38*

Michael A. Bender† Jake Christensen† Alex Conway‡ Martin Farach-Colton‡
Rob Johnson§ Meng-Tsung Tsai¶

Abstract

Balls-and-bins games have been a successful tool for modeling load balancing problems. In this paper, we study a new scenario, which we call the ***ball-recycling game***, defined as follows:

> Throw m balls into n bins i.i.d. according to a given probability distribution $\mathbf{p}$. Then, at each time step, pick a non-empty bin and ***recycle*** its balls: take the balls from the selected bin and re-throw them according to $\mathbf{p}$.

This balls-and-bins game closely models memory-access heuristics in databases. The goal is to have a bin-picking method that maximizes the ***recycling rate***, defined to be the expected number of balls recycled per step in the stationary distribution.

We study two natural strategies for ball recycling: Fullest Bin, which greedily picks the bin with the maximum number of balls, and Random Ball, which picks a ball at random and recycles its bin. We show that for general $\mathbf{p}$, Random Ball is $\Theta(1)$-optimal, whereas Fullest Bin can be pessimal. However, when $\mathbf{p} = \mathbf{u}$, the uniform distribution, Fullest Bin is optimal to within an additive constant.

1 Introduction

Balls-and-bins games have been a successful tool for modeling load balancing problems [1–3, 10, 16, 17, 19, 20, 23, 30–33, 39, 45, 46]. For example, they can be used to study the average and worst-case occupancy of buckets in a hash table [8], the worst-case load on nodes in a distributed cluster [9, 40] and even the amount of time customers wait in line at the grocery store [31]. In all these load-balancing problems, balls-and-bins games are used to study how to distribute load evenly across the resource being allocated.

In this paper we study a new scenario, which we refer to as the ***ball-recycling game***, defined as follows:

> Throw m balls into n bins i.i.d. according to a given probability distribution $\mathbf{p}$. Then, at each time step, pick a non-empty bin and ***recycle*** its balls: take the balls from the selected bin and re-throw them according to $\mathbf{p}$.

We call a bin-picking method a ***recycling strategy*** and define its ***recycling rate*** to be the expected number of balls recycled in the stationary distribution (when it exists).

The ball-recycling game models ***insertion buffers*** and ***update buffers***, which are widely used to speed up insertions in databases by batching updates to blocks on disk. The recycling rate of a recycling strategy corresponds to the speed-up obtained by an insertion buffer, so the goal studied in this paper is how to maximize the recycling rate. This relationship is described in Section 2, and the experiments in Section 6 demonstrate that it holds in practice.

In this paper, we present results for ball recycling for both general $\mathbf{p}$ and for the special case of uniform $\mathbf{p}$, which we denote by $\mathbf{u}$. As we explain in Section 2, these distributions correspond to update and insertion buffers, respectively.

We focus on three natural recycling strategies:

- Fullest Bin: A greedy strategy that recycles the bin with the most balls.
- Random Ball: A strategy that picks a ball uniformly at random and recycles its bin.
- Golden Gate: A strategy that picks the bins in round-robin fashion; after a bin is picked, the next bin picked is its non-empty successor.

Let $\|\mathbf{p}\|_{\frac{1}{2}} = (\sum \sqrt{\mathbf{p}_i})^2$ be the half quasi-norm of $\mathbf{p}$. We achieve the following result for general $\mathbf{p}$.

*This research was supported in part by NSF grants CCF 1439084, CCF 1637458, CNS 1408695, CNS 1755615, CNS 1763680, CNS 1408782, IIS 1247726, IIS 1251137, CCF 1617618, CCF 1725543, CCF-BSF 1716252, and IIS 1541613, NIH grant 1U01CA198952-01, MOST grant 107-2218-E-009-026-MY3, and by EMC, Inc, NetAPP, Inc, and Sandia National Labs.

†Stony Brook University, Stony Brook, NY 11794-2424 USA. Email: {bender,jpchristense}@cs.stonybrook.edu.

‡Rutgers University, Piscataway NJ 08855 USA. Email: {ajc179,farach}@cs.rutgers.edu.

§VMware Research, Creekside F, 3425 Hillview Ave, Palo Alto, CA 94304 USA. Email: robj@vmware.com.

¶National Chiao Tung University, Hsinchu, Taiwan. Email: mtsai@cs.nctu.edu.tw.

THEOREM 1.1. (SECTION 4) *Consider a ball-recycling game with m balls and n bins, where the balls are distributed into the bins i.i.d. according to distribution* $\mathbf{p}$. *Then* RANDOM BALL *is* $\Theta(1)$*-optimal.*

It achieves recycling rate $\mathcal{R}^{RB}$*:*

1. *If* $m \geq n$,
$$\mathcal{R}^{RB} = \Theta\left(\frac{m}{\|\mathbf{p}\|_{\frac{1}{2}}}\right).$$
2. *If* $m < n$, *let* L *be the* m *lowest-weight bins,* $q = \sum_{\ell \in L} p_\ell$, *and* $\mathcal{R}^{RB}_L$ *be the recycle rate of* RANDOM BALL *restricted to* L. *Then,*
$$\mathcal{R}^{RB} = \Theta\left(\min\left(\mathcal{R}^{RB}_L, 1/q\right)\right).$$

In order to establish this result, we first show that no recycling strategy can achieve a higher recycling rate than $(2m + n)/\|\mathbf{p}\|_{\frac{1}{2}}$. This directly establishes optimality when $m = \Omega(n)$. For $m = o(n)$, we show that RANDOM BALL performs as well as another strategy, AGGRESSIVE EMPTY, which takes an optimal strategy on a subset of high-weight bins and turns it into an optimal strategy on all the bins.

Interestingly, the greedy strategy FULLEST BIN is not generally optimal, and in particular:

Observation. There are distributions for which FULLEST BIN is pessimal, that is, it recycles at most 2 balls per round whereas OPT recycles almost m balls per round.

For example, consider the ***skyscraper distribution***, where $\mathbf{p}_0 = 1 - 1/n + 1/n^2$ and $\mathbf{p}_i = 1/n^2$, for $0 < i \leq n - 1$. Suppose that $m = \sqrt{n}$. Then FULLEST BIN will pick bin 0 every time until it has at most one ball, at which point it will pick another bin, which will almost certainly have 1 ball in it. Thus, the recycling rate of FULLEST BIN drops below 2. Suppose, instead, that we recycle the least-full non-empty bin. In this case, every approximately $\sqrt{n}$ rounds, a ball lands in a low-probability bin and is promptly returned to bin 0. Thus, the recycling rate of this strategy is nearly m. Thus, FULLEST BIN is pessimal for this distribution.

However, the uniform distribution is of particular importance to insertion buffers for databases based on B-trees. This is because (arbitrary) random B-tree insertions are nearly uniformly distributed across the leaves of the B-tree, as we show in Section 2. On the uniform distribution, FULLEST BIN and GOLDEN GATE are optimal even up to lower order terms:

THEOREM 1.2. (SECTION 5) FULLEST BIN *and* GOLDEN GATE *are optimal to within an additive constant for the ball-recycling game with distribution* $\mathbf{u}$ *for any* n *and* m. *They each achieve a recycling rate of at least* $2m/(n + 1)$, *whereas no recycling strategy can achieve a recycling rate greater than* $2m/n + 1$.

In this case, RANDOM BALL is only optimal to within a multiplicative constant in the following range:

THEOREM 1.3. (SECTION 5) *On the uniform distribution* $\mathbf{u}$, *for sufficiently large* m, RANDOM BALL *is at least* $(1/2+1/(2^3 3^4))$*-optimal and at most* $(1-3/1000)$*-optimal.*

Thus, we establish some surprising results: that FULLEST BIN can perform poorly for arbitrary $\mathbf{p}$ but is optimal for $\mathbf{u}$, up to lower-order terms; and that RANDOM BALL is asymptotically optimal for any $\mathbf{p}$ and in particular is more than 1/2-optimal but not quite optimal for $\mathbf{u}$.

In Section 6, we present experimental results showing that our analytical results for the ball recycling problem closely matches performance results in real databases. We describe the recycling strategies of several commercial and open-source database systems. In particular we focus on InnoDB, a B-tree that uses a variant of RANDOM BALL.

Our results suggest that FULLEST BIN or GOLDEN GATE would be a better choice than RANDOM BALL for InnoDB. In particular, GOLDEN GATE requires almost no additional bookkeeping, and can be implemented in InnoDB with a change of only a few lines of code. With this implementation, we measured a 30% improvement in its insertion-buffer flushing rate, which is in line with our theoretical results.

We conclude that ball recycling is a natural hitherto unexplored balls-and-bins game that closely models a widely deployed method for improving the performance of databases. Moreover, this is the first application (to our knowledge) of a balls-and-bins game to the throughput of a system. This is in contrast to past balls-and-bins analyses, which modeled load balancing and latency.

2 Ball Recycling and Insertion/Update Buffers

Ball recycling models insertion and update buffers, which are widely used in modern databases [4, 7, 14, 15, 25, 26, 34, 35, 41, 44, 47]. These implementations are discussed in more detail in Section 6.

For a key-value store, such as a database, an ***insertion buffer*** is a cache of recently inserted items. When the insertion buffer fills, the database selects a disk block and all the cached items going to that block are evicted in bulk. If k elements are flushed in bulk, then there is a speedup of k, compared to writing the

elements to the destination block as soon as they arrive. After evicting k items, there is room for k new elements in the buffer. An ***update buffer*** caches changes to existing key-value pairs but is otherwise like an insertion buffer. Although these types of buffers seem quite similar, we show that they have important differences in how they are modeled as a ball-recycling game.

The mapping to ball recycling is direct: disk blocks are bins and elements in the insertion/update buffer are balls. The probability distribution $\mathbf{p}$ is based on the distribution of items inserted or updated. Evicting all the items going to a disk block corresponds to emptying the bin associated with that disk block. After an eviction of k items, we have room for k new insertions/updates, i.e., we have k new balls to throw. The policy for selecting the target disk block of an eviction corresponds to the policy for selecting a bin to recycle, and the speedup induced by an eviction policy is its recycling rate.

For B-trees, insertion buffers and update buffers differ in an important way: updates do not change the structure of leaves of a B-tree. In contrast, insertions can change the range of keys associated with a leaf (due to leaf splits), which yields the following result:

LEMMA 2.1. *If N keys are inserted into a B-tree i.i.d. according to some key distribution $\mathbf{q}$, then provided $B = \Omega(\log N)$, the maximum probability that a leaf has of receiving the next insertion is $O(B/N)$ with probability 1. Thus the corresponding recycling game is asymptotically* **almost uniform***: no bin has probability more than a constant multiple of $\frac{1}{n}$.*

We prove the uniformity bound as follows. Let $F(\kappa)$ be the cumulative density function (CDF) of $\mathbf{q}$, which is the probability that an item sampled from $\mathbf{q}$ is less than κ. If κ is distributed according to $\mathbf{q}$, then $F(\kappa)$ will be uniformly distributed on $[0, 1]$.

If n points are sampled from $[0, 1]$ uniformly and sorted so that $x_1 \le x_2 \le \cdots \le x_n$, then $\max x_{i+B} - x_i$ is known as the ***maximal B-spacing***. It follows that:

LEMMA 2.2. *Having inserted n keys into a B-tree i.i.d. according to a distribution $\mathbf{q}$, the maximum probability that any leaf has of receiving the next insertion is less than the maximum B-spacing of the CDFs of those points.*

It is known that:

LEMMA 2.3. ([21]) *If $B = \Theta(\log n)$, then the maximum B-spacing of n points distributed uniformly on the unit interval is $\Theta(B/n)$ with probability 1.*

For $B = \omega(\log n)$, we can subdivide B into intervals of $\log n$ points, each of which will satisfy the lemma. Adding together the resulting bounds, we have that the maximal B-spacing of n points is $O(B/n)$ with probability 1. Together with Lemma 2.2, this implies Lemma 2.1.

We also note that an (almost-uniform) ball-recycling game is an imperfect model for an insertion buffer, because the ball-recycling game has a fixed number of bins, whereas in the insertion buffer, the number of disk blocks will increase. Furthermore, insertions may not be independently distributed.

