Balanced Allocations with Heterogenous Bins

Udi Wieder
Microsoft Research Silicon Valley
uwieder@microsoft.com

March 26, 2007

Abstract

Balls-into-bins processes are a useful and common abstraction for many load-balancing related problems. A well known paradigm for load balancing in distributed or parallel servers is the "multiple choice paradigm" where an item (ball) is put in the less loaded out of d uniformly chosen servers (bins). In many applications however the uniformity of the sampling probability is not guaranteed. If the system is heterogenous or dynamic it may be the case that some bins are sampled with a higher probability than others. We investigate the power of the multiple choice paradigm in the setting where bins are *not* sampled from the uniform distribution. Byers *et al* [5] showed that a logarithmic imbalance in the sampling probability could be tolerated, as long as the number of balls is linear in the number of bins. We show that if the number of balls is much larger than the number of bins, this ceases to be the case. Given a probability over bins, we prove tight upper and lower bounds for the number of choices needed in the 1-out-of-d scheme in order to maintain a balanced allocations when the number of items is arbitrarily high.

1 Introduction

Balls-into-bins processes serve as a useful abstraction for resource balancing tasks in distributed and parallel systems. Assume m balls are to be put sequentially into n bins, where typically the goal is to minimize the load, measured by the number of balls, in the most loaded bin. In the classic *single choice process* each ball is placed in a bin chosen independently and uniformly at random. For the case of n bins and $m = n$ balls it is well known that the load of the heaviest bin is at most $(1 + o(1))\frac{\ln n}{\ln \ln n}$ balls with high probability. If $m \geq n \ln n$ then the load in the heaviest bin is at most $\frac{m}{n} + \sqrt{\frac{m \log n}{n}}$, see e.g. [15].

A substantial decrease in load is achieved by the use of the *multiple choice paradigm*: Let $\textsc{Greedy}(\mathcal{U}, d)$ denote the algorithm where each ball is inserted into the less loaded among $d \geq 2$ bins, independently sampled from $\mathcal{U}$ where $\mathcal{U}$ denotes the uniform distribution over the bins. In a seminal paper Azar *et al* [2] proved that when $m = n$ balls are inserted by $\textsc{Greedy}(\mathcal{U}, d)$ the heaviest bin has load of $\frac{\log \log n}{\log d} + \Theta(1)$ with high probability, the case $d = 2$ was implicitly proved by Karp *et al* in [8]. Berenbrink *et al* [3] generalized this result and proved the following.

Theorem 1.1 ([3]). *Let γ denote a suitable constant. If m balls are allocated into n bins using* GREEDY$(\mathcal{U}, d)$ *with $d \geq 2$ then the number of bins with load at least $\frac{m}{n}+i+\gamma$ is at most $n \cdot \exp(-d^i)$ with probability at least $1 - \frac{1}{n}$.*

An immediate corollary is that w.h.p. the heaviest bin is of load $\frac{m}{n} + \frac{\log \log n}{\log d} + O(1)$. Thus, the additive gap between the maximum load and the average load is *independent* of the number of balls thrown.

The multiple choice algorithm has numerous variations and applications in many settings, see e.g. [1],[18],[17],[10]. See [13] for a survey. A typical application is the following. Assume data items (balls) are to be inserted into servers (bins) in a distributed manner. Upon an arrival of a data item d hash functions are employed on the data item in order to randomly select d servers. The data item is then stored on the less loaded among the d random servers. When searching for a data item the d hash functions identify d potential servers, one of which contains the data item (c.f. [4] [10]). Now, if each server is sampled by the hash functions with equal probability, then Theorem 1.1 guarantees that the allocation is highly balanced; with high probability no server stores more than $\frac{m}{n} + \frac{\log \log n}{\log d} + O(1)$ data items. In practice however, it is unlikely that all servers are sampled with the same probability. There are two main reasons that cause a biased sampling distribution.

Uneven allocation of the key space

In many applications the set of servers is dynamic and may change over time. Such applications naturally include P2P systems (c.f. [16],[14], [11]), but also include centrally owned data centers which occasionally add servers to the system in order to increase capacity [10]. A typical scheme for using hash functions in dynamic settings such as distributed hash tables is as follows. Assume the hash functions hash a data item's identifier to the domain $[0, 1)$. The idea is to assign to each server an i.d. from $[0, 1)$ as well. Now a hash function samples a server by choosing the server whose i.d. is closest to the sampled point in the hash function key space. Thus, the probability a server is sampled by a hash function is proportional to the size of the key space it "owns". The partition of the key space between the servers is done dynamically and changes as servers enter and leave the system. This technique is sometimes called *consistent hashing* [6] and is used in many storage systems.

There are various techniques to divide the key space between the servers as evenly as possible. The simplest thing to do is to let each server choose its i.d. uniformly at random from the key space (perhaps via a hash function as well) [16]. This results in an uneven sampling probability as some servers are sampled with probability $\log n/n$ and some with probability $1/n \log n$ [11]. A more balanced allocation of the key space is obtained by using subtler algorithms c.f. [11] [14] [12] [7], none of them however guarantee a perfectly uniform sampling distribution. In centrally owned data bases there are better schemes [9] but even they only guarantee a gap of factor 2 between the sampling probabilities. It is easy to see that such a sampling imbalance is an inherent property of the approach. There is no way to guarantee a completely balanced partition of the key space among the servers without performing n updates whenever a new server is introduced.

