > 0$, then:

$$\lim_{\gamma\to 0^+}\frac{\partial c_v(L_C)}{\partial C} < 0 \quad \forall \alpha \in \mathbb{R}^+ \backslash \left\{\frac{1}{2}, 1\right\}.$$

We now investigate under what conditions $\frac{\partial \frac{\partial c_v(L_C)}{\partial C}}{\partial \gamma} \geq 0$ and observe that:

$$\frac{\partial \frac{\partial c_v(L_C)}{\partial C}}{\partial \gamma} \geq 0 \Leftrightarrow (1-\alpha)^2\gamma^{\alpha-2}(-2\gamma^{\alpha}+1) \geq 0$$
$$\Leftrightarrow \gamma \leq 2^{-\frac{1}{\alpha}}$$

This shows that $\partial c_v(L_C)/\partial C$ is negative for $\gamma \to 0^+$, strictly increases for $0 < \gamma < 2^{-\frac{1}{\alpha}}$, reaches a global maximum for $\gamma = 2^{-\frac{1}{\alpha}}$ and then strictly decreases for $2^{-\frac{1}{\alpha}} < \gamma < 1$ tending to 0 for $\gamma \to 1^-$.

From this analysis we can conclude that $\partial c_v(L_C)/\partial C$ is strictly positive at its global maximum ($\gamma = 2^{-\frac{1}{\alpha}}$). Since $\partial c_v(L_C)/\partial C$ is continuous over $\gamma \in (0,1)$, applying the intermediate value theorem we can conclude that there exists at least a value of $\gamma \in (0, 2^{-\frac{1}{\alpha}})$ for which $\partial c_v(L_C)/\partial C = 0$. Since over that interval $\partial c_v(L_C)/\partial C$ is strictly increasing, that root is unique and it is a local minimum of $c_v(L_C)$. Also, this analysis shows that $\partial c_v(L_C)/\partial C$ cannot have roots for $2^{-\frac{1}{\alpha}} < \gamma < 1$. Therefore, the minimum of $c_v(L)$ is global.

Finally, since we know that the only root of $\partial c_v(L_C)/\partial C$ is the global minimum of $c_v(L_C)$, we obtain Eq. 19 by rearranging $\partial c_v(L_C)/\partial C = 0$.

We now focus on the two remaining cases: $\alpha = \frac{1}{2}, 1$. Solving the inequality $\partial c_v(L_C)/\partial C \geq 0$ and substituting, as above, $\gamma = \frac{C+1}{N+1}$ yields:

$$\frac{\partial c_v(L_C)}{\partial C} \geq 0 \Leftrightarrow \gamma \geq \begin{cases} -\frac{1}{4}W_{-1}\left(-\frac{1}{2}e^{-\frac{1}{2}}\right)^{-2} & \text{if } \alpha = \frac{1}{2} \\ -\frac{1}{2}\cdot W_0\left(-2e^{-2}\right) & \text{if } \alpha = 1 \end{cases}$$

From this inequality it is immediately evident that $c_v(L_C)$ has a global minimum when $\gamma$ is equal to the right hand side part, which corresponds to Eq. 20.

□

From Theorem 5 we can observe that the global minimum of $c_v(L_C)$ does not depend on the absolute values of $C$ or $N$ but on the quantity $\gamma = \frac{C+1}{N+1}$, as well as $\alpha$. In Fig. 3 we plot all values of $\gamma$ for which we observe a global minimum of $c_v(L_C)$ for various values of $\alpha$, which represent the skewness of item popularity distribution. Typical workloads can be modeled with $\alpha \in [0.6, 1.1]$. In that interval, the values of $\gamma$ that minimize load imbalance are comprised between 0.11 and 0.24. This implies that even in the worst case scenario, the frontend cache should be smaller than 11% of the size of item population to ensure that any addition of caching space reduces load imbalance. In practice such frontend caches are rarely larger than 1% of the item population. Hence, we conclude that *any typical frontend cache reduces load imbalance*.