Finally, the implementation of these strategies is a point of departure between insertion/update buffers and ball recycling. In ball recycling, it is obvious which bin each ball is in. In insertion/update buffers for a B-tree, elements have a key, but we don't necessarily know what the buckets are, since the mapping from keys to buckets depends on the pivots used to define the B-tree leaves. FULLEST BIN needs to know what the buckets are, whereas RANDOM BALL and GOLDEN GATE do not. For RANDOM BALL this is because the key of the randomly selected item can be used to fetch its target B-tree leaf, after which we know the max and min keys in that leaf, and GOLDEN GATE can be implemented by remembering the upper bound of the last leaf to which we flushed and then flushing the item with the successor of that key, along with all the other keys going to that leaf. None of these strategies require knowledge of $\mathbf{p}$.

Our results on general $\mathbf{p}$ have an interesting implication for B^ε-trees, which are known to be asymptotically optimal for insertions, in the worst case. B^ε-trees can also handle updates by propagating messages to the leaves. For some update distributions, flushing according to RANDOM BALL can achieve an update rate that is B^ε faster than FULLEST BIN. We expect to try RANDOM BALL flushing in our B^ε-tree-based file system, BεtrFS [18, 22, 27, 28, 48, 49].

3 Ball Recycling and Markov Theory

We begin our analysis of ball-recycing games with some preliminary results. In particular, we show that all finite-state ball-recycling strategies have stationary distributions.

3.1 Ball-recycling games are Markov decision processes.

This section makes use of the standard theory of Markov chains and Markov decision processes; for an introduction see *e.g.* Kallenberg [29].

In a ball-recycling game, we represent the configuration of the balls as a vector $X = (X_i)$ of length n, where X_i is the number of balls in the ith bin. Since the number of balls is finite, there are only a finite number of bin configurations.

A recycling strategy A takes as input the current

bin configuration X together with an internal state S, and selects a non-empty bin to recycle; the next state is obtained by removing all the balls from the selected bin and re-throwing them according to $\mathbf{p}$. The bin selection may be randomized. We write A^iX for the state obtained after i rounds of recycling using strategy A. In each round, the recycling algorithm earns a reward equal to the number of balls recycled in that round.

In this way, the ball-recycling game is a Markov decision process, and we are interested in policies that maximize the expected average recycling rate, defined for a policy A as

$$\mathcal{R}^A = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T} R(A^t X_0).$$

Note that Markov decision processes are very general. For example, in a Markov decision process the policy may vary over time, and may even take the entire history of the process and its own past decisions into account when deciding on its next action. Thus, for some strategies A, the limit $\mathcal{R}^A$ may not exist. In the literature of Markov decision processes, this is often handled by taking the lim sup instead of the limit. However, for any Markov decision process, any strategy that maximizes the limit also maximizes the lim sup [29]. Therefore, for simplicity, we will focus only on strategies for which the limit is well-defined, and the results will generally hold for the lim sup of arbitrary strategies.

A Markov decision process policy is ***deterministic*** if it decides on its next action based solely on the current state, *i.e.* without looking at history, the number of time steps that have passed or by flipping random coins. A deterministic policy can be represented as a simple table mapping each state to a single action to be taken whenever the system is in that state.

The specific strategies we analyze are ***finite-state*** strategies, where the internal state has only finitely many configurations. When we prove our lower bounds, we will further restrict ourselves to ***stateless*** strategies, where there is a unique internal state. In order to do so, we make use of the following lemma.

LEMMA 3.1. ([29]) *There exists a stateless deterministic recycling strategy* OPT *that achieves the optimal expected average recycling rate.*

Proof. This follows from Kallenberg's Corollary 5.4.

3.2 Stationary distributions of recycling strategies A ball-recycling game and a recycling strategy together define a Markov process on the state space; the space of pairs comprising a balls-and-bins configuration and an internal state. If the strategy is stateless, this is a Markov process on the balls and bins space.

There are a finite number of balls-and-bins configurations. Therefore, a ball-recycling game with a finite-state recycling strategy defines a finite Markov process, and so has at least one stationary distribution.

We now show that stateless recycling strategies result in Markov processes with unique stationary distributions. The following lemma shows that, when we look only at the bin configurations, recycling games have properties analogous to irreducibility and aperiodicity in Markov chains.

LEMMA 3.2. *For any ball-recycling game with m balls and n bins there is an $\epsilon > 0$ such that, for all bin configurations X and Y, and for all recycling strategies, the probability that X reaches Y within $\min(m, n)$ steps is at least ϵ. Furthermore, every bin configuration can transition to itself in one time step.*

Proof. We just need to show a sequence of outcomes for ball tosses that transform X into Y, no matter which bins the recycling strategy chooses to empty. So, imagine that, at each step, all the recycling balls land in occupied bins, so that at each step, the number of occupied bins goes down by 1. After at most $\min(m, n) - 1$ steps, all the balls will be in a single bin. On the next round, the recycling strategy must choose that bin, causing all the balls to be rethrown. There is some non-zero probability that they land in configuration Y.

For the second observation, simply note that all the recycled balls may happen to land in the bin from whence they came.

LEMMA 3.3.

1. *All ball-recycling games using stateless recycling strategies have unique stationary distributions which are equal to their limiting distributions.*

Proof. Having fixed a stateless recycling strategy, the ball-recycling game is a Markov process on the balls-and-bins configuration. By Lemma 3.2, this process is irreducible and aperiodic, and so has a unique stationary distribution equal to its limiting distribution.

Together with Lemma 3.1, we have:

COROLLARY 3.1. *For any ball-recycling game, there exists an optimal strategy with a unique stationary distribution.*

We now show that two of the three main recycling strategies studied in this paper yield unique stationary distributions. The strategies are:

- FULLEST BIN: selects the bin that has the most balls;
- RANDOM BALL: selects a ball uniformly at random and recycles whichever bin it is in;
- GOLDEN GATE: selects the bins in a round robin sequence.

FULLEST BIN is a deterministic strategy, RANDOM BALL is a stateless strategy and GOLDEN GATE is a finite-state strategy. By Lemma 3.3 we have:

LEMMA 3.4. FULLEST BIN *and* RANDOM BALL *have unique stationary distributions.*

4 Random Ball is Optimal

In this section we prove Theorem 1.1, showing that RANDOM BALL is $\Theta(1)$-optimal.

4.1 Outline of Proof We prove Theorem 1.1 in the following steps:

1. No recycling strategy has a recycling rate exceeding $(2m+n-1)/\|\mathbf{p}\|_{\frac{1}{2}}$. (Section 4.2)
2. RANDOM BALL matches that bound when $m \geq n$, leading to case (1) of Theorem 1.1. (Section 4.3)
3. There is an $\Theta(1)$-optimal strategy, AGGRESSIVE EMPTY, when $m < n$. The recycling rate of AGGRESSIVE EMPTY matches case (2) of Theorem 1.1. (Section 4.4)
4. By comparison to AGGRESSIVE EMPTY, RANDOM BALL is $\Theta(1)$-optimal when $m < n$. (Section 4.5)

4.2 The Upper Bound We begin by proving an important lemma that will be used throughout, which we refer to as the ***flow equation***. Then we proceed to prove the upper bound.

Let A be a stateless strategy with stationary distribution $\chi^{A,\mathbf{p}}$. Let ϕ_i be the event that A picks bin i to recycle. Let $\mathcal{R}_i^A = E[R^A(\chi^{A,\mathbf{p}})|\phi_i]$ be the number of balls recycled given that the strategy picks bin i, and $f_i = P(\phi_j)$, the probability of picking the bin i in the stationary distribution. We note that $\mathcal{R}_i^A$ and f_i could alternatively be defined as limits of repeated applications of A to any given starting state.

For a given bin i, we can analyze the "flow" of balls into and out of i. When k balls are thrown, $\mathbf{p}_i k$ of them are expected to land in i. For a ball to leave i, i must first be picked to be emptied by A, at which point every ball in i will be evicted. In the stationary distribution, the net flow must be zero. We can generalize to any set of bins and get:

LEMMA 4.1. *Let A be a statelss strategy for a ball-recycling game with n bins with probabilities $\mathbf{p}$. If L is a subset of the bins, $\mathbf{p}_L = \sum_{\ell \in L} \mathbf{p}_\ell$, $f_L = \sum_{\ell \in L} f_\ell$ and $\mathcal{R}_L^A$ the conditional recycle rate given A picks a bin in L, then*

$$\mathbf{p}_L \mathcal{R}^A = f_L \mathcal{R}_L^A. \tag{4.1}$$

We will mostly use the following special case of the Lemma 4.1:

LEMMA 4.2. (THE FLOW EQUATION) *Let A be a statless recycling strategy for a ball-recycling game with n bins with probabilities $\mathbf{p}$. Then, for all $0 \leq i < n$,*

$$\mathbf{p}_i \mathcal{R}^A = f_i \mathcal{R}_i^A. \tag{4.2}$$

We now describe the main upper bound on the recycling rate of any recycling strategy. In order to understand the intuition behind Lemma 4.3, consider a given bin i. Intuitively, it makes sense to think that for a reasonable recycling strategy the recycling rate of the other bins in the system will go down as the number of balls X_i in bin i grows. After all, the X_i balls in bin i aren't available for recycling, until bin i is selected. If we assume this intuition as fact for the moment, this suggests that the expected number of balls in bin i should be greater than half the recycling rate of bin i, perhaps excluding the last ball to land in the bin.

By the Flow Equation, this would suggest that

$$\mathrm{E}[X_i] \geq \frac{1}{2}(\mathcal{R}_i^A - 1) = \frac{1}{2}\left(\frac{\mathbf{p}_i}{f_i}\mathcal{R}^A - 1\right),$$

so that after summing over i, we obtain Lemma 4.3.

However, the following strategy does not satisfy this assumption: for a given bin i, have the strategy just pick the least full non-empty bins until i has a few balls, then pick the fullest ones, then pick i and repeat. Showing that better strategies do not do this is non-trivial, and we prove Lemma 4.3 by different means.

LEMMA 4.3. *Consider a ball-recycling game with m balls, n bins and distribution $\mathbf{p}$. If A is a stateless strategy that picks bin i with frequency f_i, then its recycle rate is bounded by*

$$\mathcal{R}^A \leq \frac{2m+n-1}{\sum_j \frac{\mathbf{p}_j}{f_j}}. \tag{4.3}$$

Given a strategy A, the idea of the proof is to use the invariance of the statistic

$$Z(X) = \sum_{j=1}^{n} \frac{X_j^2}{\mathbf{p}_j}, \tag{4.4}$$

under the action of A on its stationary distribution. The application of A to Z together with the flow equation creates a factor of $\sum_j \mathcal{R}_j$, which when solved for proves the bound. First, we begin with some foundational lemmas, and then proceed to prove the main results.

LEMMA 4.4. *Suppose k balls are thrown into n bins i.i.d. according to distribution $\mathbf{p}$. Let $B(j,k)$ be the binomial random variable denoting how many balls land in the jth bin. The following hold:*

$$\mathrm{E}\left[B(j,k)\right] = \mathbf{p}_j k \tag{4.5}$$

$$\mathrm{E}\left[(B(j,k))^2\right] = \mathbf{p}_j(1-\mathbf{p}_j)k + \mathbf{p}_j^2 k^2 \tag{4.6}$$

Proof. $B(j,k)$ is a binomial random variable with parameters $\mathbf{p}_j$ and k.

Next, given a state X, we compute the effect of recycling the ℓth bin on the jth component of Z. Note that if A recycles bin ℓ of state X, then $X_\ell = R^A(X)$.

LEMMA 4.5. *In a ball-recycling game with m balls, n bins and probability distribution $\mathbf{p}$, if a strategy A recycles bin ℓ in state X, then for $j \neq \ell$,*

$$\begin{aligned}\mathrm{E}\left[(AX)_j^2\right] = X_j^2 + 2X_j\mathbf{p}_j R^A(X) \\ + \mathbf{p}_j(1-\mathbf{p}_j)R^A(X) + \mathbf{p}_j^2 R^A(X)^2,\end{aligned} \tag{4.7}$$

where $R^A(X) = X_\ell$ is the number of balls recycled.

Proof.

$$\begin{aligned}\mathrm{E}\left[(AX)_j^2\right] &= \mathrm{E}\left[(X_j + B(j, R^A(X)))^2\right] \\ &= X_j^2 + 2X_j\mathrm{E}\left[B(j,R^A(X))\right] \\ &\qquad + \mathrm{E}\left[B(j,R^A(X))^2\right] \\ &= X_j^2 + 2X_j\mathbf{p}_j R^A(X) \\ &\quad + \mathbf{p}_j(1-\mathbf{p}_j)R^A(X) + \mathbf{p}_j^2\left(R^A(X)\right)^2\end{aligned}$$

We now can use this result to compute the result of applying A to Z.

