Seminar Advanced Algorithms and Data Structures: Optimal Ball Recycling[1]

Advisor: Dr. Przemysław Uznański

Buddy: Shengyu Huang

Orhan Saeedi

10.11.2018

Introduction

Balls-and-bins games are one of the most studied models in all of computer science. A **balls-and-bins game** or **balls-into-bins problem** involves m balls and n bins. Each time, a single ball is thrown into one of the bins. After all balls are thrown, we look at the number of balls in each bin[6]. These games are very popular when it comes to study how to distribute load evenly across the resource being allocated. They are used to analyse the average and worst-case occupancy of buckets in a hash-table, the worst-case load on nodes in a distributed cluster, and even the amount of time customers wait in line at the grocery store. But there are a few load distribution problems which can't be modelled with the simple balls-and-bins game. Let us look into one such problem the paper provides. A common data structure used for storing data into databases is the so called B**-Trees** defined as follows[2][5]:

Definition 1 (B**-Tree**)**.** *A self balancing multi-way tree data structure, which is common for relatively large blocks of data. Each leaf has at most B elements, internal nodes (except the root) have degree $\Theta(B)$ elements, and all leaves have equal depth.*

A B-Tree is just a generalization of a binary search tree, since a binary search tree is a B-Tree with $B = 1$. When we insert an item we propagate down till we find the leave where the element belongs. If the number of elemnts is greater than B (which violates one rule of B-trees) the tree structure hast to be renwed in a certain way, which is not important for this talk. We don't go into detail of the other operations like deletions either since it is not relevant for this paper. But instead we will define a modified B-Tree called **Buffered** B**-Tree**.[2]

Definition 2 (**Buffered** B**-Tree**)**.** *A B-**Tree** with each node maintaining a buffer of insertions and updates to its subtree. An **insertion buffer** is a cache of recently inserted key-value pairs. An **update buffer** on the other hand is a cache of changes to existing items. Delayed updates in these buffers are only propagated down to the children's buffer whenever a node's buffer overflows.*

In Figure 1 we see an example of a Buffered B-Tree, which was somehow obtained. Now let us say the buffer is empty and we want to add the elements $1, 5, 7$ and 13.
The paper inserted batches of row into an empty table running with B-Trees with and without insertion buffers. After 1 million insertions the buffered version takes 12.3% as long as the unbuffered version. This shows that Buffers are indeed really helpful. But why do we need to study them if they already are good? The paper also tested for even bigger numbers. After 50 million insertions the advantage is reduced. The

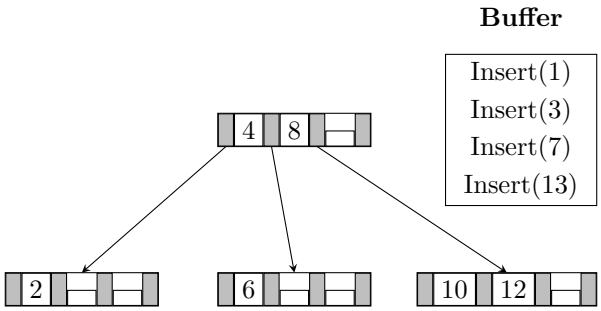


Figure 1: An example of a buffered B-Tree with an insertion buffer, where we are just interested in the insertion buffer of the root.

buffered version takes 68.3% of the time of the unbuffered version.
Because of that our goal has to be to maximize the number of items removed from the buffer in each flush. We already see that the balls-and-bins game works for one round of filling the buffer of a node, but what do we do if the node's buffer becomes full like in Figure 1? For this we will look at a modified game, which we will refer as the **ball recycling game**.