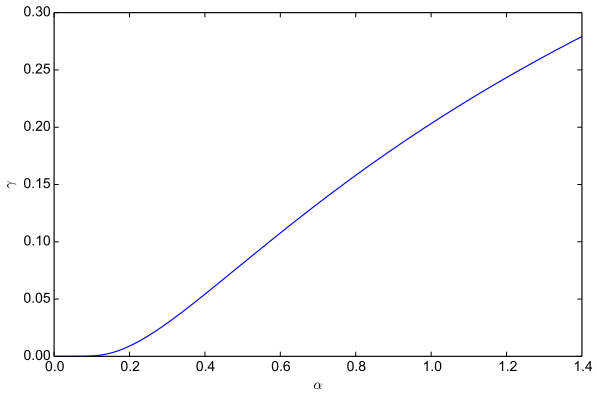



Fig. 3: Relationship between $\gamma$ and $\alpha$ parameters

As a further step in our evaluation, we quantify the actual reduction of load imbalance achieved by a perfect frontend cache and plot results in Fig. 4 for various values of $\alpha$ and cache/catalog ratio $C/N$ in the case of $N = 10^6$. In the plot, the lines labeled as *Exact* have been drawn using Eq. 16 while the lines labeled as *Model* have been drawn using Eq. 17.

This analysis allows us to draw three main conclusions. First of all, the load imbalance model, which relies on the approximation of Theorem 2, is very accurate. This is demonstrated by the almost perfect agreement between *Exact* and *Model* results in Fig. 4. Second, the imbalance reduction is greater for more skewed item distributions. This is expected because, as discussed above, more uniform distributions induce lower load imbalance and therefore there is less improvement that

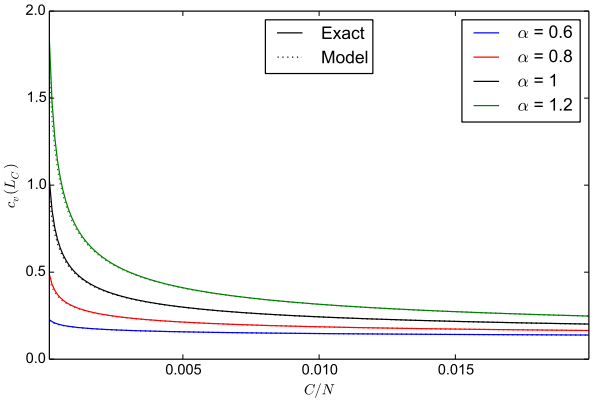



Fig. 4: Impact of a frontend perfect cache on load imbalance

a frontend cache can provide. Finally, and most importantly from a practical standpoint, *even a very small cache of around 0.5% of content catalog is sufficient to carry out a substantial reduction in load imbalance*. Further increasing cache size still reduces load imbalance but less substantially.

## IV. Cache hit ratio

We now focus on investigating the performance of a sharded system in terms of cache hit ratio, assuming that each shard is not a permanent store but evicts items according to a replacement policy independently from other shards. In particular, our objective is to *compare the performance of a system of caching shards with that of a single cache with capacity equal to the cumulative capacity of all the shards of the system*. We assume that each shard has equal size $C$, with $C < \lfloor K/N \rfloor$ items and all shards operate according to the same replacement policy.

Our analysis relies on results from *Che's approximation* which we briefly recall here. Che's approximation was originally proposed in [17] to model the cache hit ratio of an LRU cache subject to an IRM demand. It was recently generalized by Martina *et al.* [18] to support a wider variety of replacement policies, precisely FIFO, RANDOM and $q$-LRU (*i.e.,* LRU with insertion probability $q$).