Heterogenous capacities of the bins

Different capacities among bins may cause imbalance even when all bins are sampled with equal probability. Consider for instance the example above where data items are to be inserted into servers. A typical large storage system would have a large variety of server configurations with heterogenous abilities. Assume for instance, that half of the servers are old and slow and half of the servers are of a newer generation and that a new server has 4 times more capacity than an old one. Ideally each new server should hold 4 times more items, so we would like $m/5$ items to be stored in the old servers. Yet, if GREEDY$(\mathcal{U}, 2)$ is employed then in each iteration with probability $1/4$ both random servers are old servers. Thus on expectation at least $m/4$ data items are stored in the old servers causing the allocation to be imbalanced.

We claim that the problem of dealing with heterogenous capacities could be reduced to dealing with a heterogenous sampling distribution. Each new server can simulate 4 servers, so one possible solution is to have the sampling probability of the new server be 4 times larger than that of an old server. This solution is problematic in many settings as it requires global adjustments whenever a server is replaced by a stronger server. A different approach is to have a scheme which is oblivious to the different capacities of the servers. If the probability the new server is sampled is p, we can think of the new server as being composed of 4 virtual servers, each one of them sampled with probability $p/4$. We conclude that a scheme that can accommodate a gap of factor 4 between sampling probabilities would maintain a balanced allocation also in the face of heterogenous capacities.

1.1 Related Work

Byers *et al* [4] address the aforementioned problem that in P2P systems the sampling distribution of the servers may not be uniform. They prove that for a typical P2P implementation when $m = n$ the maximum load obtained by the two choice paradigm; i.e. by GREEDY$(\mathcal{D}, 2)$, remains $\log \log n$. In [5] they generalize this result and consider the following setting. Both servers and items are hashed to a geometric space. The servers are randomly placed in the geometric space. Data items choose two (or more) *locations* randomly in the geometric space and pick the one where the nearest neighbor server has the smallest load. Since the sampling procedure picks locations and not servers, the probability a server is sampled is proportional to the volume of its Voronoi cell. Such scenarios arise naturally in P2P applications. They show that if n balls are placed then the maximum load is still $\log \log n / \log d$ with high probability even when the number of choices is 2.

This is a strong result: note that the sampling distribution is far from uniform: when n servers are randomly placed in $[0, 1)$, some servers may be sampled with probability $\frac{\log n}{n}$ while others with probability $\frac{1}{n \log n}$ [11]. Thus, Byers *et al* [5] show that when $m = n$ GREEDY$(\mathcal{D}, 2)$ is effective even when some servers are sampled more often by a logarithmical factor. Yet, the result ceases to be correct when the number of balls is larger than n. Consider for instance a case where there are $n/4$ bins with a sampling probability of $1/2n$ or less and note that these numbers are likely to arise when server i.d's are chosen randomly in $(0, 1]$. Call these bins "*light*". The light bins consist of $1/8$ of the sample space. Now, assume that whenever a light bin is sampled it receives the ball. The probability some light bin is sampled is $1 - (7/8)^2 = 15/64$. Therefore when adding a new item each light bin would receive on average $(15/64) * (4/n) = 60/64n$ items. This is *smaller*

than $1/n$. This means that a heavy bin receives on average more than $1/n$ from each item. Now if the number of balls is large enough some bin must have more than $m/n + \log\log n$ balls.

The intuition behind the previous example is that the bins with the high sampling probability receive more than m/n ball on *average*. A slight increase in the average does not affect the result when the average is small compared to the additive imbalance, which is $\log\log n$. But when the number of balls is large enough so that the average number of balls per bin is larger than $\log\log n$, a slight imbalance among the bins breaks the result.

1.2 Our Contributions

We show that the imbalance in the sampling distribution could be mitigated by an increase in the number of choices the algorithm uses. Given a sampling distribution over the servers, we provide **tight** upper and lower bounds on the number of choices the allocation algorithm needs to use, in order for the maximum load to be $\frac{m}{n} + O(\log\log n) + O(1)$ with high probability.

Definition 1.2. *A distribution $(p_1, p_2, \ldots, p_n)$ over n elements is called α, β-*biased*, if for all i it holds that $\frac{1}{\alpha n} \leq p_i \leq \frac{\beta}{n}$, where p_i is the probability the i'th element is sampled, and $\alpha, \beta > 1$*

The numbers α and β measure the imbalance of the sampling distribution. For instance, when considering heterogenous servers, if it is guaranteed that the ratio between the capacity of any two servers is never more than γ, then $\alpha \cdot \beta \leq \gamma$. The exact values of α, β depend upon the number of servers with each capacity.