LEMMA 4.6. *In a ball-recycling game with m balls, n bins and probability distribution $\mathbf{p}$, if a strategy A recycles bin ℓ in state X, then*

$$\begin{aligned}\mathrm{E}\left[Z(AX)\right] = Z(X) - \left(1 + \frac{1}{\mathbf{p}_\ell}\right)\left(R^A(X)\right)^2 \\ + (2m+n-1)R^A(X)\end{aligned} \tag{4.8}$$

Proof.

$$\begin{aligned}\mathrm{E}\left[Z(AX)\right] &= \sum_{j=1}^n \mathrm{E}\left[(AX)_j^2/\mathbf{p}_j\right] \\ &= \frac{\mathrm{E}\left[(AX)_\ell^2\right]}{\mathbf{p}_\ell} + \sum_{j\neq\ell}\frac{\mathrm{E}\left[(AX)_j^2\right]}{\mathbf{p}_j} \\ &= (1-\mathbf{p}_\ell)R^A(X) + \mathbf{p}_\ell\left(R^A(X)\right)^2 \\ &\quad + \sum_{j\neq\ell}\Big(X_j^2/\mathbf{p}_j + 2X_jR^A(X) \\ &\qquad + (1-\mathbf{p}_j)R^A(X) + \mathbf{p}_j\left(R^A(X)\right)^2\Big) \\ &= Z(X) - \left(2 + \frac{1}{\mathbf{p}_\ell}\right)\left(R^A(X)\right)^2 + 2mR^A(X) \\ &\quad + \sum_j\Big((1-\mathbf{p}_j)R^A(X) + \mathbf{p}_j\left(R^A(X)\right)^2\Big) \\ &= Z(X) - \left(1 + \frac{1}{\mathbf{p}_\ell}\right)\left(R^A(X)\right)^2 \\ &\qquad + (2m+n-1)R^A(X)\end{aligned}$$

Now, we can prove Lemma 4.3.

Proof. [Proof of Lemma 4.3] Let χ^A be the stationary distribution relative to A. Let ϕ_j be the event that A recycles the jth bin of χ^A, R_j^A the random variable of how many balls are recycled given the jth bin is chosen by A, $\mathcal{R}_j^A = \mathrm{E}\left[R_j^A\right]$ and f_j the probability that A recycles that bin. Because $\chi^A = A\chi^A$ by definition, we must have $\mathrm{E}\left[Z(A\chi^A)\right] = \mathrm{E}\left[Z(\chi^A)\right]$. Therefore:

$$\begin{aligned}\mathrm{E}\left[Z(\chi^A)\right] &= \mathrm{E}\left[Z(A\chi^A)\right] \\ &= \sum_j f_j\mathrm{E}\left[Z(A\chi^A)|\phi_j\right] \\ &= \sum_j f_j\Big(\mathrm{E}\left[Z(\chi^A)|\phi_j\right] \\ &\qquad - \left(1 + \frac{1}{\mathbf{p}_j}\right)\mathrm{E}\left[\left(R_j^A\right)^2\right] \\ &\qquad + (2m+n-1)\mathcal{R}_j^A\Big) \\ &\leq \mathrm{E}\left[Z(\chi^A)\right] + (2m+n-1)\mathcal{R}^A \\ &\qquad - \sum_j f_j\left(1 + \frac{1}{\mathbf{p}_j}\right)\left(\mathcal{R}_j^A\right)^2 \\ &= \mathrm{E}\left[Z(\chi^A)\right] + (2m+n-1)\mathcal{R}^A \\ &\qquad - \sum_j \frac{1}{f_j}\left(\mathbf{p}_j^2 + \mathbf{p}_j\right)\left(\mathcal{R}^A\right)^2,\end{aligned}$$

where the inequality is due to the Cauchy-Schwartz Inequality and the last line is because of the Flow Equation.

Thus we have:

$$\mathcal{R}^A \leq \frac{2m+n-1}{\sum_j \frac{1}{f_j}\left(\mathbf{p}_j^2+\mathbf{p}_j\right)} \leq \frac{2m+n-1}{\sum_j \frac{\mathbf{p}_j}{f_j}}$$

Lemma 4.3 applies to the optimal deterministic strategy OPT promised by Corollary 3.1, and we know that $\mathcal{R}^A \leq \mathcal{R}^{\mathrm{OPT}}$ for *any* recycling strategy A. Thus, by maximizing the RHS of Lemma 4.3, we can get an upper bound on the recycling rate of any recycling strategy.

LEMMA 4.7. *Consider a ball-recycling game with m balls, n bins and distribution $\mathbf{p}$. For any recycling strategy A,*

$$\mathcal{R}^A \leq \frac{2m+n-1}{\|\mathbf{p}\|_{\frac{1}{2}}}.$$

Proof. This follows immediately from the Cauchy-Schwartz Inequality.

4.3 Random Ball with $m \geq n$ We show the following lower bound, which with Lemma 4.7, shows optimality when $m = \Omega(n)$.

LEMMA 4.8. RANDOM BALL *recycles at least $\frac{m}{\|\mathbf{p}\|_{\frac{1}{2}}}$ balls per round in expectation.*

Proof. Let $\chi^{\mathrm{RB}} = (\chi_i^{\mathrm{RB}})$ be the random variable of the number of balls in each bin in the stationary distribution of RANDOM BALL. RANDOM BALL recycles bin i with probability $\frac{\chi_i^{\mathrm{RB}}}{m}$, and therefore the expected number of balls recycled from bin i per round is $\frac{\mathrm{E}\left[\left(\chi_i^{\mathrm{RB}}\right)^2\right]}{m}$. The number of balls that land in bin i per round is $\mathbf{p}_i \sum_{j=1}^n \frac{\mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right]}{m}$. Since X is distributed stationarily, we must have

$$\mathbf{p}_i \sum_{j=1}^n \frac{\mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right]}{m} = \frac{\mathrm{E}\left[\left(\chi_i^{\mathrm{RB}}\right)^2\right]}{m} \geq \frac{\mathrm{E}\left[\chi_i^{\mathrm{RB}}\right]^2}{m},$$

using Jensen's Inequality. Clearing denominators, taking square roots and summing across i, we have

$$\left(\sum_{j=1}^n \mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right]\right)^{\frac{1}{2}} \sum_{i=1}^n \mathbf{p}_i^{\frac{1}{2}} = \sum_{i=1}^n \left(\mathbf{p}_i \sum_{j=1}^n \mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right]\right)^{\frac{1}{2}} \geq \sum_{i=1}^n \mathrm{E}\left[\chi_i^{\mathrm{RB}}\right] = m.$$

Therefore the expected recycle rate is

$$\sum_{j=1}^n \frac{\mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right]}{m} \geq \frac{m}{\left(\sum_{i=1}^n \sqrt{\mathbf{p}_i}\right)^2} = \frac{m}{\|\mathbf{p}\|_{\frac{1}{2}}}.$$

COROLLARY 4.1. *Consider a ball-recycling game with m balls and n bins. If $m = \Omega(n)$, then* RANDOM BALL *is asymptotically optimal among recycling strategies.*

4.4 Aggressive Empty is Optimal In this section, we investigate AGGRESSIVE EMPTY strategies, which aggressively recycle balls outside a given subset of bins. An AGGRESSIVE EMPTY strategy runs one strategy on a fixed subset of bins, but always chooses to recycle a bin outside of this set if there exists one which has any balls. Specifically, we show that a $\Theta(1)$-optimal strategy on a particular $O(m)$ subset of the bins can be extended to an $\Theta(1)$-optimal strategy on the full ball-recycling game by aggressively emptying the rest.

Consider a ball-recycling game with m balls, n bins, and ball distribution $\mathbf{p}$. Let L be some subset of bins and S be a strategy on the ***induced ball-recycling game*** of L, which is the ball-recycling game with m balls, $|L|$ bins, and ball distribution $\mathbf{q}$, where

$$\mathbf{q}_i = \frac{\mathbf{p}_i}{\sum_{\ell \in L} \mathbf{p}_\ell}.$$

Therefore, $\mathbf{q}$ is $\mathbf{p}$'s conditional probability distribution on L. We define L, S-AGGRESSIVE EMPTY to be the strategy which empties the lowest weight non-empty bin in the complement of L if one exists and otherwise performs S on L. Note that all the balls will be in L whenever S is performed, so this is well-defined.

We begin by showing that there exists an L and S such that $|L| = O(m)$, L contains all bins with weight at least $\frac{1}{m}$, and L, S-AGGRESSIVE EMPTY is asymptotically optimal. Note that when $m = \Omega(n)$, this is trivial, because we can take L to be all the bins and S to be a $\Theta(1)$-optimal strategy; however, this section provides stronger bounds when $m = o(n)$. Intuitively, the idea is that very low weight bins won't be able to effectively accumulate balls, so strategies do better to recover any balls in them than to wait for more balls to land there.

LEMMA 4.9. *There exists an L and S such that $|L| = O(m)$, L contains all bins of weight at least $\frac{1}{m}$ and L, S-*AGGRESSIVE EMPTY *is asymptotically optimal.*

Proof. By Lemma 3.1, there exists an optimal deterministic strategy OPT. Using the flow equation, Lemma 4.3 can be rewritten as:

$$\sum_{i=1}^n \mathcal{R}_i^{\mathrm{OPT}} \leq 2m+n.$$

Because OPT will never recycle an empty bin, each $\mathcal{R}_i^{\text{OPT}} \geq 1$. Therefore, there can be at most m bins with average recycle rates at least 3. Let L be this set of bins, together with any bins of weight at least $\frac{1}{m}$, and we will construct a strategy S that aggressively empties the remaining bins into L.

S aggressively empties the complement of L, but also keeps a virtual configuration of where OPT thinks the balls are, as well as a log of where S has moved them. So when S aggressively empties a bin, it also updates the log of each ball it throws, indicating where it landed. When L^c is empty, it asks OPT which bin to recycle based on the virtual configuration. If it says to recycle a bin in L^c, we use the logs to update where those balls will land in the virtual configuration. If it says to recycle a bin in L, we recycle those balls that are there in the virtual configuration, and leaving any others behind in that same bin. Thus S performs OPT but rushes ahead to recycle those balls outside of L.

Now, consider t rounds of OPT. For large enough t, OPT will recycle on average at most 3 balls at a time from L^c. S recycles at least 1 ball at a time from L^c and exactly as many balls at a time from L. Therefore for large t, t rounds of OPT will correspond to at most $3t$ rounds of S, and during this period S will recycle the same number of balls. Thus S is 1/3-optimal.

Next we compute the recycle rate of L, S-Aggressive Empty as a function of the recycle rate of S on the induced ball-recycling game on L.

Lemma 4.10. *If $\mathcal{R}^S$ is the recycle rate of S (on L), and q is the probability of a ball landing in L^c, then the recycle rate of L, S-Aggressive Empty is*

$$\mathcal{R}^{AE} = \Theta\left(\frac{1}{(1-q)/\mathcal{R}^S + q}\right)$$

Proof. Consider a collection of recycling rounds of L, S-Aggressive Empty where t of those times L, S-Aggressive Empty recycles a bin from L. Say b balls are thrown from bins in L and a balls land in L^c. Now, if m balls are thrown into bins of size at most $\frac{1}{m}$, then the expected number of empty bins is at most

$$m\left(1 - \frac{1}{m}\right)^m \leq \frac{m}{e}.$$

Because fewer thrown balls will have fewer collisions, this means the expected number of non-empty bins when $k \leq m$ balls are thrown into L^c is at least $\left(1 - \frac{1}{e}\right)k$, requiring at least as many time steps to aggressively empty. Thus, for large t, the expected number of turns required to empty the a balls out of L^c is at least $\left(1 - \frac{1}{e}\right)\frac{a}{1-q}$. Whereas even if the balls were recycled from L^c one at a time this expected number of turns is at most $\frac{a}{1-q}$ turns. The number of balls recycled during this period is $b + \frac{a}{1-q}$, and we have shown the number of rounds ρ satisfies:

$$\rho = \Theta\left(t + \frac{a}{1-q}\right).$$

For large enough t, $b = \Theta\left(t\mathcal{R}^S\right)$ and $a = \Theta\left(tq\mathcal{R}^S\right)$, so the overall recycle rate $\mathcal{R}^{\text{AE}}$ therefore satisfies

$$\begin{aligned}\mathcal{R}^{\text{AE}} &= \Theta\left(\frac{t\mathcal{R}^S + tq\mathcal{R}^S/(1-q)}{t + tq\mathcal{R}^S/(1-q)}\right) \\ &= \Theta\left(\frac{1}{(1-q)/\mathcal{R}^S + q}\right)\end{aligned}$$

4.5 Random Ball is Optimal In this section we will further examine the performance of Random Ball and show that it is asymptotically optimal. We first describe a sufficient condition for optimality of a strategy based on its recycle rate on L, then show that Random Ball satisfies this criterion.