This approximation introduces the concept of *characteristic time* $T$ of a cache, which corresponds to the time that an item spends in an LRU cache from its last request (which pushes it at the top of the cache) to its eviction. The characteristic time is a random variable specific to each item. Approximating it with a single constant value for all items, as proposed by Che *et al.* [17], still yields remarkably accurate results [18], [19] and widely simplifies performance analysis because it decouples the dynamics of a specific item from all other items.

A formal definition of the *characteristic time* adopted in Che's approximation [17] and its generalization of [18] is the following.

**Definition 1.** The *characteristic time* $T$ of a cache is, for LRU and $q$-LRU caches, the average time elapsed between the last request for an item and its eviction from the cache or, for FIFO and RANDOM caches, the average time elapsed between insertion and eviction of an item.

Because of the IRM demand assumption, the probability that an item is in the cache $p_{in}$ is equal by definition to its hit probability $p_{hit}$, as a consequence of the arrival theorem of a Poisson process. Applying Che's approximation, it is possible to determine the hit ratio of each item $i$ simply knowing the request probability of that item $p_i$ and the characteristic time of the cache $T$. In the specific case of the LRU replacement policy, this corresponds to:

$$p_{hit}(p_i, T) = p_{in}(p_i, T) = 1 - e^{-p_i T} \tag{22}$$

Finally, the characteristic time of a cache $T$ can be computed by solving the following equation:

$$\sum_{i=1}^{N} p_{in}(p_i, T) = C \tag{23}$$

For all cache replacement policies listed above (*i.e.,* LRU, FIFO, RANDOM and $q$-LRU), this equation does not have a closed-form solution, but can be easily solved numerically. We refer the reader to [18] for equations to derive hit ratios for those replacement policies under Che's approximation.

We formally define a cache replacement policy to which Che's approximation can be applied as follows.

**Definition 2.** A cache replacement policy is *defined by characteristic time* if the steady-state per-item hit ratio of a cache operating under such policy subject to an IRM demand is exclusively an increasing function of the request probability of the item and the characteristic time of the cache.

We note that a special consideration needs to made for the LFU replacement policy. While LFU cannot be defined by characteristic time, it was shown in [18] that it yields steady-state cache hit ratio performance identical to the $q$-LRU replacement policy (which is instead defined by characteristic time) when $q \to 0$. Therefore, all considerations regarding policies defined by characteristic time equally apply to LFU.

After these initial considerations, we present the following theorem and devote the remainder of this section to prove it and to discuss its applicability in realistic operational conditions.

**Theorem 6.** *Let $\mathcal{C}_S$ be a caching shard of capacity $C$ belonging to a system of $K$ shards, all with the same capacity and operating under the same replacement policy $\mathcal{R}$ defined by characteristic time. The overall system is subject to an IRM demand $\Lambda = \{p_1, p_2, \ldots, p_N\}$ and items $1, \ldots, N$ are mapped to shards $1, \ldots, K$ uniformly. Then, for large $C$, the cache hit ratio of $\mathcal{C}_S$ converges to the cache hit ratio of a single cache of size $K \cdot C$ operating under replacement policy $\mathcal{R}$ subject to demand $\Lambda$.*

*Proof.* We denote $T_S$ as the characteristic time of a shard of size $C$ belonging to a system of $K$ shards and $T$ as the characteristic time of a single cache of size $K \cdot C$ subject to the same demand of the overall sharded system. Assuming

uniform assignment of items to shards and applying Eq. 23 we obtain:

$$\psi\left(T_S\right) = \sum_{i=1}^{N} p_{in}(X_i\, p_i, T_S) = C \tag{24}$$

$$\phi(T) = \sum_{i=1}^{N} p_{in}(p_i, T) = K \cdot C \tag{25}$$

where $T_S$ is a random variable whose outcome depends on $X_i$, while $T$ is constant. It should be noted that $p_{in}(\cdot)$ in Eq. 24 and 25 are the same function because both caches operate according to the same replacement policy and $p_{in}$ depends only on $p_i$ and $T$ because we assumed that such policy is determined by characteristic time.

