

Lock-Free Algorithms under Stochastic Schedulers

Dan Alistarh
Microsoft Research
Cambridge
dan.alistarh@microsoft.com

Thomas Sauerwald
Computer Laboratory
University of Cambridge
tms41@cam.ac.uk

Milan Vojnović
Microsoft Research
Cambridge
milanv@microsoft.com

ABSTRACT

In this work, we consider the following random process, motivated by the analysis of lock-free concurrent algorithms under high memory contention. In each round, a new scheduling step is allocated to one of n threads, according to a distribution $\mathbf{p} = (p_1, p_2, \dots, p_n)$, where thread i is scheduled with probability p_i . When some thread first reaches a set threshold of executed steps, it registers a *win*, completing its current operation, and resets its step count to 1. At the same time, threads whose step count was close to the threshold also get reset because of the win, but to 0 steps, being penalized for *almost* winning. We are interested in two questions: how often does *some* thread complete an operation (*system latency*), and how often does a *specific* thread complete an operation (*individual latency*)?

We provide asymptotically tight bounds for the system and individual latency of this general concurrency pattern, for arbitrary scheduling distributions \mathbf{p} . Surprisingly, a simple characterization exists: in expectation, the system will complete a new operation every $\Theta(1/\|\mathbf{p}\|_2)$ steps, while thread i will complete a new operation every $\Theta(\|\mathbf{p}\|_2/p_i^2)$ steps. The proof is interesting in its own right, as it requires a careful analysis of how the higher norms of the vector \mathbf{p} influence the thread step counts and latencies in this random process. Our result offers a simple connection between the scheduling distribution and the average performance of concurrent algorithms, which has several applications.

Categories and Subject Descriptors

E.1 [Data Structures]: Distributed Data Structures; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems

General Terms

Theory

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PODC'15, July 21–23, 2015, Donostia-San Sebastián, Spain.
Copyright © 2015 ACM 978-1-4503-3617-8/15/07 ...\$15.00.
DOI: <http://dx.doi.org/10.1145/2767386.2767430>.

Keywords

Shared memory; lock-free algorithms; scheduling

1. INTRODUCTION

One of the main computing trends of the last decade has been increased pressure on concurrent algorithms and data structures to *scale* under concurrency. *Lock-free (non-blocking)* data structures [15] represent a promising solution to this problem, thanks to their good performance and relatively simple structure. Unlike lock-based approaches, these data structures follow the principle that some operation should always complete every finite number of steps; the blocking of a single execution thread should never prevent the whole system from progressing. Efficient lock-free implementations are now known for many classic data structures, such as counters, stacks, queues, lists, or trees [10, 15, 16, 18, 22, 24].

Recently, there has been increased interest in the analytical properties of lock-free data structures [1, 9, 16]. Herlihy and Shavit [16] suggested that, on realistic schedules, lock-free algorithms ensure that *each* operation completes in a finite number of its own steps, i.e., that lock-free algorithms are in fact *wait-free* [13]. Alistarh, Censor-Hillel, and Shavit [1] extended this idea and gave a simple stochastic condition on schedules under which any lock-free algorithm becomes wait-free with probability 1: roughly, it is enough for the schedule to have some non-zero probability of picking each thread i in each step. They also gave evidence that such schedules are common in *high memory contention scenarios*, and characterized the performance of a general family of lock-free algorithms, called *Single-CAS Universal (SCU)*,¹ under an idealized *uniform stochastic scheduler*, which picks the next processor to schedule in each step with uniform probability.

More precisely, any algorithm in *SCU* follows a classic *scan-and-validate* pattern: a *scan* region designed to assess the state of the concurrent data structure and to locally compute an updated state, ending with a *validate* step consisting of a read-modify-write operation which atomically checks that the state of the data structure is still consistent with the scan, and, if so, commits the updated state. Crucially, this last validation step may fail if the data structure state changes between the scan and the validation step,

¹The SCU algorithmic pattern is *lock-free universal* [1, 13] in the sense that it can be used to provide a lock-free concurrent implementation for every object that can be specified sequentially.

meaning that some other operation succeeded in the meantime. If validation fails, the current operation restarts. This pattern is common in lock-free algorithms, e.g., [18, 24], and in databases with optimistic concurrency control [21].

The Stochastic Game. The analysis of algorithms in *SCU* under stochastic schedulers reduces to the following process [1]: we are given constant operation length $k \geq 2$, n bins (each associated with a thread), and probability distribution $\mathbf{p} = (p_1, p_2, \dots, p_n)$. In each step, we allocate one new ball (scheduler step) into one of the bins, according to the distribution \mathbf{p} , where bin i gets the ball with probability p_i . Bins continue to acquire balls, until one of them first reaches a threshold of $k + 1$ balls. At this time, the corresponding bin registers a *win* (its thread completes its operation), and gets reset to containing a single ball. Moreover, bins which contain k balls at this time get *reset* to having 0 balls, being penalized for *almost* winning. All other bins maintain their ball count. The rationale for this modeling choice is as follows. A newly completed operation changes the shared state; therefore, any thread that was *about to win*, i.e., within a step of performing its validation will have to first *fail* its validation, after which it must restart its current operation.