Lemma 4.11. *Let L be a set of $O(m)$ bins for which there exists a strategy T such that L, T-Aggressive Empty is asymptotically optimal. Let $\mathcal{R}^{\text{OPT}_L}$ be the recycle rate of the optimal strategy on the induced ball-recycling game of L. For a given strategy S, let $\mathcal{R}_L^S$ be the conditional recycle rate of S in the stationary distribution given that a ball in L is selected, and q be the probability that a ball lands in L^c, i.e. $q = \sum_{k \in L^c} \mathbf{p}_k$. If either*

$$\mathrm{E}\left[\mathcal{R}_L^S\right] = \Omega(\mathcal{R}^{\text{OPT}_L}) \quad \textit{or} \quad \mathrm{E}\left[\mathcal{R}_L^S\right] = \Omega\left(\frac{1}{q}\right),$$

then S is asymptotically optimal.

Proof. By applying Lemma 4.1, the subset variant of the flow equation, to L,

$$f_L\mathcal{R}_L^S = (1-q)(f_L\mathcal{R}_L^S + (1-f_L)\mathcal{R}_{L^c}^S),$$

where f_L is the stationary probability of S picking a bin in L. Solving for f_L,

$$f_L = \frac{\mathcal{R}_{L^c}^S}{q\mathcal{R}_L^S + \mathcal{R}_{L^c}^S}. \tag{4.9}$$

Suppose $\mathcal{R}_L^S = o\left(\frac{1}{q}\right)$ and $\mathcal{R}_L^S$ is $\Theta(1)$-optimal on L. If $\mathcal{R}_L^S \leq \frac{1}{q}$, then $f_L \geq \frac{1}{2}$, and so because $\mathcal{R}^S = f_L\mathcal{R}_L^S + (1-f_L)\mathcal{R}_{L^c}^S$, we must have $\mathcal{R}^S = \Omega(\mathcal{R}_L^S)$.

Now, using Lemma 4.9, let L and T be such that L, T-Aggressive Empty is asymptotically optimal, and let $\mathcal{R}^{\text{AE}}$ be its expected recycle rate. By

Lemma 4.10,

$$\begin{aligned}\mathcal{R}^{\mathrm{AE}} &= \Theta\left(\frac{1}{(1-q)/\mathcal{R}^T + q}\right)\\ &= O\left(\mathcal{R}^T\right) = O\left(\mathcal{R}^S_L\right) = O\left(\mathcal{R}^S\right),\end{aligned}$$

so S must be asymptotically optimal.

If $\mathcal{R}^S_L = \Omega\left(\frac{1}{q}\right)$, then $\mathcal{R}^S_L > \frac{\alpha}{q}$ for some α. Rearranging Equation (4.9) and multiplying by $\mathcal{R}^S_L$ yields

$$f_L \mathcal{R}^S_L = \frac{1}{\frac{q}{\mathcal{R}^S_{L^c}} + \frac{1}{\mathcal{R}^S_L}}.$$

Here $\frac{1}{\mathcal{R}^S_L} \leq \frac{q}{\mathcal{R}^S_{L^c}}$, so $f_L \mathcal{R}^S_L = \Omega\left(\frac{1}{q}\right)$, and thus $\mathcal{R}^S = \Omega\left(\frac{1}{q}\right)$ as well. Now we can compare to L,T-Aggressive Empty as above:

$$\mathcal{R}^{\mathrm{AE}} = \Theta\left(\frac{1}{(1-q)/\mathcal{R}^T + q}\right) = O\left(\frac{1}{q}\right) = O\left(\mathcal{R}^S\right)$$

so in this case S is asymptotically optimal as well.

We can now prove Theorem 1.1.

Proof. [Proof of Theorem 1.1] If $m = \Omega(n)$, then by Lemmas 4.7 and 4.8 we are done.

Otherwise, let L be a set of $O(m)$ bins for which there exists a strategy T such that L,T-Aggressive Empty is asymptotically optimal. We will prove the result for a slightly modified Random Ball that only recycles 1 ball outside of L even if more are available; that is, it moves only one of the balls in the bin. Since this strategy is worse than Random Ball, this will be sufficient. We number the bins so that the first $|L|$ bins comprise L.

If $\mathcal{R}_L \geq \frac{1-q}{q}$, then we are done by Lemma 4.11. Otherwise, in the stationary distribution, when a bin in L is recycled, the expected number of balls which land in L^c is $q\mathcal{R}_L < 1-q$. When a bin in L^c is recycled, the expected number of balls which land in L is $1-q$. Thus Random Ball must pick a bin in L more than half the time, and so the expected number of balls in L must be more than $\frac{m}{2}$.

Now analogously to the proof of Lemma 4.8, we have:

$$\begin{aligned}\mathbf{p}_i\left(\sum_{j=1}^{|L|}\mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right] + \sum_{j=|L|+1}^{n}\mathrm{E}\left[\chi_j^{\mathrm{RB}}\right]\right) &= \mathrm{E}\left[\left(\chi_i^{\mathrm{RB}}\right)^2\right]\\ &\geq \mathrm{E}\left[\chi_i^{\mathrm{RB}}\right]^2.\end{aligned}$$

Thus,

$$\mathrm{E}\left[\chi_i^{\mathrm{RB}}\right] \leq \sqrt{\mathbf{p}_i}\left(\sum_{j=1}^{|L|}\mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right] + \frac{m}{2}\right)^{\frac{1}{2}}.$$

Summing over $i \leq |L|$ yields

$$\mathrm{E}\left[\chi_L^{\mathrm{RB}}\right] \leq \left(\sum_{i=1}^{|L|}\sqrt{\mathbf{p}_i}\right)\left(\sum_{j=1}^{|L|}\mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right] + \frac{m}{2}\right)^{\frac{1}{2}},$$

where χ_L^{RB} is the expected number of balls in L. Now,

$$\begin{aligned}\mathcal{R}_L^{\mathrm{RB}} &\geq \frac{1}{m}\sum_{j=1}^{|L|}\mathrm{E}\left[\left(\chi_j^{\mathrm{RB}}\right)^2\right]\\ &\geq \frac{\mathrm{E}\left[\chi_L^{\mathrm{RB}}\right]^2}{m\left(\sum_{i=1}^{|L|}\sqrt{\mathbf{p}_i}\right)^2} - \frac{1}{2}\\ &> \frac{m}{4\|\mathbf{p}_L\|_{\frac{1}{2}}} - \frac{1}{2},\end{aligned}$$

where $\mathbf{p}_L$ is the conditional probability distribution on L obtained from $\mathbf{p}$. The last inequality holds because there are at least $\frac{m}{2}$ balls in L in expectation.

Thus by Lemma 4.7, Random Ball is asymptotically optimal on the induced system of L, and therefore Random Ball is asymptotically optimal by Lemma 4.11.

5 The Uniform Case

The results of Section 4 hold for any distribution of the balls into the bins. In this section we consider the special case where they are uniformly distributed, which models insertion buffers as discussed in Section 2. We then show that Golden Gate and Fullest Bin are optimal, up to lower-order terms, in this setting, whereas Random Ball is at least 1/2- and at most $(1-\epsilon)$-optimal, for some constant $\epsilon > 0$.

For a ball-recycling game with uniformly distributed balls, Lemma 4.7 implies:

Corollary 5.1. *Consider a ball-recycling game with m balls, n bins and uniform distribution $\mathbf{u}$. For any recycling strategy A,*

$$\mathcal{R}^A \leq \frac{2m+n-1}{n} < 2\frac{m}{n} + 1.$$

The average number of balls in a bin is m/n, so Corollary 5.1 suggests that any "reasonable" strategy will be at least 1/2-optimal in the uniform case.

We now show that Golden Gate and Fullest Bin are within an additive constant of optimal on strictly uniform distributions.

Lemma 5.1. Golden Gate *and* Fullest Bin *each recycle at least $2m/(n+1)$ balls per round in expectation.*

Proof. Let S be the random variable denoting the number of balls thrown in a given round with GOLDEN GATE. GOLDEN GATE will recycle the bins in order starting from the next one and cycling around. Therefore, we can consider the collection of bins to be a queue. After throwing the balls, the average place in the queue in which a ball lands is the $[(n-1)/s]$th bin, due to uniformity. Each ball thrown will therefore sit for an average of at most $(n-1)/2$ rounds before it is thrown again. Therefore, $m - \mathrm{E}[S] \leq \mathrm{E}[S](n-1)/2$, and we have the result after solving for $\mathrm{E}[S]$.

Let T be the random variable denoting the number of balls thrown in a given round with FULLEST BIN. If after removing the balls in the FULLEST BIN, we list the bins in order of fullness, we can again think of the bins as a sort of queue. When we throw the balls, the average place in the queue which a ball lands is the $[(n-1)/2]$th bin as above, due to uniformity. Now, we reorder the bins back into fullness order. During the reordering more balls are moved up the queue than down, thus each ball thrown into the system will sit for an average of less than $(n-1)/2$ rounds before it is thrown again. Therefore, as above, $m - \mathrm{E}[S] \leq \mathrm{E}[S](n-1)/2$, and we are done.

Corollary 5.1 and Lemma 5.1 together prove Theorem 1.2. Despite these strong performance bounds, recall that FULLEST BIN can perform arbitrarily badly on non-uniform $\mathbf{p}$. RANDOM BALL on the other hand is always $\Theta(1)$-optimal.

5.1 Random Ball in the Uniform Case However, RANDOM BALL does not achieve this level of optimality on uniform distributions. In this section we will show in Theorem 1.3 that RANDOM BALL recycles at most $1 + (2-\epsilon)m/n$ balls per round in expectation, for some $\epsilon > 0$. The upper bound is given in Lemma 5.3 and Corollary 5.2, and the lower bound is given in Lemma 5.4.

We begin with the following lemma:

LEMMA 5.2. *Let χ^{RB} be the stationary distribution relative to* RANDOM BALL, *$R^{RB}(X)$ the random variable of how many balls* RANDOM BALL *recycles from ball configuration X, and $\mathcal{R}^{RB} = \mathrm{E}\left[R^{RB}\left(\chi^{RB}\right)\right]$ the expected recycle rate of* RANDOM BALL. *Then,*

$$\frac{\mathrm{E}\left[R^{RB}\left(\chi^{RB}\right)^2\right]}{\mathcal{R}^{RB}} = \frac{2m+n-1}{n+1} \leq 1 + \frac{2m}{n}.$$

Proof. Consider the random variable of the number of distinct unordered pairs of balls which are in the same bin in χ^{RB}. In expectation, a round of RANDOM BALL eliminates

$$\binom{\mathcal{R}^{\mathrm{RB}}}{2}$$

and creates

$$\sum_{k=0}^{\mathcal{R}-1} \frac{m - \mathcal{R}^{\mathrm{RB}} + k}{n}$$

such pairs. In the stationary distribution, these must be equal, so

$$\frac{\mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2\right]}{2} - \frac{\mathcal{R}^{\mathrm{RB}}}{2} = \frac{(2m-1)\mathcal{R}^{\mathrm{RB}}}{2n} - \frac{\mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2\right]}{2n}.$$

After rearranging we have the result.

LEMMA 5.3. *There exists a constant $\alpha > 0$ such that* RANDOM BALL *is at most $(1-\alpha)$-optimal.*

Proof. Let χ^{RB} be the stationary distribution relative to RANDOM BALL, $R^{\mathrm{RB}}(X)$ the random variable of how many balls RANDOM BALL recycles from ball configuration X, and $\mathcal{R}^{\mathrm{RB}} = \mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)\right]$ the expected recycle rate of RANDOM BALL. We will prove the result by contradiction, so assume that for all constant $\epsilon > 0$

$$\mathcal{R}^{\mathrm{RB}} \geq 1 + \frac{(2-\epsilon)m}{n}.$$

Let $c \in (1,2)$ be a constant to be determined later. We say a bin is ***light*** if it contains at most cm/n balls. Let L be the random variable of the number of balls in light bins in the stationary distribution. Then the probability q_L that RANDOM BALL recycles a light bin in the stationary distribution is $\mathrm{E}[L]/m$. We proceed by cases.