For a given distribution $\mathcal{D}$ over n elements let Greedy($\mathcal{D}, d$) be the algorithm that randomly samples d bins according to $\mathcal{D}$ and places the ball in the less loaded among the d bins. The main result of the paper is the following theorem:

Theorem 1.3. *For any $\alpha, \beta > 1$ define*

$$f(\alpha, \beta) := \frac{\ln(\frac{\alpha\beta - 1}{\alpha - 1})}{\ln(\frac{\alpha\beta - 1}{\alpha\beta - \beta})}$$

1. ***Lower bound:*** *If $d \leq (1 - \epsilon) f(\alpha, \beta)$ for some $\epsilon > 0$ then there is an (α, β)-biased distribution $\mathcal{D}$ such that if m balls are inserted using Greedy($\mathcal{D}, d$) the expected load on the heaviest bin is at least $\frac{m}{n} \cdot \left(\frac{\alpha\beta - 1}{\alpha - 1}\right)^{\epsilon}$.*

2. ***Upper bound:*** *If $d \geq (1 + \epsilon) \cdot f(\alpha, \beta)$ then for every (α, β)-biased distribution $\mathcal{D}$, if m balls are inserted using Greedy($\mathcal{D}, d$) , then with probability $1 - o(1/n)$ the maximal bin will have at most $\frac{m}{n} + \frac{\ln\ln n}{\ln(1+\epsilon)} + O(1)$ balls.*

Note that while d is an integer, $f(\alpha, \beta)$ is typically not an integer. Thus if α, β are constants independent of n then Theorem 1.3 claims that when $d = \lfloor f(\alpha, \beta) \rfloor$ the gap grows linearly with m while if $d = \lceil f(\alpha, \beta) \rceil$ the gap remains $O(\log\log n)$. In other words, the theorem provides *tight* upper and lower bounds on the number of choices needed to maintain a balanced allocation.

2 Proof of the Main Result

We first prove the lower bound. The intuition behind the lower bound is that if the number of choices is not large enough, the bins with small sampling probability will not be sampled often enough, thus causing the other servers to become overloaded. Assume that $d \leq (1-\epsilon)f(\alpha,\beta)$ for some $\epsilon > 0$. Let $\mathcal{D}$ be as follows: there are $\frac{n(\alpha-1)}{\alpha\beta-1}$ bins with sampling probability $\frac{\beta}{n}$, and $\frac{n(\alpha\beta-\alpha)}{\alpha\beta-1}$ bins with sampling probability $\frac{1}{n\alpha}$. One can verify that $\frac{n(\alpha-1)}{\alpha\beta-1}\cdot\frac{\beta}{n}+\frac{n(\alpha\beta-\alpha)}{\alpha\beta-1}\cdot\frac{1}{\alpha n}=1$. For notational brevity we assume that $\frac{n(\alpha-1)}{\alpha\beta-1}$ is an integer. Denote by H the set of $\frac{n(\alpha-1)}{\alpha\beta-1}$ bins with the high sampling probability. The probability the bins in H receive a ball is minimized when the bins in H are the $\frac{n(\alpha-1)}{\alpha\beta-1}$ most loaded bins. In that case the probability a ball falls in H is

$$\left(\frac{\alpha\beta-\beta}{\alpha\beta-1}\right)^{d}=\left(\frac{\alpha\beta-\beta}{\alpha\beta-1}\right)^{(1-\epsilon)\frac{\ln(\frac{\alpha\beta-1}{\alpha-1})}{\ln(\frac{\alpha\beta-1}{\alpha\beta-\beta})}}=\left(\frac{\alpha-1}{\alpha\beta-1}\right)^{1-\epsilon}$$

Therefore, after inserting m balls the expected total number of balls in H is at least $m\left(\frac{\alpha-1}{\alpha\beta-1}\right)^{1-\epsilon}$. Averaging over the $\frac{n(\alpha-1)}{\alpha\beta-1}$ bins in H we have that the expected load of a bin in H after throwing m balls is at least $\frac{m}{n}\cdot\left(\frac{\alpha\beta-1}{\alpha-1}\right)^{\epsilon}$, which proves the lower bound. Note that since $\left(\frac{\alpha\beta-1}{\alpha-1}\right)^{\epsilon}>1$ the gap between maximum load and average load is *linear* in m, causing the system to get unbalanced very quickly, as could be seen in Section 3.

2.1 The Upper Bound

We first prove the Theorem for the case $d \geq 2f(\alpha,\beta)$, the generalization to the case $d \geq (1+\epsilon)f(\alpha,\beta)$ is done in Section 2.2. We model the state of the system by *load vectors*. A load vector $\mathbf{x}=(x_1,x_2\ldots,x_n)$ specifies the number of balls in each bin, where x_i specifies the load of bin i. We assume the vectors are *normalized*; i.e. $x_1 \geq x_2 \geq \cdots \geq x_n$. Note that after an insertion of a ball the order may change and bins may need to be renamed. Denote by $\mathbf{x}(t)=(x_1,x_2,\ldots,x_n)$ the vector which specifies the load in each bin after applying Greedy$(\mathcal{D},k)$ t times. Note that by the convention that $\mathbf{x}(t)$ is normalized, bin i at time t is not necessarily bin i at time $t+1$. Denote by $\mathbf{y}(t)$ the normalized load vector which is obtained by applying Greedy$(\mathcal{U},k)$ t times.

Our general approach is to show that for large enough d, the maximum bin in the non-uniform case is stochastically dominated by the maximum bin in the two choice uniform case. Since in Theorem 1.1 it is shown that for $k \geq 2$, the maximum bin obtained by Greedy$(\mathcal{U},k)$ has a load of $\frac{m}{n}+\log\log n/\log k+O(1)$ w.h.p., a stochastic dominance implies Theorem 1.3. We prove the stochastic dominance via a coupling argument. Coupling was used to prove stochastic dominance in the context of balls-and-bins in [2] and [3]. But first we need some more notation.