Definition 3 (**Ball Recycling Game**). *Throw m balls into n bins i.i.d according to a given probability distribution 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 p.*

This game models insertion buffers and update buffers. Disk blocks of buffers are bins and elements in the insertion/update buffer are balls. The probability distribution p is analogous to the observed probability of items inserted or updated. Evicting all the items in a a disk block is the same as 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 bin-picking method.
We want to find a **recycling strategy**/bin-picking method, which has a high as possible expected number of balls recycled, which we will call the **recycling rate**. The higher the recycling rate of a certain recycling strategy, the better the speed-up of such an insertion buffer.
In Figure 2 we have an example with $n = 7$ bins and $m = 10$ balls, which all are thrown in the moment of the first picture. Let the distribution p be defined as follows. Let $p_i = \frac{1}{10}$ if i is odd and else let $p_i = \frac{1}{5}$ i.e. the bin has an even number. In the figure we reached a part of the game where all balls are already thrown i.e., the node's buffer is full. It picks a non-empty bin (in this case bin 3) and **recycles** its balls (shown in red). The re-thrown balls land in bin 1, 3, and 5 (shown in blue). This happens with probability $\frac{1}{10}$ each. This also shows, that it is also possible to throw the balls into bin 3 again.
We will look at three different and very natural recycling strategies given Figure 2.

- **Fullest Bin** *A greedy strategy that recycles the bin with the most balls.* In Figure 2 we would exactly pick bin 3, because it has at least one more ball than all other bins.
- **Random Ball**: *A strategy that picks a ball uniformly at random and recycles its bin.* In this strategy it is important to notice that we don't directly pick a bin. So the scenario of Figure 2 occurs with a probability of $\frac{3}{m} = \frac{3}{10}$.
- **Golden Gate**: *A strategy that picks the bins in round-robin fashion; after a bin is picked, the next bin is its non-empty successor.* We can also state a important property of this strategy with Figure 2. Assuming, we recycled bin 2 in the last time step. We will have to recycle bin 3 in the current time step, but in the next time step, we will have to recycle bin 5, since bin 4 has no balls in it.