By definition, if an item is never requested it cannot be in the cache, hence $p_{in}(0, T) = 0$. Since $X_i \in \{0, 1\}$, then $p_{in}(X_i\, p_i, T_S) \equiv X_i\, p_{in}(p_i, T_S)$. We can then rewrite Eq. 24 as:

$$\psi\left(T_S\right) = \sum_{i=1}^{N} X_i\, p_{in}(p_i, T_S) = C \tag{26}$$

Let $\overline{T}_S$ be the value of $T_S$ for which $\mathrm{E}[\psi(T_S)] = C$. Therefore:

$$\begin{aligned} \mathrm{E}\left[\psi\left(\overline{T}_S\right)\right] &= \mathrm{E}\left[\sum_{i=1}^{N} X_i\, p_{in}(p_i, \overline{T}_S)\right] \\ &= \frac{1}{K} \sum_{i=1}^{N} p_{in}(p_i, \overline{T}_S) = C \end{aligned} \tag{27}$$

We now proceed showing that $T_S \approx \overline{T}_S$ for large $C$. First, we note that since by definition $p_{in}(p_i, \overline{T}_S) \in [0, 1]$, $\forall i \in [1 \dots N]$, the following inequality holds:

$$\sum_{i=1}^{N} \left[p_{in}(p_i, \overline{T}_S)\right]^2 \leq \sum_{i=1}^{N} p_{in}(p_i, \overline{T}_S) \tag{28}$$

Jointly applying this inequality and Eq. 27, we derive the following upper bound of $\mathrm{Var}(\psi(\overline{T}_S))$:

$$\begin{aligned} \mathrm{Var}\left(\psi\left(\overline{T}_S\right)\right) &= \mathrm{Var}\left(\sum_{i=1}^{N} X_i\, p_{in}(p_i, \overline{T}_S)\right) \\ &= \frac{K-1}{K^2} \sum_{i=1}^{N} \left(p_{in}(p_i, \overline{T}_S)\right)^2 \\ &\leq \frac{K-1}{K^2} \sum_{i=1}^{N} p_{in}(p_i, \overline{T}_S) \\ &= \left(1 - \frac{1}{K}\right) C \end{aligned} \tag{29}$$

Then, jointly applying Chebyshev's inequality and Eq. 29 to $\psi(\overline{T}_S)$ we obtain:

$$\begin{aligned} P(\left|\psi(\overline{T}_S) - E\left[\psi(\overline{T}_S)\right]\right| \geq \epsilon C) &\leq \frac{\mathrm{Var}(\psi(\overline{T}_S))}{\epsilon^2 C^2} \\ &\leq \frac{1 - 1/K}{\epsilon^2 C^2} E\left[\psi(\overline{T}_S)\right] \\ &= \left(1 - \frac{1}{K}\right) \frac{1}{\epsilon^2 C} \end{aligned} \tag{30}$$

From Eq. 30, we can immediately observe that fluctuations of $\psi(\overline{T}_S)$ around its mean increase with the number of shards $K$ and decrease as the cache size $C$ increases. If single caches of the system are large enough and/or there is a reasonably small number of shards, $\psi(\overline{T}_S)$ fluctuates very little around its mean. In these cases we can then reasonably assume that $T_S \approx \overline{T}_S$.