Denote by $\varphi_i(t)$ the probability at time t GREEDY$(\mathcal{U},k)$ puts a ball in one of the i heaviest bins, where $k \geq 2$. Loyal to our convention that load vectors are normalized we assume the i heaviest bins are $1,2,\ldots,i$. The algorithm puts a ball into a bin in $[1,i]$ iff all k samples are in $[1,i]$ therefore $\varphi_i(t)=(\frac{i}{n})^k$ for all t. Since $\varphi_i(t)$ is the same for all t we abbreviate notation and write φ_i.

Similarly, let $\psi_i(t)$ denote the probability that at time t $\mathsf{Greedy}(\mathcal{D}, d)$ inserts a ball into a bin in $[1, i]$. As before

$$\psi_i(t) = \left(\sum_{j=1}^{i} p_j(t)\right)^d$$

where $p_j(t)$ is the probability the j'th heaviest bin at time t is sampled by $\mathcal{D}$. Note that after each insertion of a ball we sort the bins, so the value of $\psi_i(t)$ indeed may change with t. The probability $\mathsf{Greedy}(\mathcal{D}, d)$ inserts a ball in bin i at time t is therefore $\psi_i(t) - \psi_{i-1}(t)$. Similarly the probability $\mathsf{Greedy}(\mathcal{U}, k)$ puts a ball in bin i is $\varphi_i - \varphi_{i-1}$.

Definition 2.1 (stochastic dominance)**.** *For two random variable X, Y we say that X* stochastically dominates *Y, denoted by $X \succeq Y$ if $\forall \alpha$ it holds that $\Pr[X \leq \alpha] \leq \Pr[Y \leq \alpha]$.*

Informally speaking X dominates Y if for every threshold X is more likely to be larger than the threshold than Y. Now, if we could prove that for every t it holds that $\mathbf{y}(t)_1 \succeq \mathbf{x}(t)_1$ i.e. that the random variable specifying the load of the heaviest bin obtained by $\mathsf{Greedy}(\mathcal{U}, k)$ dominates the load of the heaviest bin obtained by $\mathsf{Greedy}(\mathcal{D}, d)$, then Theorem 1.1 implies the upper bound. The next lemma specifies a condition under which $\mathbf{y}(t)_1 \succeq \mathbf{x}(t)_1$.

Lemma 2.2. *If $\forall i \in [n]$ and for every time $t > 0$ it holds that $\psi_i(t) \leq \varphi_i$ then for every time $t > 0$ it holds that the maximum bin in $\mathsf{Greedy}(\mathcal{D}, d)$ is stochastically dominated by the maximum bin in $\mathsf{Greedy}(\mathcal{U}, k)$ i.e. that $\mathbf{y}(\mathbf{t})_\mathbf{1} \succeq \mathbf{x}(\mathbf{t})_\mathbf{1}$*

Proof. The proof is by a coupling argument. The processes $\big(\mathbf{x}(t)\big)_t$ and $\big(\mathbf{y}(t)\big)_t$ are Markov chains with transition probabilities specified by the allocation rule. A (Markovian) coupling is a Markov chain $\big(\mathbf{x}(t), \mathbf{y}(t)\big)_t$ in which the marginal process $\big(\mathbf{x}(t), \cdot\big)_t$ is an identical copy of $\big(\mathbf{x}(t)\big)_t$ and the marginal process $\big(\cdot, \mathbf{y}(t)\big)_t$ is an identical copy of $\big(\mathbf{y}(t)\big)_t$. The two marginal processes may depend on one another in any arbitrary way.

We need to find a coupling $\big(\mathbf{x}(t), \mathbf{y}(t)\big)_t$ such that on one hand each of the two marginal processes is identical to the respective random process. On the other hand we need to show that at every time t $\mathbf{y}(t)_1 \geq \mathbf{x}(t)_1$. In other words we should find a coupling in which the dependency between $\big(\mathbf{x}(t), \cdot\big)_t$ and $\big(\cdot, \mathbf{y}(t)\big)_t$ is such that $\mathbf{y}(t)_1 \geq \mathbf{x}(t)_1$. Such a coupling immediately implies that the random variable $\mathbf{x}(t)_1$ is dominated by the random variable $\mathbf{y}(t)_1$. We in fact prove a slightly stronger claim. We will show that throughout the coupling $\mathbf{y}(t)$ is *majorized* by $\mathbf{x}(t)$.

Definition 2.3 (majorization)**.** *For two vectors $\boldsymbol{x}, \boldsymbol{y}$ each with n elements we say that $\boldsymbol{y}$ is* majorized *by $\boldsymbol{x}$ written $\boldsymbol{x} \succeq \boldsymbol{y}$ if $\forall j \leq n$ it holds that $\sum_{i=1}^{j} y_i \geq \sum_{i=1}^{j} x_i$.*

We use the following coupling.