Case 1. Suppose $\mathrm{E}[L] \geq \delta m$ for some constant $\delta > 0$. Then $q_L \geq \delta$ and for $c \leq 2 - 2\epsilon$ and $\epsilon < 1/2$,

$$\begin{aligned}
\mathrm{Var}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)\right] &= \mathrm{E}\left[\left(R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right) - \mathcal{R}^{\mathrm{RB}}\right)^2\right] \\
&\geq q_L\left(1 + \frac{(2-\epsilon)m}{n} - \frac{cm}{n}\right)^2 \\
&\geq \frac{\epsilon^2\delta}{4}\left(\frac{4m^2}{n^2} + \frac{4m}{n} + 1\right) \\
&\geq \frac{\epsilon^2\delta}{4}\left(\mathcal{R}^{\mathrm{RB}}\right)^2.
\end{aligned}$$

Thus by the definition of variance, we have

$$\mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2\right] \geq \left(1 + \frac{\epsilon^2\delta}{4}\right)\left(\mathcal{R}^{\mathrm{RB}}\right)^2.$$

Now by Lemma 5.2,

$$\mathcal{R}^{\mathrm{RB}} \leq \left(1+\frac{\epsilon^2\delta}{4}\right)^{-1}\left(1+\frac{2m}{n}\right).$$

Since ϵ, δ are constants greater than 0, we have our contradiction for the first case.

Case 2. Otherwise, $\mathrm{E}[L] < \delta m$. Since $L \in [0, m]$, $\mathrm{E}\left[L^2\right] < \delta m^2$. Lemma 5.2 implies $\mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2\right] \leq (1+2m/n)^2$. Together Hölder's inequality we have

$$\begin{aligned}\mathrm{E}\left[LR^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)\right] &\leq \left(\mathrm{E}\left[L^2\right]\mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2\right]\right)^{1/2} \\ &< \left(\delta m^2\left(1+\frac{2m}{n}\right)^2\right)^{1/2} \\ &= \sqrt{\delta}m\left(1+\frac{2m}{n}\right) \qquad (5.10)\end{aligned}$$

Let Y be the random variable of the number of balls in the stationary distribution which start in a light bins, but end up begin among the first $1+cm/n$ balls in a heavy bin after an application of Random Ball. Let Φ be the random variable of the number of distinct unordered pairs of balls that are in the same light bin in the stationary distribution. Applying Random Ball in expectation creates at most

$$\begin{aligned}&\mathrm{E}\left[\sum_{k=0}^{\mathcal{R}^{\mathrm{RB}}-1}\frac{L+k}{n}\right] \\ &= \mathrm{E}\left[\frac{2LR^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)+R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2-R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)}{2n}\right]\end{aligned}$$

such pairs, and eliminates at least

$$\mathrm{E}\left[\frac{Y}{1+\frac{cm}{n}}\binom{1+\frac{cm}{n}}{2}\right].$$

In the stationary distribution these quantities must be equal, so rearranging together with Equation (5.10), we have

$$\begin{aligned}\mathrm{E}[Y] &\leq \frac{2\mathrm{E}\left[LR^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)\right]+\mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2\right]-\mathcal{R}^{\mathrm{RB}}}{cm} \\ &< \frac{2\sqrt{\delta}}{c}\left(1+\frac{2m}{n}\right)+\frac{\mathrm{E}\left[R^{\mathrm{RB}}\left(\chi^{\mathrm{RB}}\right)^2\right]-\mathcal{R}^{\mathrm{RB}}}{cm} \\ &< \left(1+\frac{2m}{n}\right)\left(\frac{2\sqrt{\delta}}{c}+\frac{2}{cn}\right),\end{aligned}$$

where we have used Lemma 5.2 for the last inequality.

We now compute the effect on $\mathrm{E}[L]$ of applying Random Ball to the stationary distribution. By Markov's inequality, there must be more than $(1-1/c)n$ light bins, and so the probability that a ball is thrown into a light bin is more than $1-1/c$. Therefore, at least $(1-1/c)\mathcal{R}^{\mathrm{RB}}$ balls land in light bins in expectation. We expect at most $\mathrm{E}[Y]$ balls to be in light bins which turn into heavy bins. Finally, we recycle at most cm/n balls from a light bin $\mathrm{E}[L]/m$ of the time. Since the net change to L must be 0 in expectation,

$$\left(1-\frac{1}{c}\right)\mathcal{R}^{\mathrm{RB}} < \frac{c}{n}\mathrm{E}[L]+\mathrm{E}[Y].$$

However, this is a contradiction. Indeed, the LHS is at least

$$\left(1-\frac{1}{c}\right)\left(1+\frac{(2-\epsilon)m}{n}\right),$$

but the RHS is less than

$$\delta c\frac{m}{n}+\left(1+\frac{2m}{n}\right)\left(\frac{2\sqrt{\delta}}{c}+\frac{2}{cn}\right).$$

Thus, if we pick a sufficiently small $\delta > 0$, $\epsilon = 0.01$, $c = 1.98$ and $n \geq 3$, we have a contradiction. For $n \leq 2$, the contradiction follows immediately from Lemma 5.2.

Corollary 5.2. *Setting*

$$(\epsilon, c, \delta) = (0.001, 1.456, 0.042)$$

in the proof of Theorem 1.3, we obtain

$$\mathcal{R}^{RB} < 1+1.994\frac{m}{n}.$$

Lemma 5.4. *For all $c > 0$, there exists a c' such that if $m \geq c'n\log n$, the uniform random ball policy has expected recycle rate at least*

$$\left(1+\frac{1}{6^4}-c\right)\frac{m}{n}.$$

Proof. Let $X_{t,k}$ be the random variable denoting the number of balls in the kth bin at the beginning of the tth round. Because of symmetry, $X_{t,k}$ follows the same distribution as $X_{t,\ell}$ for any $k \neq \ell$. For simplicity, we let X_t be a random variable that follows the same distribution as $X_{t,k}$ for all k.

We pick t to be sufficiently large so that the system enters its stationary state after t rounds. Thus, X_t and $X_{t'}$ follows the same distribution for any $t' > t$.

Let Y_t be the random variable denoting the number of balls recycled in the tth round. By definition, we have

$$\mathrm{E}[Y_t] = \sum_{1\leq k\leq n}\frac{\mathrm{E}\left[X_{t,k}^2\right]}{m} = \frac{n}{m}\left(\mathrm{E}[X_t]^2+\mathrm{Var}[X_t]\right).$$

Note that $\mathrm{E}[X_t] = m/n$, and so $\mathrm{E}[Y_t] \geq m/n$.

To show $\mathrm{E}[Y_t]$ deviates from m/n, we derive a lower bound for $\mathrm{Var}[X_t]$.

If $\mathrm{P}(X_t \leq (1-\epsilon)m/n) \geq \delta$, then $\mathrm{E}[Y_t] \geq (1+\epsilon^2\delta)m/n$.

Otherwise $\mathrm{P}(X_t > (1-\epsilon)m/n) > 1-\delta$. We will show that if δ is small enough, then this case does not exist.

We say a bin is *heavy* if it has more than $(1-\epsilon)m/n$ balls. Let Z_t be the random variable denoting the number of heavy bins at the beginning of the tth round. We have

$$\mathrm{E}[Z_t] = \sum_{1 \leq k \leq n} \mathrm{E}\left[\mathrm{I}\left[X_{t,k} > (1-\epsilon)\frac{m}{n}\right]\right] > (1-\delta)n.$$

Z_t is a non-negative variable in $[0, n]$ and has expected value more than $(1-\delta)n$. By Markov's inequality,

$$\Pr[Z_t \leq (1-2\delta)n] < \frac{1}{2} \text{ and } \Pr[Z_t > (1-2\delta)n] > \frac{1}{2}.$$

We compute $\mathrm{E}\left[Z_{t+n/2}\right]$ from the Z_t. If $Z_t > (1-2\delta)n$ for some constant $\delta < 1/4$, the following hold during P, the time period between the tth round and the $(t+n/2)$th round:

1. At least $(1/2-2\delta)n$ bins in H_t are recycled,
2. At least $(1/2-2\delta)(1-\epsilon)m$ balls are recycled,
3. At least $(1/2-2\delta)n$ bins in H_t are not recycled,

where H_t denotes the set of heavy bins at the beginning of the tth round.

Given (c), we can find a subset $S_t \subset H_t$ that is composed of $(1/2-2\delta)n$ bins in H_t not recycled during P. Note that which bins are recycled and which are not depend on the random choices made by the system. Hence, S_t varies.

Next, we derive a lower bound on the expected number of balls in any S_t. The balls which stay in S_t come from two different sources. There are those that stay in S_t at the beginning of the tth round, of which there are at least $|S_t|(1-\epsilon)m/n$. There are also those which are recycled during P, of which there are at least $|S_t|(1-\epsilon')|B|/n$ by Lemma 5.5, to follow. Combining the two sources, the expected number of balls in S_t is at least

$$\Gamma = (1-\epsilon)\left(\left(\frac{1}{2}-2\delta\right) + \left(\frac{1}{2}-2\delta\right)^2(1-\epsilon')\right)m.$$

LEMMA 5.5. *Let B be the multiset of the first $(1/2-2\delta)(1-\epsilon)m$ balls recycled during P. B is well-defined thanks to 2. above. Let L_i be the random variable denoting the number of balls in B that land on the ith bin. For all $\epsilon' > 0$, there exists a c' such that if $m \geq c'n\log n$*

$$\mathrm{E}[\min\{L_1, L_2, \ldots, L_n\}] \geq (1-\epsilon')|B|/n.$$

Proof. For $i \in [1,n]$, $\mathrm{E}[L_i] = |B|/n$. By Chernoff bounds,

$$\Pr[|L_i - \mathrm{E}[L_i]| \geq (\epsilon'/2)\mathrm{E}[L_i]] \leq \frac{1}{n^2}$$

for some sufficiently large c'. Consequently, by the union bound,

$$\Pr[\min\{L_1, L_2, \ldots, L_n\} \leq (1-\epsilon'/2)|B|/n] \leq \frac{1}{n}.$$

Because the L_i's are non-negative, we are done.

Given Γ, we obtain the following bound:

$$\begin{aligned}\mathrm{E}\left[Z_{t+n/2} \,\middle|\, Z_t > (1-2\delta)n\right] &\leq |S_t| + \frac{m-\Gamma}{(1-\epsilon)m/n} \\ &= \left(\frac{1}{1-\epsilon} - \left(\frac{1}{2}-\delta\right)^2(1-\epsilon')\right)n \\ &\approx \left(\frac{3}{4}+\epsilon+\delta\right)n\end{aligned}$$

Together with the trivial bound $\mathrm{E}\left[Z_{t+n/2} \,\middle|\, Z_t \leq (1-2\delta)n\right] \leq n$, $\mathrm{E}\left[Z_{t+n/2}\right]$ equals

$$\begin{aligned}&\mathrm{P}(Z_t \leq (1-2\delta))\,\mathrm{E}\left[Z_{t+n/2} \,\middle|\, Z_t \leq (1-2\delta)\right] \\ &\quad + \mathrm{P}(Z_t > (1-2\delta))\,\mathrm{E}\left[Z_{t+n/2} \,\middle|\, Z_t > (1-2\delta)\right] \\ &\qquad < \frac{1}{2}\left(1 + \left(\frac{3}{4}+\epsilon+\delta\right)\right)n \\ &\qquad \approx \left(\frac{7}{8} + \frac{\epsilon+\delta}{2}\right)n\end{aligned}$$

This leads to a contradiction if $\epsilon+\delta$ is small enough. This is because we have $\mathrm{E}[Z_t] > (1-\delta)n$ and $\mathrm{E}\left[Z_{t+n/2}\right] = \mathrm{E}[Z_t]$, because the system is stationary. As a result, we have a contradiction if $\frac{\epsilon+3\delta}{2} < \frac{1}{8}$.

Combining the results for the two cases, we wish to maximize $1+\epsilon^2\delta$ subject to $\epsilon+3\delta < \frac{1}{4}$. Picking $\epsilon = 1/6$ yields the result.

Theorem 1.3 follows from Corollaries 5.1 and 5.2 and Lemma 5.4.