Applying this result to Eq. 27 and rearranging, we obtain:

$$\sum_{i=1}^{N} p_{in}(p_i, T_S) = K \cdot C \tag{31}$$

Finally, comparing Eq. 31 with Eq. 25, it can be immediately proved that the characteristic time of a cache of size $C$ in a system of $K$ caching shards is approximately equal to the characteristic time of a single cache of $K \cdot C$ size. Therefore:

$$T_S^{(C)} = \psi^{-1}(C) \approx \phi^{-1}(KC) = T^{(KC)} \tag{32}$$

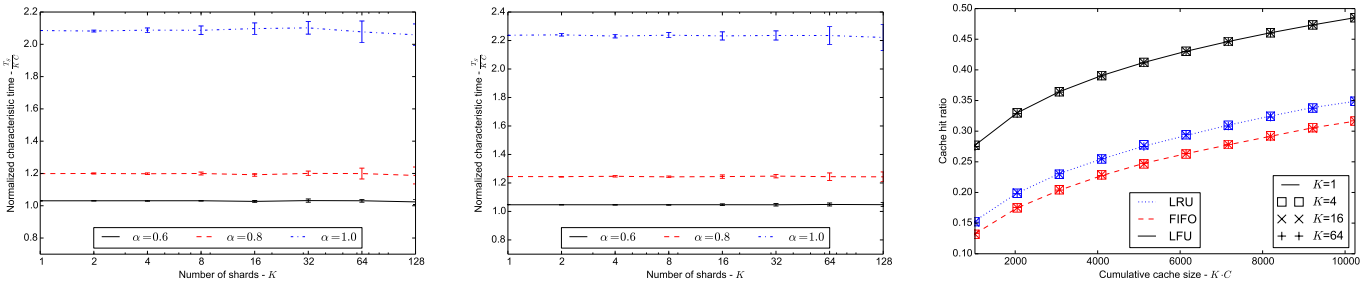
□

We further validate the approximation of Theorem 6 numerically and draw results in Fig. 5a and 5b, where we show the value of $T_S$ for different values of $K$ for the cases of LRU and FIFO/RANDOM caches subject to a demand with Zipf-distributed item popularity. The results are consistent with our analysis. The mean value of $T_S$ is not affected by variations in number of shards $K$ and is the same as in the case of a single cache ($K = 1$). The standard deviation of $T_S$, represented by the error bars, increases with $K$, as expected, but it remains low in comparison to the mean value of $T_S$.

The insensitivity of $T_S$ against $K$ is also expectedly reflected in caching performance. We measured the cache hit ratio yielded by systems of shards subject to a Zipf-distributed IRM demand (with $\alpha = 0.8$), operating under the LRU, FIFO and LFU replacement policies for various caching sizes and values of $K$. The results, plotted in Fig. 5c, show again insensitivity of cache hit ratio with respect to $K$, further validating our analysis.

## V. Conclusions

In this paper, we presented novel and practical results shedding light on the performance of sharded caching systems. Our analysis focused primarily on load balancing and caching performance aspects. With respect to load balancing, we characterized load imbalance among shards caused by content popularity skewness and heterogeneous item sizing. We also quantified how item chunking and the deployment of small frontend caches can reduce such imbalance. With respect to




(a) Characteristic time ($T_S$) *vs.* number of shards ($K$) and $\alpha$ (LRU case)

(b) Characteristic time ($T_S$) *vs.* number of shards ($K$) and $\alpha$ (FIFO/RANDOM case)

(c) Cache hit ratio for single cache and systems of shards

Fig. 5: Caching performance of sharded systems

caching performance, we characterized the cache hit ratio of sharded systems and showed that sharding does not affect caching performance if caches are large enough. We believe that our result are easily applicable and can improve the design of such systems.

## ACKNOWLEDGEMENTS

We would like to thank Emilio Leonardi for insightful comments on earlier drafts of this paper.

The research leading to these results was funded by the UK Engineering and Physical Sciences Research Council (EPSRC) under grant no. EP/K019589/1 (the COMIT Project) and the EU-Japan initiative under European Commission FP7 grant agreement no. 608518 and NICT contract no. 167 (the GreenICN Project).

## REFERENCES

[1] D. Zhou, B. Fan, H. Lim, D. G. Andersen, M. Kaminsky, M. Mitzenmacher, R. Wang, and A. Singh, "Scaling up clustered network appliances with ScaleBricks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication (SIGCOMM'15)*. New York, NY, USA: ACM, 2015, pp. 241–254.