1. Pick a number α uniformly at random in $[0, 1)$.
2. Let i be such that $\varphi_{i-1} \leq \alpha < \varphi_i$. Let j be such that $\psi_{j-1}(t) \leq \alpha < \psi_j(t)$.
3. Define $\mathbf{e_i}$ to be the vector with 1 at the i'th location and 0 everywhere else. Set

   $$\big(\mathbf{x}(t+1), \mathbf{y}(t+1)\big)_{t+1} := \big(\mathbf{x}(t) + \mathbf{e_j}, \mathbf{y}(\mathbf{t}) + \mathbf{e_i}\big)$$

   In other words, $\mathbf{y}(t+1)$ is obtained by adding a ball in bin i and sorting. Similarly $\mathbf{x}(t+1)$ is obtained by adding a ball in j and sorting.

It is straightforward to verify that this is indeed a valid coupling: for every t, the probability $\mathsf{Greedy}(\mathcal{U}, k)$ puts a ball in bin i is $\varphi_i - \varphi_{i-1}$. The probability $\mathsf{Greedy}(\mathcal{D}, d)$ puts a ball in bin j is $\psi_j(t) - \psi_{j-1}(t)$.

It remains to show that throughout the coupling $\mathbf{y}(t)$ is majorized by $\mathbf{x}(t)$. We prove this by induction on t. Clearly the claim holds when $t = 0$. Assume that $\mathbf{y}(t-1)$ is majorized by $\mathbf{x}(t-1)$. At time t a ball is put in the i'th bin of $\mathbf{y}(t-1)$ and in the j'th bin of $\mathbf{x}(t-1)$. Since by assumption $\psi_i(t-1) \leq \varphi_i(t-1) \leq \alpha$ it must be that $j \geq i$. The proof is therefore concluded by the following claim:

Claim 2.4. *Let* $\mathbf{x}$ *and* $\mathbf{y}$ *be two normalized integer vectors such that* $\mathbf{x} \succeq \mathbf{y}$ *. If* $i \leq j$ *then* $\mathbf{x} + \mathbf{e_i} \succeq \mathbf{y} + \mathbf{e_j}$ *where* $\mathbf{e_i}$ *is the* i*'th unit vector and* $\mathbf{x} + \mathbf{e_i}$ *and* $\mathbf{y} + \mathbf{e_j}$ *are normalized.*

Proof. Claim 2.4 is similar to Lemma 3.4 in [2]. Lemma 3.4 in [2] states that $\mathbf{x} + \mathbf{e_i} \succeq \mathbf{y} + \mathbf{e_i}$. Trivially it holds that $\mathbf{y} + \mathbf{e_i} \succeq \mathbf{y} + \mathbf{e_j}$, therefore, by the transitivity of the majorization relation we have that $\mathbf{x} + \mathbf{e_i} \succeq \mathbf{y} + \mathbf{e_j}$.

□

This concluded the proof of Lemma 2.2. □

Lemma 2.2 specifies a condition under which Theorem 1.3 could be proven. Namely, all we need to show is that $\psi_i(t) \leq \varphi_i$.

Claim 2.5. *For every* $k > 0$*, if* $d \geq kf(\alpha, \beta)$ *then for every* $i \leq n$ *and* $t \geq 0$ *it holds that* $\psi_i(t) \leq \varphi_i$*.*

Proof. There could be at most $n\frac{\alpha-1}{\alpha\beta-1}$ bins with sampling probability $\frac{\beta}{n}$, otherwise one of the remaining bins would have a sampling probability smaller than $\frac{1}{\alpha n}$. Denote $i^* = n\frac{\alpha-1}{\alpha\beta-1}$. We prove Claim 2.5 by considering separately the case that $i \leq i^*$.

If $i \leq i^*$ then $\psi_i(t) \leq \left(\frac{i\beta}{n}\right)^d$, so it is enough to show that for every $i \leq i^*$ it holds that $\left(\frac{i\beta}{n}\right)^d \leq \left(\frac{i}{n}\right)^k$. First note that since $d \geq kf(\alpha, \beta)$

$$\left(\frac{i^*\beta}{n}\right)^d = \left(\frac{\alpha\beta-\beta}{\alpha\beta-1}\right)^d \leq \left(\frac{\alpha\beta-\beta}{\alpha\beta-1}\right)^{kf(\alpha,\beta)}$$
$$= \left(\frac{\alpha-1}{\alpha\beta-1}\right)^k = \left(\frac{i^*}{n}\right)^k \quad (1)$$

Define the function $G(z) := \frac{\left(\frac{z\beta}{n}\right)^d}{\left(\frac{z}{n}\right)^k} = z^{d-k}\beta^d n^{k-d}$. It is sufficient to show that $G(z) \leq 1$ for any real number $z \in [0, i^*]$. Equation (1) implies that $G(i^*) \leq 1$. It is therefore enough to show that $\frac{G(z)}{\partial z} > 0$ for $0 \leq z \leq i^*$. Indeed, since $d > k$ we have $\frac{G(z)}{\partial z} = (d-k)z^{d-k-1}\beta^d n^{k-d} > 0$.