6 Database Experiments

In this section, we consider insertion buffers as they are used in practice. We demonstrate through simulations as well as experiments on real-world systems, that the theoretical results in the prior sections hold and can be used to improve performance.

6.1 Insertion Buffers in Database Systems Many databases cache recently inserted items in RAM so that they can write items to disk in batches. Examples include Azure [4], DB2 [25], Hbase [47], Informix [26], InnoDB [14], NuDB [34], Oracle [35], SAP [41], and Vertica [44]. They are also used to accelerate inserts in several research prototypes, such as the buffered Bloom filter [15] and buffered quotient filter [7]. By batching updates to disk, these insertion buffers reduce the amortized number of I/Os per insert, which can substantially improve insertion throughput. Facebook claims that the insertion buffer in InnoDB speeds up some production workloads by a factor of 5 to 16, and accelerates some synthetic benchmarks by up to a factor of 80 [14].

A motivating factor for the use of insertion buffers is that they can significantly mitigate the precipitous performance drop that databases can experience when the data set grows too large to fit in RAM. Figure 1a shows the time per 1,000 insertions into a MySQL database using the InnoDB backend, with and without InnoDB's insertion buffer enabled. For the first $200,000$ insertions, the entire database fits in RAM, and so insertions are fast, even without the insertion buffer.

Once the database grows larger than RAM, insertion performance without the insertion buffer falls off a cliff. In fact, once the database reaches 1M rows, it can perform only about 200 insertions per second, suggesting that the throughput is limited by the random-I/O performance of the underlying disk. In the benchmark with the insertion buffer enabled, on the other hand, performance degrades by only a small amount.

Based on the performance of the first 1M insertions, it appears that InnoDB's insertion buffer effectively eliminates the performance cliff that can occur when the database grows larger than RAM. This improvement explains the popularity of insertion buffers in database design.

However, in our experiment, as the database continues to grow, the efficacy of the insertion buffer declines. Figure 1b shows the time per 10,000 insertions as the database grows to 50M rows. Although the performance without the insertion buffer drops more quickly early on, it remains relatively stable thereafter. Performance with the insertion buffer, on the other hand, slowly declines over the course of the benchmark until it is only about a third faster than without the insertion buffer. This is well below the $5-80\times$ speedups reported above.

As these experiments show, it can be difficult to extrapolate from small examples the performance gains that insertion buffers can provide for large databases. Therefore, it is no wonder that reported speedups from insertion buffers vary wildly from as little as $2\times$ to as high as $80\times$ [14]. Some have even suggested that insertion buffers may provide many of the benefits of write-optimization [13], i.e., that insertion buffers can bring the performance of B-trees up to that of LSM-trees [38], COLAs [5], Fractal Trees [42], xDicts [11], or B^{ε}-trees [12].

6.2 Experimental Validation Here we validate our theoretical study of insertion buffers by showing that our analysis above can have a material impact on the performance of databases with insertions buffers. We simulated workloads of random insertions to a B-tree, with varying distributions on the inserted keys. We found that, as predicted, the performance was independent of the input distribution and closely matched the performance predicted by our theorems.

We then ran workloads of random insertions into InnoDB and measured the average batch size of flushes from its insertion buffer. InnoDB implements a variant of the random-item flushing strategy. We modified it to implement the golden-gate flushing strategy. Despite the additional complexities of InnoDB's insertion buffer implementation, we found that performance closely tracked our theoretical predictions and was independent of the distribution of inserted keys. We also found that the golden-gate flushing strategy improved InnoDB's flushing rate by about 30% over the course of our benchmark.

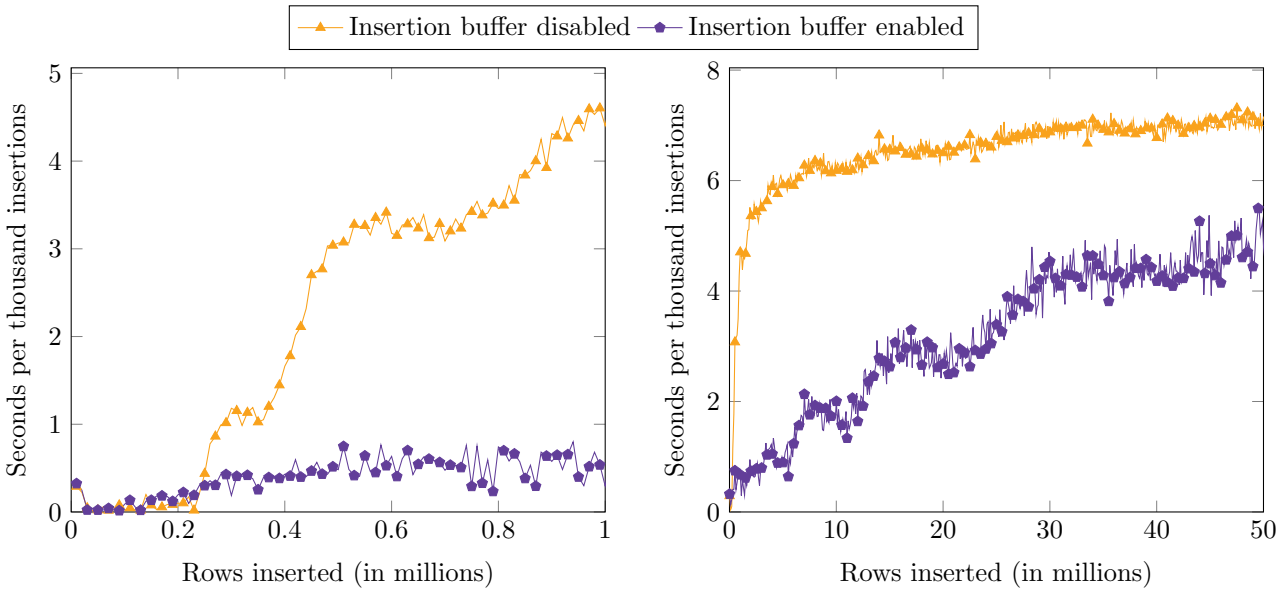
Our analysis explains why insertion buffers can provide dramatic speedups for small databases, but only small gains are available as the database grows. Our results also provide useful guidance to implementers about which flushing strategy will provide the most performance improvement.

Our results also show that insertion buffers cannot deliver the same asymptotic performance improvements that are possible with write-optimized data structures, such as LSM-trees and B^{ϵ}-trees.

6.3 Insertion-Buffer Background This section describes insertion buffers are actually implemented and used in deployed systems and recent research prototypes.

SAP: The SAP IQ database supports an in-memory row-level versioning (RLV) store, and insertions are performed to the RLV store and later merged into the main on-disk store [41].

NuDB: The NuDB SSD-based key-value store buffers all insertions in memory, and later flushes it to SSD [34]. Flushes occur at least once per second, or more often if insertion activity causes the in-memory buffer to fill.



(a) First million insertions

(b) All fifty million insertions

Figure 1: The cost of inserting batches of rows into an empty table in InnoDB with and without the insertion buffer. The rows are inserted in batches of 10,000 to avoid slowdown in parsing, and the keys are distributed uniformly. After 1M insertions, the buffered version takes 12.3% as long as the unbuffered version (measured over 50000 insertions); after 50M insertions, the advantage is reduced so that the buffered version takes 68.3% of the time of the unbuffered. (Lower is better)

Buffered Bloom and quotient filters: Bloom filters are known to have poor locality for both inserts and lookups. The buffered Bloom filter [15] improves the performance of insertions to a Bloom filter on SSD by buffering the updates in RAM. The on-disk Bloom filter is divided into pages, and each page has a buffer of updates in RAM. When a page's buffer fills, the buffered changes are written to the page.

The buffered quotient filter stores newly inserted items in an in-memory quotient filter [6, 7]. When the in-memory quotient filter fills, its entire contents are flushed to the on-disk quotient filter.

InnoDB: The InnoDB [36] B-tree implementation used in the MySQL [37] and MariaDB [24] relational database systems includes an insertion buffer.

Our experiments in this paper focus on InnoDB as an archetypal and open-source implementation of an insertion buffer, so we describe it in detail.

InnoDB structures its insertion buffer as a B-tree. When the insertion buffer becomes full, it selects the items to be flushed by performing a random walk from the root to a leaf. The random walk is performed by selecting, at each step, uniformly randomly from among the children of the current node. Once it gets to a leaf, it it picks a single item to insert into the on-disk B-tree. This item, along with any other items in the insertion buffer that belong in that leaf, are inserted into the leaf and removed from the insertion buffer.

InnoDB's insertion buffer is complicated in several ways. First, the size of the insertion buffer changes over time, as InnoDB allocates more or less space to other buffers and caches.

InnoDB also has a leaf cache. Whenever a leaf is brought into cache for any reason, all inserts to that leaf that are currently in the insertion buffer are immediately applied to the leaf, and any future inserts to that leaf also skip the insertion buffer as long as the leaf remains in cache.

Finally, it performs some flushing when the buffer is not full. Roughly every second, InnoDB performs a small amount of background flushing. Moreover, it prematurely flushes its buffer to a leaf when it calculates that such a flush will cause the leaf to split. We hypothesize that this feature exists to simplify the transactional system.

6.4 Leaf Probabilities in B-trees

In Section 5, we established that, on insertion, the leaf probabilities are nearly uniform. We empirically verify this uniformity property by simulating insertions into the leaves of a B-tree. We insert real-valued keys i.i.d. according to uniform, Pareto (real-valued Zipfian) and normal

distributions; the leaves of the B-tree split when they are full, and we measure the ratio of the maximal weight leaf to $1/n$. Lemma 2.1 tells us that this ratio should be asymptotically at most constant, but as Figure 2 shows, our experimental analysis shows further that this constant is generally less than 2. Because leaves generally split in 2, this makes some intuitive sense.

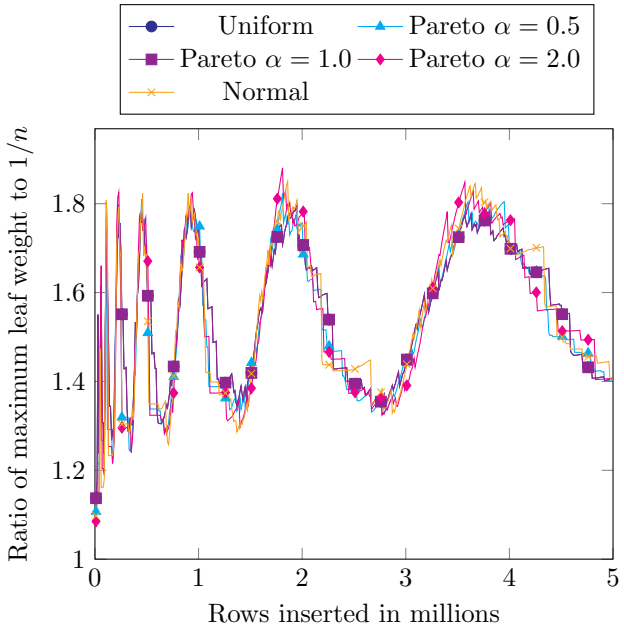
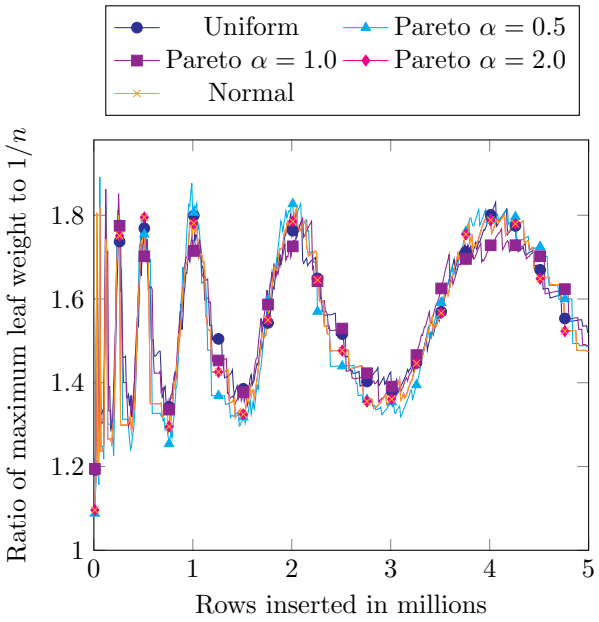


Figure 2: Deviation of the maximum weight leaf from uniform in simulation

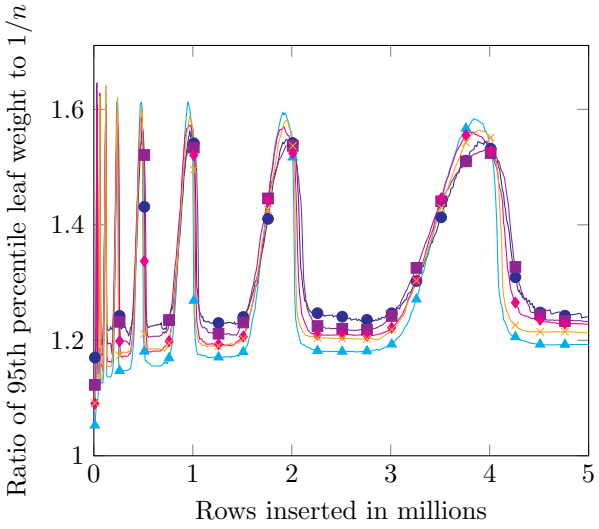
We also verify the these results using the InnoDB storage engine. We insert 5 million rows into a database using uniform, Pareto and normal distributions on the keys. the results are summarized in Fig. 3a. The maximum ratio does not exceed 2.3, and the 95th percentile ratio does exceed 1.6. Thus the distribution of the keys to the leaves is in fact almost uniform.