Given this process, we are interested in two questions: how often does *some bin* register a win (*system latency*), and how often does a *specific bin* i register a win (*individual latency*)?

The Question. Reference [1] gave bounds for these parameters under a *uniform scheduler distribution* $\mathbf{p} = (1/n, \dots, 1/n)$. However, their technique hinges on the symmetry of the distribution, and does not apply to non-uniform scheduling, leaving open the question of performance under non-uniform contention.

Bounding latencies for general \mathbf{p} is important for several reasons. First, modern processors exhibit consistently non-uniform scheduling, due to non-uniform memory access times. (See references [8, 17] for examples.) Second, key questions remain unsolved: what is the relative *performance advantage* of a thread that gets scheduled with higher probability? What is the *worst-case* scheduling distribution for lock-free algorithms? What is the *optimal* system latency achievable, while guaranteeing some minimal progress guarantees for individual threads?

Contribution. In this paper, we give asymptotically tight bounds on the performance of lock-free algorithms following the universal lock-free *SCU* pattern, in the common case where operations have constant length, with respect to any *arbitrary* sequence of scheduling distributions $\mathbf{p}(n)$, where n is the number of processors.²

THEOREM 1. *Given an arbitrary scheduling distribution $\mathbf{p} = (p_1, p_2, \dots, p_n)$, and constant operation length k , algorithms in *SCU* have system latency $\Theta(1/\|\mathbf{p}\|_2)$. Each process i has individual latency $\Theta(\|\mathbf{p}\|_2/p_i^2)$.*

Given the relatively complex structure of the algorithms, we find it surprising that such a simple characterization exists for general \mathbf{p} , and that this characterization does not depend on any higher norm of \mathbf{p} . By contrast, we also show that simple modifications of the pattern, where all bins

get reset to one ball upon reset, induce system latency $\Theta(1/\|\mathbf{p}\|_k)$ and individual latency $\Theta(p_i^k/\|\mathbf{p}\|_k)$. (These modifications correspond to obstruction-free algorithms [4, 14].)

Analysis Overview. The technical argument behind these bounds is quite non-trivial. We express the balls-into-bins game corresponding to an algorithm as a Markov chain, where each state is a possible configuration of occupancies of the bins, and each transition corresponds to a new ball being placed. An interesting feature of this chain is that a win by bin i also affects the state of bins containing k balls before i 's win, which induces non-trivial correlations between occupancies of the bins. It would be enough to characterize the stationary distribution of this chain: the rate at which bin i wins is the stationary probability of bin i transitioning from k balls to 1 ball, while the win rate of the system is the sum of individual rates. These win rates are the reciprocals of individual and system latencies, respectively.

Computing the stationary distribution of such a process for general \mathbf{p} , n , and k is infeasible, so we adopt a more elaborate strategy. We begin by bounding the system latency. Our main gadget for this part of the argument is to analyze a simplified chain, where all bin loads are considered modulo k . This choice is somewhat unintuitive, since this simplification obscures exactly the probability that a bin has k balls, which directly determines the bin winning rates. However, the stationary distribution of this simpler chain is easier to compute, and we show that it implies a clean $\sqrt{k}/\|\mathbf{p}\|_2$ lower bound on the expected system latency.

The system latency upper bound is more involved. We split the execution into phases, where a phase is the interval between two wins. (The expected length of this interval is the system latency.) We consider the state of the chain at a *random* time t , and wish to bound the *residual* (remaining) phase length from such a time. We partition bins according to the number of balls they contain at t , where a bin is at level $1 \leq \ell \leq k + 1$ if it still needs ℓ balls to reach $k + 1$ and win the phase. Ideally, we would like to argue that there are many bins at low levels, and that these bins will dominate the phase competition, winning within a small number of steps.

However, since the distribution \mathbf{p} may be unbalanced, the probability weight contained by each level ℓ at time t might not be well-concentrated. We overcome this technical obstacle by a careful argument which merges the Hoeffding bounds for each level ℓ , showing that the expected residual phase length has to be $O(1/\|\mathbf{p}\|_2)$, irrespective of the structure of \mathbf{p} . We prove high probability bounds similarly.

We then focus on bounding individual latencies, and in particular on upper and lower bounds on the probability w_i that a specific bin (thread) i wins a *random* phase. While the $\Omega(p_i^2/\|\mathbf{p}\|_2^2)$ lower bound on w_i is relatively simple, following by the previous analysis, the upper bound on w_i is non-standard.