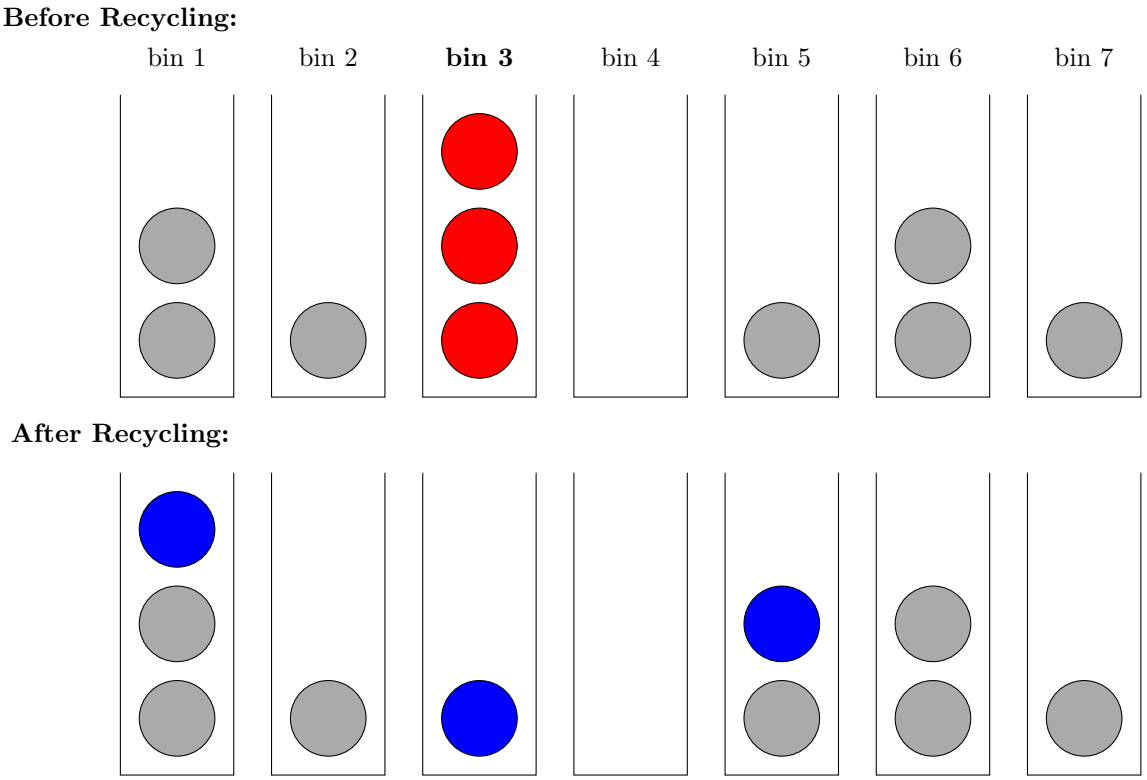


Figure 2: An example of a ball recycling game with $n = 7$ bins and $m = 10$ balls

You might think that the **Fullest Bin** is always optimal for the general distribution. But there are distributions for which **Fullest Bin** is pessimal. This means that it recycles at most 2 balls per round whereas the optimal recycling strategy recycles almost m balls per round. Let us look into such an example.

Example 1. *The **skyscraper distribution** is defined as follows:* $p_0 = 1 - \frac{1}{n} + \frac{1}{n^2}$ *and* $p_i = \frac{1}{n^2}$*, for* $0 < i \leq n - 1$*. Suppose that* $m < n$ *Now with **Fullest Bin** we will almost every time step pick bin 0. The number of the balls in the other bins will in expectation always be incremented in every nth step. We will choose bin 0 until it has 1 ball in it, at which point it will pick another bin, which will almost certainly have 1 ball in it. After this point, we will always either almost certainly recycle 2 (bin 0) or 1 (the rest of the bins) ball. Our recycle rate will slowly drop below 2. You see that pretty well in Figure 3 When we instead recycle the least-full non empty bin we will avoid that balls stay to long in a bin with low probability and have a recycling rate of nearly* m*.*

We will see that the strategies **Fullest Bin** and **Golden Gate** are optimal for the uniform distribution, which we denote as u and **Random Ball** is optimal for the general distribution, which we will denote as p.

Short Introduction to Markov Chains[3][4]

You all are familiar with the so called finite automata. Systems are often modelled by automata, and discrete events are transitions from one state to another. If you want to analyse such discrete event systems, you assume that the events are stochastic, and you want to know how your system behaves on average. If the events happen in discrete time (for example, there is an event every hour), the tool to model the system is called Discrete Time Markov Chains. This is not quite what we need, since our recycling events are not based on some time interval. So we define following