6.5 Simulating Insertion Buffers The ball-and-bins models described above are based on a static leaf structure. However, in practice inserting into a database causes the leaf structure (the number and probability distribution of bins in the model) to change. However, we can still perform the same strategies, and by simulating an insertion buffer in front of a database, we can compare their efficiency as well as verify that much of the static analysis empirically applies to the dynamic system.

We insert real-valued keys into the simulation according to one of several distributions of varying skewness: uniform on $[0, 1000]$, Pareto with parameter $\alpha = \{0.5, 1.0, 2.0\}$, and uniform centered at 0, with standard



(a) Maximum leaf weight relative to uniform



(b) 95th percentile leaf weight relative to uniform

Figure 3: Deviation of the maximum and 95th percentile weight leaves from uniform as observed in InnoDB.

deviation 1000. We have a buffer which stores 2,500 keys; when it fills we choose a leaf according to the chosen strategy and flush all the buffered keys destined to it. Initially we have one leaf, and the leaves split when they exceed 160 keys, as uniformly as possible.

As shown in Figure 4, the key distribution doesn't affect the recycle rate of the insertion buffer, and as the number of leaves gets larger, the recycle rate decreases. Generally fullest bin does better than golden gate, and golden gate does better than random ball. Demonstrated with the normal distribution (all distri-

butions perform very similarly), Figure 4f shows that golden gate initially outperforms random ball by about 30%, which then decreases as the number of bins grows.

6.6 Real-World Performance (InnoDB) In this section, we empirically the performance of insertion buffers in InnoDB, the default storage engine in MySQL.

Analogously to the experiments in Section 6.5, we insert rows into the MySQL database, and after every 10000 insertions, we check the "merge ratio" reported by InnoDB. This is the number of rows merged into the database from the buffer during each buffer flush, and corresponds to the recycling rate in the balls and bins model. We also check the reported memory allocated to the buffer, which allows us to control for memory usage.

The keys of the rows are i.i.d. according to the same real-valued probability distributions as in Section 6.5: uniform on $[0, 1000000]$, Pareto with parameter $\alpha = \{0.5, 1, 2\}$, and normal centered at 0 with standard deviation 1000. The results for the different distributions are shown in Figures 5a to 5e. The structure of the plot generally does not depend on the key distribution, and while there is more noise, the overall picture is similar to the plots in Figure 4.

If we were to hold the number of leaves roughly constant and change the buffer size, Lemma 5.1 suggests that the relationship with recycle rate would be roughly linear. To test this, we ran the above experiment with buffer sizes from 8mb to 128mb in 2mb increments. We performed 11 million insertions with uniformly distributed keys each time, and then took the average recycle rate for the last million rows. As demonstrated in Figure 5f, the resulting plot is approximately linear.

7 Related Work

Balls-and-bins games are one of the most studied models in all of computer science, so it would be impossible to do justice to the entire literature here. Rather, we focus on prior work on ***dynamic*** balls-and-bins games. In dynamic balls-and-bins games, balls are added and removed from bins according to some rules, and the goal is to understand the long-term behavior of the system. Thus, ball-recycling games are instances of this general class, although previously studied dynamic balls-and-bins games are quite different, and are typically used to study load-balancing problems, rather than throughput.

Previously studied models differ from ball-recycling games in several ways:

- The process for removing balls is assumed to be random, e.g. a random ball is removed, or a random bin is selected for emptying. Prior research has assumed these events are determined by some external process, so they have not studied algorithms for controlling this process. As a result, although the theory of Markov chains has played a major role in the study of balls-and-bins games, the theory of Markov decision processes, which we use extensively, has not shown up at all.

- The balls are thrown uniformly randomly. This is a natural assumption for hashing and randomized load-balancing problems, but is not appropriate when studying updates to existing keys in a database.

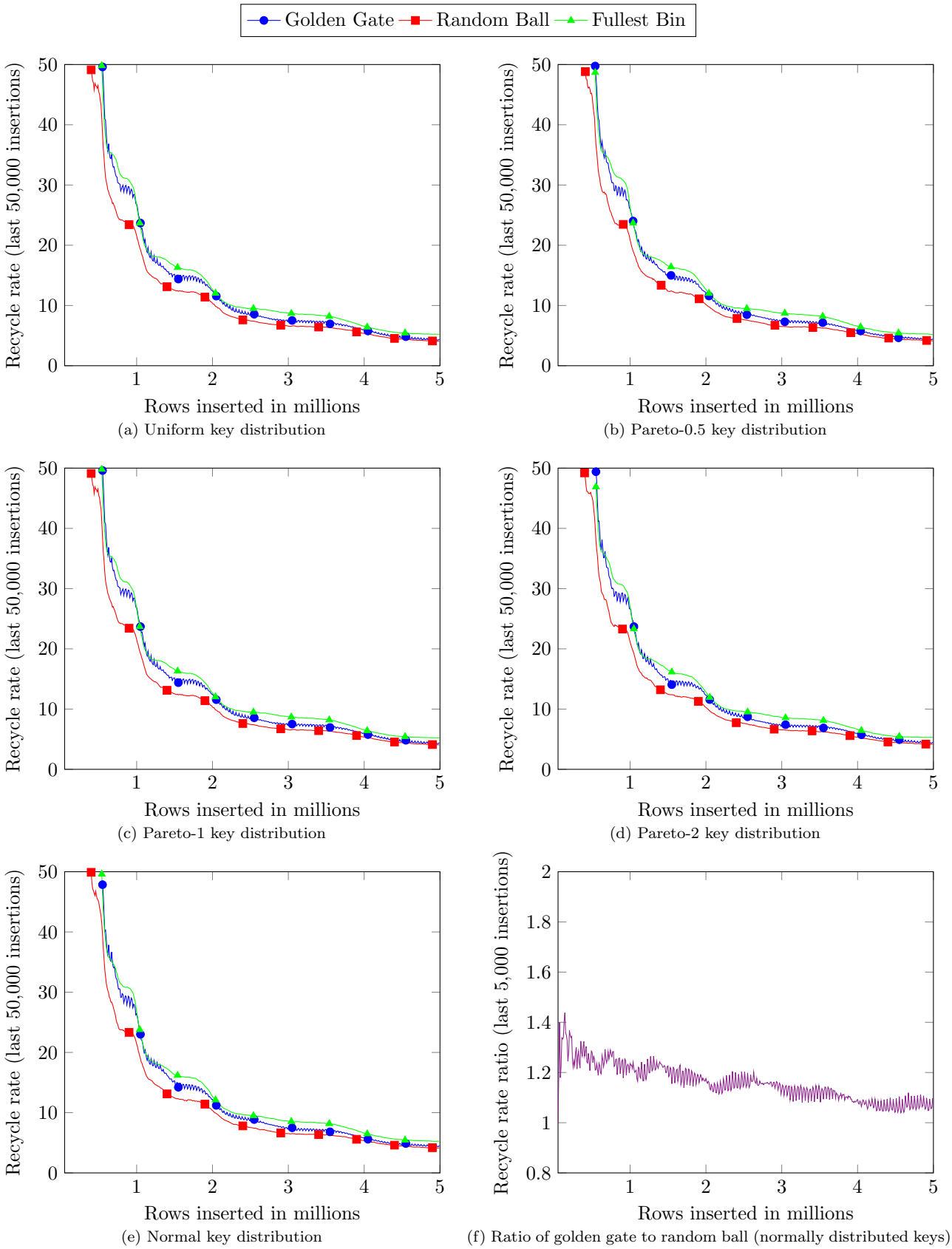
- The objective is to analyze the occupancy of the fullest bin or, in some models, to analyze the amount of time that balls wait in a queue. Our objective is to analyze the number of balls recycled in each time step.

- The number of balls in the system is not fixed. Prior models were used for load balancing, in which balls correspond to tasks and bins resources, so it makes sense to model new balls entering the system asynchronously. In our setting, the balls are the resource (i.e. they correspond to slots in a buffer), so they are fixed.

- The study of dynamic balls-and-bins games was introduced simultaneously with the study of the power of multiple choices [3], and most past work on dynamic balls-and-bins games has been in the same model. In our model, balls have only a single choice (i.e. their on-disk location), although it may be possible to extend our work to systems in which each ball has multiple choices (e.g. for an insertion buffer for an on-disk cuckoo hash table).

Azar, et al. [3] introduced both the power of two choices and dynamic balls and bins games. They showed that if, at each time step, a random ball is rethrown with d uniformly random choices, then, in n^3 time steps, the fullest bin has $\ln \ln n / \ln d + O(1)$ balls w.h.p.

In his dissertation [33], Mitzenmacher studied several dynamic load-balancing problems. This included several variants on the supermarket model, in which balls arrive according to a Poisson process and enqueue themselves in the shortest of d queues that they select uniformly randomly from n queues. Mitzenmacher showed that $d > 1$ exponentially reduced the average time a ball spent in a queue. He also studied a variant in which, at each time step, one ball was removed from one queue and was immediately re-enqueued according to the above procedure. Adler, et al. [1] studied a variant of the supermarket model in which balls arrived in batches of size m and chose their queues in parallel, and