Now consider the second case when $i \geq i^*$. Now $\psi_i(t) \leq (1 - \frac{n-i}{\alpha n})^d$. Define

$$G(z) := \frac{(1 - \frac{n-z}{\alpha n})^d}{\left(\frac{z}{n}\right)^k}$$

Again, we need to show that $G(z) \leq 1$ for $z \in [i^*, n]$. We know that $G(i^*) \leq 1$ and that $G(n) \leq 1$. We will show that $G(z)$ has at most one local minima in $[i^*, n]$ and that $\frac{G(n)}{\partial z} > 0$. This implies that $G(z) \leq 1$ in the entire range. First we find the points where $\frac{G(z)}{\partial z} = 0$, we have:

$$\frac{G(z)}{\partial z} = \frac{\frac{dz^k}{\alpha n^{k+1}}\left(1 - \frac{n-z}{\alpha n}\right)^{d-1} - \frac{kz^{k-1}}{n^k}\left(1 - \frac{n-z}{\alpha n}\right)^d}{\left(\frac{z}{n}\right)^{2k}} = 0$$

and therefore

$$\frac{dz^k}{\alpha n^{k+1}}\left(1 - \frac{n-z}{\alpha n}\right)^{d-1} - \frac{kz^{k-1}}{n^k}\left(1 - \frac{n-z}{\alpha n}\right)^d = 0$$
$$\frac{dz}{\alpha n} - k\left(1 - \frac{n-z}{\alpha n}\right) = 0 \tag{2}$$

Equation (2) is linear in z and therefore there could be at most one minima or maxima in the range. It remains to show that $\frac{G(n)}{\partial z} > 0$. Indeed, substituting in Equation (2) we see that $\frac{G(n)}{\partial z} > 0$ as long as $d > k\alpha$. The proof is therefore concluded by the following claim:

Claim 2.6. *For every $\alpha > 1$ and $\beta > 1$ it holds that:*

$$\alpha < \frac{\ln(\frac{\alpha\beta-1}{\alpha-1})}{\ln(\frac{\alpha\beta-1}{\alpha\beta-\beta})} < \alpha\beta$$

Proof. The proof of Claim 2.6 is a routine exercise in high-school math. Define $u(\beta) := \ln(\frac{\alpha\beta-1}{\alpha-1})$ and $v(\beta) := \ln(\frac{\alpha\beta-1}{\alpha\beta-\beta})$, so given α we have $f(\alpha, \beta) = \frac{u(\beta)}{v(\beta)}$. First we show that if $\alpha > 1$ then $\alpha < f(\alpha, \beta)$ for $\beta \in (1, \infty]$. Clearly it holds that $\lim_{\beta\to\infty} f(\alpha, \beta) = \infty$. Using L'Hopital's rule we have:

$$\lim_{\beta\to 1} f(\alpha, \beta) = \lim_{\beta\to 1} \frac{u'(\beta)}{v'(\beta)} = \alpha$$

It is therefore enough to show that the derivative $\frac{f(\alpha,\beta)}{\partial\beta}$ is positive. We have $\frac{f(\alpha,\beta)}{\partial\beta} < 0$ iff $f(\alpha, \beta) < \alpha$. Now since $\lim_{\beta\to\infty} f(\alpha, \beta) = \infty$ if there had been a value for β for which $f(\alpha, \beta) < \alpha$ then there must be a value of β for which on one hand $f(\alpha, \beta) < \alpha$ and on the other hand $\frac{f(\alpha,\beta)}{\partial\beta} > 0$ which implies, in contradiction that $f(\alpha, \beta) > \alpha$. We conclude that in the range $\beta \in (1, \infty]$ it must be that $f(\alpha, \beta) > \alpha$. Similar arguments show that $f(\alpha, \beta) < \alpha\beta$. □

This concludes the proof of Claim 2.5 □

Now, since $k \geq 2$, Claim 2.5 and Lemma 2.2 imply the upper bound of Theorem 1.3 for the case $d \geq 2f(\alpha, \beta)$.

2.2 A Tight Upper Bound

In this section we sketch how the upper bound in Theorem 1.3 could be sharpened to the case $d \geq (1+\epsilon)\cdot f(\alpha, \beta)$. In order to do that we define a new algorithm Greedy($\mathcal{U}, 1+\epsilon$). The algorithm Greedy($\mathcal{U}, 1+\epsilon$) inserts m balls sequentially where each round the probability a ball is inserted to

one of the ith most heavy bins is exactly $\left(\frac{i}{n}\right)^{1+\epsilon}$. Note that since $1+\epsilon$ is not an integer there is no immediate way to implement Greedy($\mathcal{U}, 1+\epsilon$) as a 1-out-of-d algorithm. Rather, the algorithm needs to take into account the load on *all* bins in order to sample the bin which receives the ball. Thus, Greedy($\mathcal{U}, 1+\epsilon$) is a not a good algorithm to run in practice. It is used as a tool to prove the effectiveness of Greedy($\mathcal{D}, d$). The idea is to show on one hand that Greedy($\mathcal{U}, 1+\epsilon$) produces a balanced allocation and on the other hand that Greedy($\mathcal{U}, 1+\epsilon$) is majorized by Greedy($\mathcal{D}, d$) . The majorization part is straightforward. Note that in the proof of Lemma 2.2 and Claim 2.5 we did not use the assumption that $d \geq 2$ and in fact the proof holds for any positive d. The only place the integrality of d was assumed was in Theorem 1.1. Thus, in order to sharpen the upper bound it suffices to prove the following lemma.