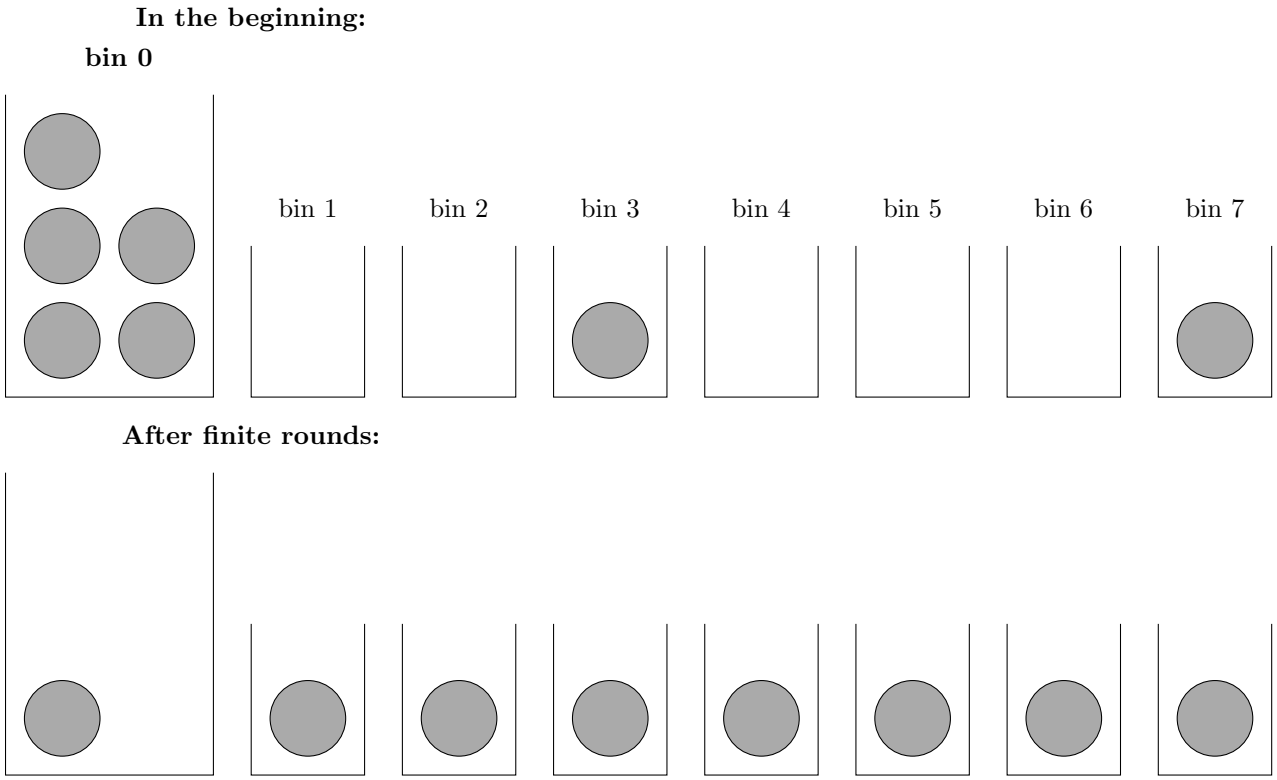


Figure 3: How the **Fullest Bin** on the skyscraper distribution would result.

Definition 4 (**Continuous Time Markov Chain, Deterministic**). *Let S be a finite or countably set of states. A **Continuous Time Markov Chain** is a continuous time stochastic process $\{X_t : t \in \mathbb{R}_{\geq 0}\}$ with $X_t \in S$ for all t that satisfies the continuous Markov property.*
*A Markov chain satisfies the **Markov property, deterministic** if the probability for the next state depends only on the current state, and not the history.*

We will need **Continuous Time Markov Chains** for this report/presentation, since in our ball recycling game events (transitions) happen in arbitrary times, to be exact, whenever the node's buffer is full. Now we go on with mapping our ball recycling game onto this newly learned Markov Chains. In a ball recycling game, we represent the configuration of the balls as a vector X of length n (number of bins), 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/states. Intuitively we see that ball recycling games and a recycling strategy are Markov decision processes. Let λ_i be some distribution of state i describing the time the process stays in state i and further let $\lambda_{j,i}$ be the transition rate from state i to j. We now further introduce an important property of deterministic, stateless Markov Chains for finite-state distributions:

Definition 5 (**Stationary Distribution**). *For $t \to \inf$, π is a **stationary distribution** if for all $i \in S$,*

$$\sum_{j:j\neq i} \pi_j \cdot \lambda_{j,i} - \pi_i \cdot \lambda_i = 0. \tag{1}$$

*The sum on the LHS describes the **net flow** of a Markov process.*

We know further add that the three recycling strategies we learned are finite-state recycling strategies, and therefore the property of the stationary distribution works for ball recycling games. We will use following lemma without a proof:

Lemma 1. *There exists a deterministic recycling strategy OPT that achieves the optimal expected average recycling rate.*

We could prove this with the fact, that OPT has an unique stationary distribution i.e., has a net flow of zero. The proof follows from Kallenberg's Corollary 5.4[3]. Now we just have to define some notations and we can start the really interesting part.