(a) Uniform key distribution

(b) Pareto-0.5 key distribution

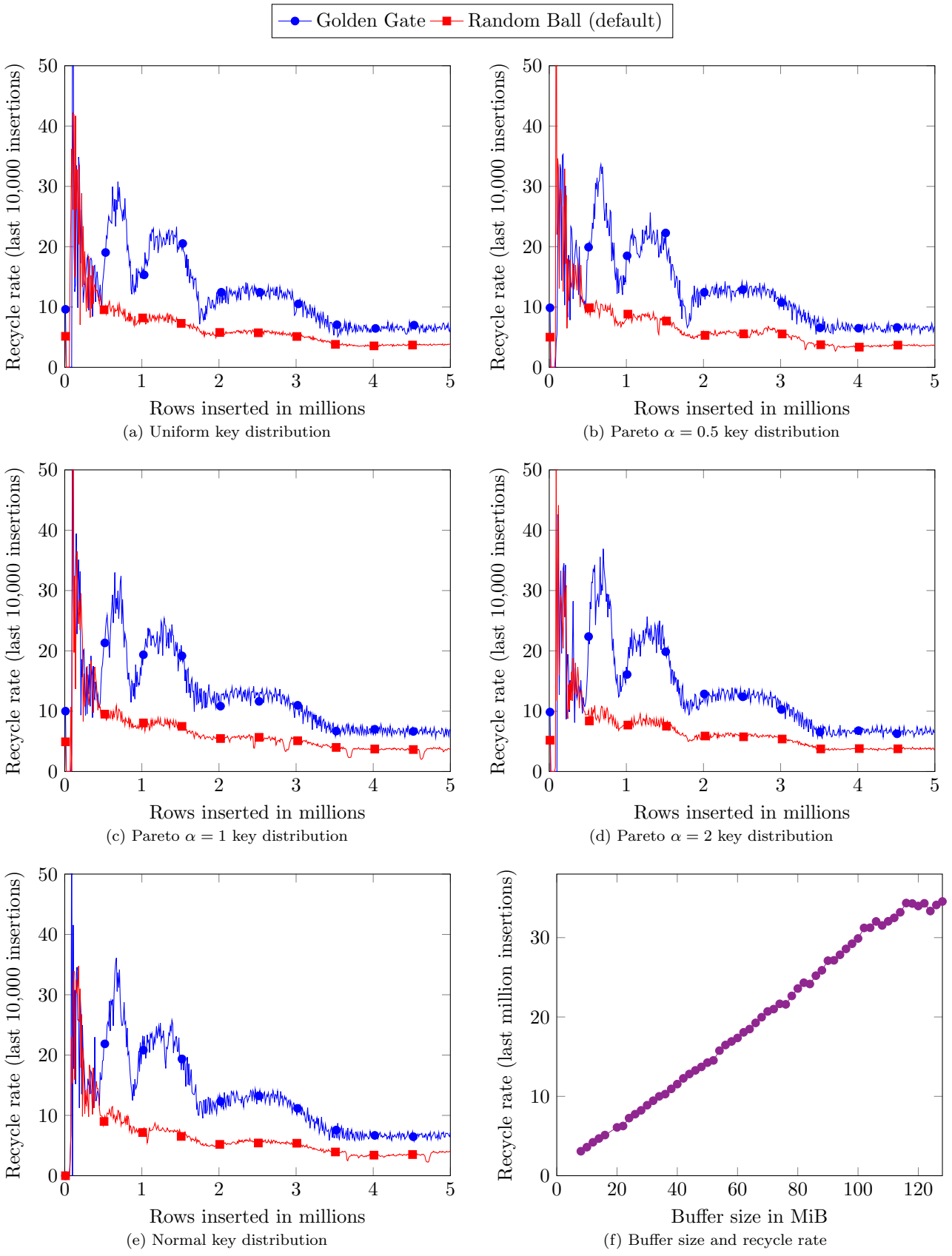
(c) Pareto-1 key distribution

(d) Pareto-2 key distribution

(e) Normal key distribution

(f) Ratio of golden gate to random ball (normally distributed keys)

Figure 4: Simulated results with various key distributions and recycling strategies. (Higher is better)



(a) Uniform key distribution

(b) Pareto $\alpha = 0.5$ key distribution

(c) Pareto $\alpha = 1$ key distribution

(d) Pareto $\alpha = 2$ key distribution

(e) Normal key distribution

(f) Buffer size and recycle rate

Figure 5: InnoDB Insertion buffer recycle rates for various key distributions and memory sizes. (Higher is better)

showed that the average waiting time remains $O(\ln \ln n)$ as long as m is sufficiently smaller than n.

Cole, et al. [16] studied a model in which balls are recycled one-at-a-time according to a recycling plan chosen in advance. In their model, there are an infinite number of labeled balls, and the adversary specifies in advance two ball IDs for each time step: the first ID specifies a ball to be removed from the system, and the second ID specifies a ball to be inserted. Thus the number of balls currently in the system is always n. The first time a ball is inserted, it chooses d bins uniformly at random and picks the least loaded. From then on, whenever that ball reenters the system, it always goes into that bin. In their model, the adversary cannot examine the state of the current system when deciding which ball to recycle, as in our model. Given this restriction, they show that the fullest bin has roughly $\ln \ln n / \ln d$ balls. Vöcking [45] showed the shocking result that, by choosing bins non-uniformly and breaking ties asymmetrically, the max load could be reduced to $\ln \ln n / d \ln \phi_d + O(1)$ w.h.p.

Cole, et al. [17] extended their results to routing through a network: an adversary specified in advance the start time, end time, source, and destination of flows, and the system used a power of two choices variant of Valiant's randomized routing paradigm [43] to limit congestion to $O(\ln \ln n)$ w.h.p.

Czumaj and Stemann [20] study a load balancing problem in which, at each time step, a random ball is removed from the system and a new ball is thrown using d choices. After the new ball is inserted, the d bins examined during its insertion are rebalanced. Surprisingly, the max load is still $O(\ln \ln n)$.

References

[1] M. Adler, P. Berenbrink, and K. Schröder. Analyzing an infinite parallel job allocation process. In *ESA '98*.

[2] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen. Parallel randomized load balancing. *Random Structures & Algorithms*, 13(2):159–188, 1998.

[3] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced allocations. In *STOC '94*, pages 593–602.

[4] M. Azure. How to use batching to improve SQL database application performance. `https://docs.microsoft.com/en-us/azure/sql-database/sql-database-use-batching-to-improve-performance`, 2016.

[5] M. A. Bender, M. Farach-Colton, J. T. Fineman, Y. Fogel, B. Kuszmaul, and J. Nelson. Cache-oblivious streaming B-trees. In *SPAA*, pages 81–92, 2007.

[6] M. A. Bender, M. Farach-Colton, M. Gowami, R. Johnson, S. McCauley, and S. Singh. Bloom filters, adaptivity, and the dictionary problem. In *FOCS '18*.

[7] M. A. Bender, M. Farach-Colton, R. Johnson, R. Kraner, B. C. Kuszmaul, D. Medjedovic, P. Montes, P. Shetty, R. P. Spillane, and E. Zadok. Dont thrash: How to cache your hash on flash. *PVLDB*, 5(11):1627–1637, 2012.

[8] I. Benjamini and Y. Makarychev. Balanced allocation: Memory performance tradeoffs. *Ann. Appl. Probab.*, 22(4):1642–1649, 08 2012.

[9] P. Berenbrink, T. Friedetzky, P. Kling, F. Mallmann-Trenn, L. Nagel, and C. Wastell. Self-stabilizing balls & bins in batches: The power of leaky bins [extended abstract]. In *PODC*, pages 83–92, 2016.

[10] K. Bringmann, T. Sauerwald, A. Stauffer, and H. Sun. Balls into bins via local search: Cover time and maximum load. *Random Structures & Algorithms*, 48(4):681–702, 2016.

[11] G. S. Brodal, E. D. Demaine, J. T. Fineman, J. Iacono, S. Langerman, and J. I. Munro. Cache-oblivious dynamic dictionaries with update/query tradeoffs. In *SODA '10*.

[12] G. S. Brodal and R. Fagerberg. Lower bounds for external memory dictionaries. In *SODA '03*.

[13] M. Callaghan. An obscure performance problem with the insert buffer. `https://www.facebook.com/notes/mysql-at-facebook/an-obscure-performance-problem-with-the-insert-buffer/479735920932/`, 2010.

[14] M. Callaghan. Something awesome in InnoDB – the insert buffer. `https://www.facebook.com/notes/mysql-at-facebook/something-awesome-in-innodb-the-insert-buffer/492969385932/`, 2011.

[15] M. Canim, C. A. Lang, G. A. Mihaila, and K. A. Ross. Buffered bloom filters on solid state storage. In *ADMS*, 2010.

[16] R. Cole, A. Frieze, B. M. Maggs, M. Mitzenmacher, A. W. Richa, R. Sitaraman, and E. Upfal. On balls and bins with deletions. In *RANDOM '98*.

[17] R. Cole, B. M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A. W. Richa, K. Schröder, R. K. Sitaraman, and B. Vöcking. Randomized protocols for low-congestion circuit routing in multistage interconnection networks. In *STOC*, pages 378–388, 1998.

[18] A. Conway, A. Bakshi, Y. Jiao, W. Jannen, Y. Zhan, J. Yuan, M. A. Bender, R. Johnson, B. C. Kuszmaul, D. E. Porter, and M. Farach-Colton. File systems fated for senescence? nonsense, says science! In *FAST '17*.

[19] A. Conway, M. Farach-Colton, and P. Shilane. Optimal Hashing in External Memory. In *ICALP '18*.

[20] A. Czumaj and V. Stemann. Randomized allocation processes. In *FOCS '97*, pages 194–203.

[21] P. Deheuvels and L. Devroye. Strong laws for the maximal k-spacing when $k \leq c \log n$. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 66(3):315–334, 1984.

[22] J. Esmet, M. A. Bender, M. Farach-Colton, and B. C. Kuszmaul. The tokufs streaming file system. In *Hot Storage '12*.

[23] M. Farach-Colton, R. J. Fernandes, and M. A.

Mosteiro. Bootstrapping a hop-optimal network in the weak sensor model. *ACM Trans. Algorithms*, 5(4):37:1–37:30, Nov. 2009.
[24] M. Foundation. MariaDB Foundation, 2017. https://mariadb.org.
[25] IBM. Buffered inserts in partitioned database environments. https://www.ibm.com/support/knowledgecenter/SSEPGG_10.5.0/com.ibm.db2.luw.apdv.embed.doc/doc/c0061906.html, 2017.
[26] I. Informix. Understanding SQL insert cursors.
[27] W. Jannen, J. Yuan, Y. Zhan, A. Akshintala, J. Esmet, Y. Jiao, A. Mittal, P. Pandey, P. Reddy, L. Walsh, M. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. BetrFS: A right-optimized write-optimized file system. In *FAST '15*.
[28] W. Jannen, J. Yuan, Y. Zhan, A. Akshintala, J. Esmet, Y. Jiao, A. Mittal, P. Pandey, P. Reddy, L. Walsh, M. A. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. BetrFS: write-optimization in a kernel file system. *Trans. Storage*, 11(4):18:1–18:29, Nov. 2015.
[29] L. Kallenberg. Markov decision processes. 2016.
[30] M. Mitzenmacher. Load balancing and density dependent jump markov processes. In *FOCS '96*.
[31] M. Mitzenmacher. On the analysis of randomized load balancing schemes. In *SPAA '98*, pages 292–301.
[32] M. Mitzenmacher, A. W. Richa, and R. Sitaraman. The power of two random choices: A survey of techniques and results. In *Handbook of Randomized Computing*, pages 255–312. Kluwer, 2000.
[33] M. D. Mitzenmacher. *The Power of Two Choices in Randomized Load Balancing*. PhD thesis, Citeseer, 1996.
[34] NuDB. Nudb: A fast key/value insert-only database for ssd drives in c++11. https://github.com/vinniefalco/NuDB, 2016.
[35] Oracle. Tuning the database buffer cache. https://docs.oracle.com/database/121/TGDBA/tune_buffer_cache.htm#TGDBA294, 2017.
[36] Oracle, Inc. Introduction to innodb, 2017. https://dev.mysql.com/doc/refman/5.6/en/innodb-introduction.html.
[37] Oracle, Inc. MySQL, 2017. https://www.mysql.com.
[38] P. ONeil, E. Cheng, D. Gawlick, and E. ONeil. The log-structured merge-tree (lsm-tree). *Acta Informatica*, 33(4):351–385, 1996.
[39] G. Park. A generalization of multiple choice balls-into-bins: Tight bounds. *Algorithmica*, 77(4):1159–1193, Apr 2017.
[40] M. Petrova, N. Olano, and P. Mähönen. Balls and bins distributed load balancing algorithm for channel allocation. In *WONS '10*.
[41] SAP. Rlv data store for write-optimized storage. http://help-legacy.sap.com/saphelp_iq1611_iqnfs/helpdata/en/a3/13783784f21015bf03c9b06ad16fc0/content.htm, 2017.
[42] Tokutek, Inc. TokuDB and TokuMX, 2014. http://www.tokutek.com.
[43] L. G. Valiant and G. J. Brebner. Universal schemes for parallel communication. In *STOC '81*, pages 263–277.
[44] Vertica. Wos (write optimized store). https://my.vertica.com/docs/7.1.x/HTML/Content/Authoring/Glossary/WOSWriteOptimizedStore.htm, 2017.
[45] B. Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4):568–589, July 2003.
[46] R. West, P. Zaroo, C. Waldspurger, X. Zhang, and H. Zheng. Online computation of cache occupancy and performance, July 19 2016. US Patent 9,396,024.
[47] J. Xiang. Apache hbase write path. http://blog.cloudera.com/blog/2012/06/hbase-write-path/, 2012.
[48] J. Yuan, Y. Zhan, W. Jannen, P. Pandey, A. Akshintala, K. Chandnani, P. Deo, Z. Kasheff, L. Walsh, M. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. Optimizing every operation in a write-optimized file system. In *FAST '16*, pages 1–14.
[49] J. Yuan, Y. Zhan, W. Jannen, P. Pandey, A. Akshintala, K. Chandnani, P. Deo, Z. Kasheff, L. Walsh, M. Bender, M. Farach-Colton, R. Johnson, B. C. Kuszmaul, and D. E. Porter. Optimizing every operation in a write-optimized file system. *Trans. Storage*, 13(1), 2017.