[2] K. Ross, "Hash routing for collections of shared web caches," *IEEE Network*, vol. 11, no. 6, pp. 37–44, Nov 1997.

[3] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web," in *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing (STOC'97)*. New York, NY, USA: ACM, 1997, pp. 654–663.

[4] D. G. Thaler and C. V. Ravishankar, "Using name-based mappings to increase hit rates," *IEEE/ACM Transactions on Networking*, vol. 6, no. 1, pp. 1–14, Feb. 1998.

[5] B. M. Maggs and R. K. Sitaraman, "Algorithmic nuggets in content delivery," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, pp. 52–66, Jul. 2015.

[6] Q. Huang, H. Gudmundsdottir, Y. Vigfusson, D. A. Freedman, K. Birman, and R. van Renesse, "Characterizing load imbalance in real-world networked caches," in *Proceedings of the 13th ACM Workshop on Hot Topics in Networks (HotNets-XIII)*. New York, NY, USA: ACM, 2014, pp. 8:1–8:7.

[7] D. Perino, M. Varvello, L. Linguaglossa, R. Laufer, and R. Boislaigue, "Caesar: A content router for high-speed forwarding on content names," in *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'14)*. New York, NY, USA: ACM, 2014, pp. 137–148.

[8] H.-g. Choi, J. Yoo, T. Chung, N. Choi, T. Kwon, and Y. Choi, "CoRC: Coordinated Routing and Caching for Named Data Networking," in *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'14)*. New York, NY, USA: ACM, 2014, pp. 161–172.

[9] H. Lim, D. Han, D. G. Andersen, and M. Kaminsky, "MICA: A holistic approach to fast in-memory key-value storage," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI'14)*. Berkeley, CA, USA: USENIX Association, 2014, pp. 429–444.

[10] E. G. Coffman, Jr. and P. J. Denning, *Operating Systems Theory*. Prentice Hall Professional Technical Reference, 1973.

[11] M. Raab and A. Steger, "Balls into bins - a simple and tight analysis," in *Randomization and Approximation Techniques in Computer Science*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, vol. 1518, pp. 159–170.

[12] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: evidence and implications," in *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99)*, vol. 1, Mar 1999.

[13] B. Atikoglu, Y. Xu, E. Frachtenberg, S. Jiang, and M. Paleczny, "Workload analysis of a large-scale key-value store," in *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '12)*. New York, NY, USA: ACM, 2012, pp. 53–64.

[14] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani, "Scaling Memcache at Facebook," in *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation (NSDI'13)*. Berkeley, CA, USA: USENIX Association, 2013, pp. 385–398.

[15] Y.-J. Hong and M. Thottethodi, "Understanding and mitigating the impact of load imbalance in the memory caching tier," in *Proceedings of the 4th Annual Symposium on Cloud Computing (SOCC'13)*. New York, NY, USA: ACM, 2013, pp. 13:1–13:17.

[16] B. Fan, H. Lim, D. G. Andersen, and M. Kaminsky, "Small cache, big effect: Provable load balancing for randomly partitioned cluster services," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC'11)*. New York, NY, USA: ACM, 2011, pp. 23:1–23:12.

[17] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas of Communications*, vol. 20, no. 7, pp. 1305–1314, Sep. 2006.

[18] V. Martina, M. Garetto, and E. Leonardi, "A unified approach to the performance analysis of caching systems," in *Proceedings of the 2014 IEEE Conference on Computer Communications (INFOCOM'14)*, April 2014, pp. 2040–2048.

[19] C. Fricker, P. Robert, and J. Roberts, "A versatile and accurate approximation for LRU cache performance," in *Proceedings of the 24th International Teletraffic Congress (ITC'12)*. International Teletraffic Congress, 2012, pp. 8:1–8:8.