Basic Definitions

m	The number of balls.
n	The number of bins.
p	General distribution.
u	Uniform distribution.
p_i	Probability that some thrown ball lands in bin i, where $i \leq n$.
R^A	The recycling rate of some recycling strategy A.
RB	Short for the strategy **Random Ball**
FB	Short for the strategy **Fullest Bin**
GG	Short for the strategy **Golden Gate**
$\|p\|_{\frac{1}{2}}$	$= \left(\sum_{i=1}^{n} \sqrt{p_i}\right)^2$, the half quasi-norm of p

Main upper Bound for all recycling strategies

Let's get started. First we want to get a main upper bound for some strategy A with unique stationary distribution. Let p be some general distribution on the bins. We first introduce a lemma, which is very important for stating proofs for this kind of balls-and-bins game and follows directly from our observations on Markov Chains. Let ϕ_i be the event that the strategy A picks bin i. Let $R_i^A = \mathbb{E}\left[R^A | \phi_i\right]$ be the number of balls recycled given that the strategy A picks bin i and $f_i = P\left(\phi_i\right)$, the probability of picking bin i in the stationary distribution. The so called **flow equation** is defined as follows:

Lemma 2 (**Flow Equation**)**.** *Let A be a recycling strategy with a unique stationary distribution for a ball recycling game with n bins with probabilities p. Then, for all $0 \leq i < n$,*

$$p_i R^A = f_i R_i^A.$$

Proof. We use Defintion 5 for proving this Lemma. We see that for Ball recycling games we have that $\lambda_i = f_i$ for $i \in [n]$ and following special case on the transition rate: $\lambda_{k,i} = \lambda_{l,i} = p_i$ for $k, l, i \in [n]$. This means no matter which bin was recycled before, the probability of throwing a ball in a certain bin does not change. Further on from the assumption we take that A is a strategy with a unique stationary distribution. So the Equation 1 can be rewritten as follows:

$$\sum_{j:j \neq i} p_i R_j^A - f_i R_i^A = 0.$$

Rearranging gives us exactly the **Flow Equation** □

In other words, when R^A balls are thrown, $p_i R^A$ of them are expected to land in bin i. Now this ball can only leave bin i if bin i is picked by A, at which point all balls in bin i will be evicted. Because A has an unique stationary distribution, the net flow must be zero.

The following Lemma 3 has a longer rather mathematical prove, which needs the introduction of three other lemmas before you can prove it. For this reason, we won't prove the lemma. Instead an intuition is given to the lemma, which is probably a more helpful variant to understand it.
If we assume we have a reasonable recycling strategy (not all strategies satisfy this following assumption). Then if the number of balls X_i in bin i grows, the recycling rate of the other bins will go down, because the X_i balls aren't available for recycling, until bin i is selected. This means that most of the time we have a rather high number of balls X_i in bin i, because the less balls are available to throw into this bin i. If we assume this intuition as fact for the moment, this suggests that the expected number of balls $\mathbb{E}[X_i]$ in bin i should be greater than half the recycling rate of bin i, perhaps excluding the last ball to land in the bin. Meaning

$$\mathbb{E}[X_i] \geq \frac{1}{2}\left(R_i^A - 1\right) = \frac{1}{2}\left(\frac{p_i}{f_i}R^A - 1\right),$$

where in the equality the Flow Equation (Lemma 2) is used. Now we sum this over i and use that the sum over all expected number is equal to the number of balls in the game m.

$$\begin{aligned}\sum_{i=1}^{n}\mathbb{E}[X_i] &\geq \sum_{i=1}^{n}\frac{1}{2}\left(\frac{p_i}{f_i}R^A - 1\right)\\ m &\geq \frac{1}{2}\left(R^A\sum_{i=1}^{n}\left(\frac{p_i}{f_i}\right) - n\right)\end{aligned}$$