Lemma 2.7. *For every $\epsilon > 0$, after throwing m balls into n bins using Greedy($\mathcal{U}, 1+\epsilon$) , the most loaded bin has $m/n + \log\log n/\log(1+\epsilon) + O(1)$ balls with probability $1 - o(1/n)$.*

Sketch: The idea of the proof is the observation that the proof of Theorem 1.1 in [3] doesn't in fact require that d be an integer and carries on without modifications as long as $d > 1$. The proof in [3] is long and technical, in this sketch we will not repeat it but rather point out the main techniques and demonstrate why it doesn't depend on the integrality of d. The proof of Theorem 1.1 relies on two main lemmas.

The first lemma proves Theorem 1.1 for the case where the number of balls is polynomial in the number of bins. This is done via a layered induction technique, which is a generalization of the technique used in [2]. While the proof in [3] is very technical, it relies on the techniques of [2], for which it is fairly simple to see that the proof doesn't require d to be an integer: Let b_i be an upper bound on the fraction of bins which have i balls or more throughout the process. The observation that lies in the heart of the proof in [2] is that in order for a ball to land in height $i+1$ or more, all d choices must be of height i or more, therefore the probability a ball is of height $i+1$ or more is at most b_i^d. Note that this is true even if d is not an integer; i.e. the probability Greedy($\mathcal{U}, 1+\epsilon$) puts a ball in height $\geq i+1$ is at most $b_i^{1+\epsilon}$. Now, given that there are n balls we have that the expected number of balls of height $\geq i+1$ is at most nb_i^d. The number of bins of load $\geq i+1$ is at most the number of balls of height $\geq i+1$, therefore $\mathbb{E}[b_{i+1}] \leq b_i^d$. Now if we set $b_3 = \frac{1}{3}$ we have that $b_{\log_d \log n} < 1$. We ignore in this sketch many technical issues, but they do not affect the main observation.

The second lemma is the main tool in the reduction from an arbitrary large m to $m < poly(n)$. The reduction is done via an iterative process of sharpening a week upper bound. Before we discuss the second lemma we note that the initial weak bound showing that w.h.p the max load is at most $\frac{m}{n} + \sqrt{\frac{m\log n}{n}}$ is true for the single process algorithm, and therefore by a straightforward majorization argument holds for Greedy($\mathcal{U}, 1+\epsilon$) as well.

The main lemma used in the reduction is a *short memory* lemma, showing that if the current configuration has a $\max - \min$ gap of Δ then after throwing $\Delta poly(n)$ more balls the original configuration is "forgotten". The short memory lemma is proven via a coupling argument showing the rapid mixing of the Markov process which underlies the multiple choice paradigm. The sole property the coupling proof relies on is that the allocation process is *biased* towards the light bins, where this bias is polynomial; i.e. $\Pr[\text{ball falls in bin } i+1] \geq \Pr[\text{balls fall in bin } i] + 1/poly(n)$. This property clearly holds with Greedy($\mathcal{U}, 1+\epsilon$) :

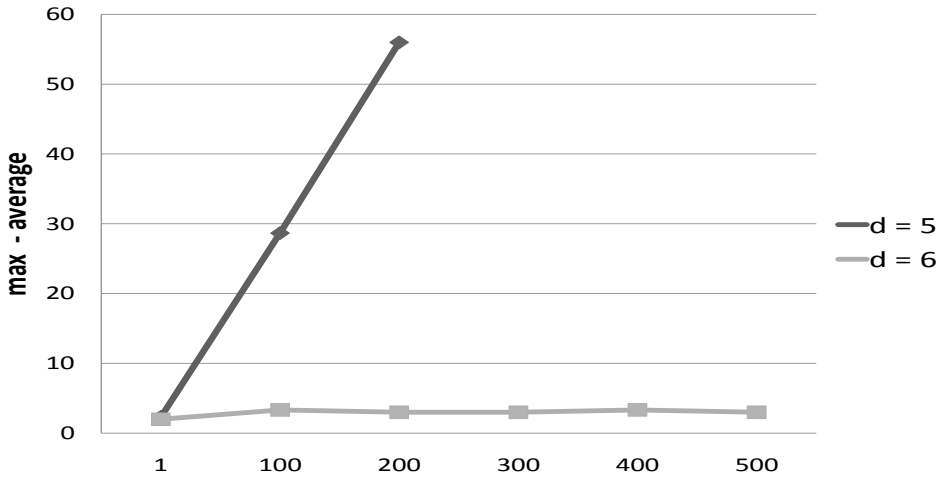

Figure 1: The max-avg gap as a function of the number of balls per bin, with 500 bins and $f(\alpha, \beta) = 5.78$.