After rearranging we obtain following Lemma:

Lemma 3. *Consider a ball-recycling game with m balls, n bins and distribution p. If A is a strategy with a unique stationary distribution that picks bin i with frequency f_i, then its recycle rate is bounded by*

$$R^A \leq \frac{2m+n-1}{\sum_{i=0}^{n}\frac{p_i}{f_i}}.$$

Lemma 3 applies to the optimal deterministic strategy OPT promised by Lemma 1, and we know that $R^A \leq R^{OPT}$ for any recycling strategy A. Thus by maximizing the RHS of Lemma 3, we can get an upper bound on the recycling rate of any recycling strategy.

Lemma 4. *Consider a ball-recycling game with m balls, n bins and distribution p. For any recycling strategy A,*

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

Proof. We want to prove that:

$$\frac{2m+n-1}{\sum_{i=0}^{n}\frac{p_i}{f_i}} \leq \frac{2m+n-1}{\|p\|_{\frac{1}{2}}}. \tag{2}$$

We must have that

$$\begin{aligned}\sqrt{\sum_{i=0}^{n}\frac{p_i}{f_i}} &\leq \sqrt{\sum_{i=0}^{n}p_i}\\ &\leq \sum_{i=0}^{n}\sqrt{p_i},\end{aligned}$$

where the first inequality comes from the fact that $0 < f_i \leq 1$ for all $i \in [1, n]$ and the second inequality is a special case of the Chauchy-Schwartz Inequality. When we square both sides we get

$$\begin{aligned}\sum_{i=0}^{n} \frac{p_i}{f_i} &\leq \left(\sum_{i=0}^{n} \sqrt{p_i}\right)^2 \\ &= \|p\|_{\frac{1}{2}},\end{aligned}$$

□

which directly implies Equation 2.

The Uniform Case

Before continuing with the General Case, we will look into the special case when the bins are uniformly distributed, which models insertion buffers the best. The reason is that we mostly insert items with uniformly-distributed keys. Then I give a short proof on why **Fullest Bin** and **Golden Gate** are optimal. We get following Corollary from Lemma 4

Corollary 1. *Consider a ball-recycling game with m balls, n bins and uniform distribution u. For any strategy A*

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

Theorem 1. ***Fullest Bin*** *and* ***Golden Gate*** *are optimal for the ball recycling game with distribution* $\boldsymbol{u}$ *for any n bins and m balls to within an additive constant. They each achieve a recycling rate of at least $\frac{2m}{n+1}$. Meaning*

$$\begin{aligned} R^{FB} &\geq \frac{2m}{n+1} \\ \textit{i.e. } R^{GG} &\geq \frac{2m}{n+1}, \end{aligned} \tag{3}$$

whereas no recycling strategy can achieve a recycling rate greater than $\frac{2m}{n} + 1$

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 to start from the next one and cycling around. Therefore a ball lands in the $[(n-1)/2]$th bin, due to uniformity. Each ball thrown will therefore sit for an average of less than $(n-1)/2$ round before it is thrown again.
Let T be the random variable denoting the number of balls thrown in a given round with **Fullest Bin**. We can order the bins in order of fullness. Therefore the ball lands in 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 ordered list than down, thus each ball thrown int the system will sit for an average of less than $(n-1)/2$ rounds before it is thrown again.