Let ℓ_i be the probability the algorithm inserts a ball in bin i, so $\ell_i = \frac{i^{1+\epsilon} - (i-1)^{1+\epsilon}}{n^{1+\epsilon}}$ and

$$\ell_i - \ell_{i-1} \geq \left(\frac{2}{n}\right)^{1+\epsilon}$$

We conclude that the proof of Theorem 1.1 in [3] in fact implies Lemma 2.7 as well and refer to [3] for more details. □

3 Applications of the Result

Assume a distributed storage system is composed of n servers and say we have a hashing scheme that samples each server with equal probability. Say that once in a while the capacities of servers in the market increase by a factor of 5, in which case half of the old servers are replaced by the stronger servers. Theorem 1.3 tells us exactly how many choices should be used in order for the two choice paradigm to be effective. If there are n new servers and n old servers and each new server should simulate 5 servers then all in all there are $6n$ servers. An old server is sampled with probability $\frac{1}{2n} = \frac{3}{6n}$ so $\beta = 3$. The probability a new server is sampled is also $\frac{1}{2n}$. But each new server simulated 5 servers, so the probability a simulated server is sampled is $\frac{1}{10n} = \frac{1}{5/3 \cdot 6n}$ so $\alpha = 5/3$. Now we have that $f(\alpha, \beta) \approx 2.58$. Theorem 1.3 implies that if GREEDY$(\mathcal{U}, 3)$ is used then we can have an allocation in which the new servers have 5 times more data than old ones. If GREEDY$(\mathcal{U}, 2)$ is used, then the old servers will hold more than $1/6$ of the data items and the allocation will not accommodate the heterogeneous nature of the system. Note that the number of

choices is affected not only by the gap in capacity, but also by the decision policy to replace half of the servers each time. If less than half of the servers are replaced then more choices are needed. We conclude that in terms of load balancing, it is beneficial to replace as many servers as possible.

In Figure 3 we plot the result of a simulation done with $n = 500, \alpha = 3, \beta = 5$. So $f(\alpha, \beta) = 5.78$. The horizontal axes measures the number of balls per bin and the vertical axes measures the gap between the maximum bin and the average. Indeed, as Theorem 1.3 predicts, when $d = 5$ the gap between the most loaded bin and the average bin increases as the number of balls increases. With 200 balls per bin the maximum bin has 256 balls, with 500 balls per bin the maximum bin has 638 balls . When the number of choices was increased by one to be $d = 6$, the gap between maximum and average remained 3 even when the number of balls per bin is 1000. The purpose of this simulation is to demonstrate that a poor choice of d may result in a strikingly unbalanced allocation. The conclusion system designers should take from this note is that if the multiple choice paradigm is to be used in a heterogenous environment the number of choices d is a crucial parameter which depends on the amount of heterogeneity of the system.

Acknowledgments

The author is deeply indebted to John MacCormick, Nick Murphy, Rama Ramasubramanian, Lidong Zhou and Kunal Talwar.

References

[1] Micah Adler, Soumen Chakrabarti, Michael Mitzenmacher, and Lars Rasmussen. Parallel randomized load balancing. In *Proc. of Symp. on Theory of Computing (STOC)*, pages 238–247, 1995.

[2] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM Journal Computing*, 29(1):180–200, 1999.

[3] Petra Berenbrink, Artur Czumaj, Angelika Steger, and Berthold Vöcking. Balanced allocations: the heavily loaded case. *Siam Journal Computing*, 35(6):1350–1385, 2006.

[4] John Byers, Jeffrey Considine, and Michael Mitzenmacher. Simple load balancing for distributed hash tables. In *Proc. of Intl. Workshop on Peer-to-Peer System (IPTPS)*, pages 80–87, 2003.

[5] John W. Byers, Jeffrey Considine, and Michael Mitzenmacher. Geometric generalizations of the power of two choices. In *Proc. of Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 54–63, New York, 2004.

[6] David Karger, Eric Lehman, F. Thomson Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Symp. on Theory of Computing (STOC)*, pages 654–663, 1997.

[7] David R. Karger and Matthias Ruhl. Simple efficient load balancing algorithms for peer-to-peer systems. In *Proc. of Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 36–43, 2004.

[8] Richard M. Karp, Michael Luby, and Friedhelm Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proc. 24th Symp. on Theory of Computing (STOC)*, pages 318–326, 1992.

[9] Witold Litwin. Linear hashing: A new tool for file and table addressing. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, pages 212–223, 1980.

[10] John MacCormick, Nick Murphy, Venugopalan Ramasubramanian, Udi Wieder, Jinfeng Yang, and Lidong Zhou. Kinesis: The power of controlled freedom in distributed storage systems. In *manuscript*.

[11] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proc. 21st ACM Symp. on Principles of Distributed Computing, (PODC)*, pages 183–192, 2002.

[12] Gurmeet S. Manku. Balanced binary trees for id management and load balance in distributed hash tables. In *Proc. 23rd Symp. on Principles of Distributed Computing, (PODC)*, pages 197–205, 2004.

[13] Michael Mitzenmacher, Andrèa Richa, and Ramesh Sitaraman. *Handbook of Randomized Computing*, chapter The power of two random choices: A survey of the techniques and results. Kluwer, 2000.

[14] Moni Naor and Udi Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *Symp. on Parallelism in Algorithms and Architectures (SPAA)*, pages 50–59, 2003.

[15] Martin Raab and Angelika Stege. Balls into bins - a simple and tight analysis. In *RANDOM*, pages 159–170, 1998.

[16] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[17] Kunal Talwar and Udi Wieder. Balanced allocaitons: The weighted case. In *Proc. of Symp. on Thoery of Computing (STOC)*, 2007.

[18] B. Vöcking. How asymmetry helps load balancing. In *Proc. of Symp. on Foundations of Computer Science (FOCS)*, pages 131–141, 1999.