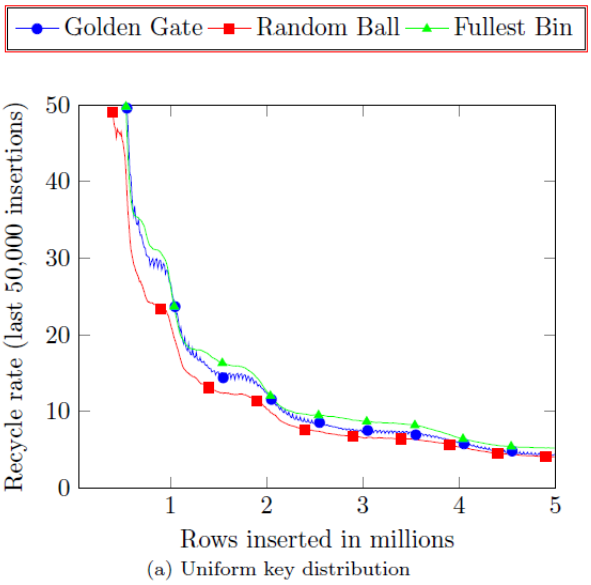
$$\begin{aligned} \frac{\mathbb{E}[S](n-1)}{2} &\geq m - \mathbb{E}[S] \\ \mathbb{E}[S](n-1) &\geq 2(m - \mathbb{E}[S]) \\ \mathbb{E}[S](n+1) &\geq 2m \\ \mathbb{E}[S] &\geq \frac{2m}{n+1}. \end{aligned}$$

analogous for T. And we are done! □

Further on the paper proves why **Random Ball** in the Uniform Case recycles at most $1 + (2 - \epsilon)\frac{m}{n}$ balls per round for some $\epsilon > 0$ and at least $\left(1 + \frac{1}{6^4} - c\right)\frac{m}{n}$. We don't have the time to look into this, but we will show the results from a database experiment presented by the paper.

Example on the Uniform Case

We want to look at an example where we clearly see why we prefer **Fullest Bin** and **Golden Gate** over **Random Ball** in the uniform case. Since the difference is very small on small m. We will look at Database Experiments made by the paper with really high numbers.



(a) Uniform key distribution

Figure 4: Simulated results with various key distributions and recycling strategies. (Higher is better)[1]

In Figure 4 we see that **Random Ball** is always a little bit slower (has a smaller recycling rate) for big numbers.

The General Case

Unlike insertions updates do not change the structure of the B-Tree. Thus, if we buffer updates, then we can have non-uniform leaf/bin-weights. **Update buffers** do therefore match the general case. Also we can not just assume that we will always have uniformly distributed keys. So this is also worth looking at for **insertion buffers**.
In this section, we want to look at why **Random Ball** is $\Theta(1)$-optimal for the general distribution p. As before let R^A be the recycle rate of some strategy A. Further on we the half-quasi norm defined as follows: $\|p\|_{\frac{1}{2}} = \left(\sum_{i=1}^{n} \sqrt{p_i}\right)^2$. We have that

Theorem 2. ***Random Ball*** *is optimal for the ball recycling game with distribution* $\boldsymbol{p}$*. Formally for strategy* RB

1. *If* $m \geq n$,

$$R^{RB} = \Theta\left(\frac{m}{\|p\|_{\frac{1}{2}}}\right). \tag{4}$$

2. *If* $m < n$*, let* L *be the* m *lowest-weight bins,* $q = \sum_{l \in L} p_l$*, and* R_L^{RB} *be the recycle rate of* ***Random***

Ball *restricted to L. Then,*

$$R^{RB} = \Theta\left(\min\left(R_L^{RB}, \frac{1}{q}\right)\right). \tag{5}$$

We won't look at the whole proof due to time reasons.
We already looked at why that no recycling strategy has a recycling rate exceeding $\frac{2m+n-1}{\|p\|_{\frac{1}{2}}}$ (main upper bound). We will bind **Random Ball** additionally from below so that it matches that bound for $m \geq n$ leading to Equation 4. The paper additionally introduces a new recycling strategy **Aggressive Empty**, which is optimal for $m < n$. At the end they compare **Random Ball** to **Aggressive Empty**, which then implies that **Random Ball** is also optimal for $m < n$ (Equation 5), concluding the whole theorem.

Proof of Theorem 2 for $m \geq n$

Proof. **Case** $m \geq n$
Let X_i^{RB} be the random variable of the number of balls in bin i in the stationary distribution of **Random Ball**. **Random Ball** recycles bin i with probability $\frac{X_i^{RB}}{m}$, and therefore the expected number of balls recycled from bin i per round is $\frac{X_i^{RB}}{m} \cdot \mathbb{E}\left[X_i^{RB}\right] = \frac{\mathbb{E}\left[\left(X_i^{RB}\right)^2\right]}{m}$. The number of balls that land in bin i per round is $p_i \sum_{j=1}^{n} \frac{\mathbb{E}\left[\left(X_j^{RB}\right)^2\right]}{m}$. Since X is distributed stationarily, we must have

$$p_i \sum_{j=1}^{n} \frac{\mathbb{E}\left[\left(X_j^{RB}\right)^2\right]}{m} = \frac{\mathbb{E}\left[\left(X_i^{RB}\right)^2\right]}{m} \geq \frac{\mathbb{E}\left[X_i^{RB}\right]^2}{m}$$

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

$$\sum_{i=1}^{n} \sqrt{p_i \sum_{j=1}^{n} \mathbb{E}\left[\left(X_j^{RB}\right)^2\right]} \geq \sum_{i=1}^{n} \mathbb{E}\left[X_i^{RB}\right]$$

$$\sqrt{\sum_{j=1}^{n} \mathbb{E}\left[\left(X_j^{RB}\right)^2\right]} \sum_{i=1}^{n} \sqrt{p_i} \geq m$$

wher we used that $\sum_{i=1}^{n} \mathbb{E}\left[X_i^{RB}\right] = m$. We will square both sides again and obtain the expected recycle rate:

$$\sum_{j=1}^{n} \mathbb{E}\left[\left(X_j^{RB}\right)^2\right] \left(\sum_{i=1}^{n} \sqrt{p_i}\right)^2 \geq m^2$$

$$\sum_{j=1}^{n} \frac{\mathbb{E}\left[\left(X_j^{RB}\right)^2\right]}{m} \geq \frac{m}{\left(\sum_{i=1}^{n} \sqrt{p_i}\right)^2}$$

$$R^{RB} \geq \frac{m}{\|p\|_{\frac{1}{2}}}$$

which proves the Equation 4. □

Conclusion

We've come to the end and seen how modifying the classic **balls-and-bins game** can model a totally new problem in Computer Science. **Insertion Buffers and Update Buffers in B-Trees**. We got an intuition

on how using different **recycle strategies** can result in different **recycle rates**. **Random Ball is $\Theta(1)$-optimal**, whereas **Fullest Bin** can be pessimal. However, when $p = u$, in the uniform distribution, **Fullest Bin and Golden Gate are optimal** within an additive constant.
I invested a lot of time in the introduction and the upper bound. The paper proved a lot more stuff. They give a proof to Lemma 3, the 2nd case of Theorem 2 and they prove why Random Ball is not optimal on the uniform case. They also give a ton of results on Database Experiments, which are really interesting to look at, but was unfortunately hard to present on a blackboard.

References

[1] Michael A Bender, Jake Christensen, Alex Conway, Martín Farach-Colton, Rob Johnson, and Meng-Tsung Tsai. Optimal ball recycling. arXiv preprint arXiv:1807.01804, 2018.

[2] Gerth Stolting Brodal and Rolf Fagerberg. Lower bounds for external memory dictionaries. In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, pages 546–554. Society for Industrial and Applied Mathematics, 2003.

[3] Lodewijk Kallenberg. Markov decision processes.

[4] Jochen Seidel Roger Wattenhofer. Chapter 6 - queueing, 2018.

[5] Wikipedia. B-tree — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=B-tree&oldid=866450218`, 2018. [Online; accessed 08-November-2018].

[6] Wikipedia. Balls into bins — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Balls%20into%20bins&oldid=840789406`, 2018. [Online; accessed 08-November-2018].