To obtain the bound, we characterize the occupancies of *all* bins at the beginning of a random phase, to ensure that bin i gets enough competition. These occupancies are complex, and not well-concentrated, as we do not restrict \mathbf{p} and k . We circumvent this problem by first considering the occupancy distribution at a random time step. Then we build a connection between the probability that the bin occupancies are unbalanced at a *random time* and the probability that the bin occupancies are unbalanced at the *beginning of*

²We express complexity in terms of the number of threads n , as is standard; throughout, we write \mathbf{p} instead of $\mathbf{p}(n)$ for brevity.

a *random phase*. The proof of this relation employs an interesting adversarial counting argument. We roughly show that bin i gets a lot of competition on average, which implies a matching upper bound of $O(p_i^2/\|\mathbf{p}\|_2^2)$ on i 's winning probability, implying individual latency bounds.

Applications. Our result relates the scheduling distribution \mathbf{p} with the average performance of a lock-free algorithm in *SCU*, and provides simple answers to the questions on the relative latency advantage of processors (*ratio of scheduling probabilities squared*), and the worst-case scheduling distribution (*uniform*).

It has two additional practical applications: given the simple relation between \mathbf{p} and the performance metrics, determining the scheduling probabilities which minimize system latency, i.e., the optimal backoff scheme, under a set of individual latency constraints, becomes a well-defined optimization problem.

Second, a useful by-product of our technical framework is that we can understand the average-case behavior of various other concurrency patterns under arbitrary scheduling distributions. A particularly interesting pattern is when, on a win by thread i , *all* threads get reset to 1, irrespective of their current step count, which corresponds to the general family of *obstruction-free* concurrent algorithms, e.g. [4, 14], and to database transactions with optimistic concurrency control, e.g., [21, Chapter 7]. Our framework shows that the system latency of such algorithms is $\Theta(1/\|\mathbf{p}\|_k)$. This implies a separation in terms of average complexity between this algorithmic class and lock-free algorithms, for virtually any scheduling distribution \mathbf{p} .

2. RELATED WORK

Aspnes [3] was the first to consider distributed algorithms under randomized schedulers, solving consensus efficiently under a scheduler model different from the one we consider. Alistarh, Censor-Hillel, and Shavit [1] introduced the *stochastic* scheduler model, and isolated the *SCU* algorithmic pattern.

Their work implies bounds on the system and individual latencies for *SCU* given $\mathbf{p} = (1/n, 1/n, \dots, 1/n)$. Specifically, they showed that system latency is $O(\sqrt{n})$ steps, while individual latency is $O(n\sqrt{n})$ steps. In brief, their analysis works by expressing the algorithm as a Markov chain, and noticing that, since both the operation code and the scheduling distribution are *symmetric*, the general Markov chain can be collapsed onto a simpler chain, called the *global* chain. This mapping is a proper *lifting* [7, 12] of the general chain, and characterizing the stationary distribution of the global chain is enough to bound the latency of the algorithm. One main technical step is bounding the stationary probabilities of the global chain; this works by carefully characterizing the chain for $k = 2$, and then generalizing to general constants k by symmetry.

We analyze algorithms in *SCU* under *arbitrary distributions* \mathbf{p} , which requires us to take a different approach. We cannot employ Markov chain lifting, since states that are symmetric in terms of bin occupancies may now have completely different probabilities in the stationary distribution of the chain. Similarly, we can no longer reduce the system latency analysis to the case $k = 2$, since the generalization to arbitrary k relies on symmetry of the distribution. Finally, in the symmetric case, system latency immediately determines the individual latencies, as two bins have exactly

the same probability of winning an arbitrary phase. This is clearly no longer the case for non-uniform \mathbf{p} , and, in fact, determining the individual winning probabilities given the system latency is one of the main technical components of our argument. For uniform \mathbf{p} , we obtain the same asymptotic bounds as [1], although their lifting argument yields tighter constants.

Balls-into-bins processes [5] are popular abstractions for resource allocation problems, where balls (tasks, requests) are randomly allocated to bins (resources, processors). Such processes are relatively well understood in the uniform setting [5, 19], but recently there has been a growing interest in non-uniform settings, where either balls have different weights [23] or bins have correlated probabilities to be picked [11].

3. PRELIMINARIES

3.1 Modeling

The Shared Memory Model. In this model, n processes (threads) communicate through registers, on which they perform read, write, or compare-and-swap (CAS) atomic operations. A CAS operation takes three arguments (R , *expectedVal*, *newVal*), where R is the register on which it is applied, *expectedVal* is the expected value of the register, and *newVal* is the new value to be written. The operation compares *expVal* with the current value v of R , and atomically updates R to *newVal* if the expected value matches the current value, i.e. *expVal* = v . In this case, we say that the CAS *succeeds*. Otherwise, the value of R is not updated, and the CAS *fails*.

Each algorithm implements a shared object, which is an abstraction providing a set of methods, each given by its sequential specification. The algorithm is composed of shared-memory steps and local computation. The order in which these steps are performed is controlled by a *scheduler*.

Stochastic Schedulers. In general, a scheduler for n processes can be defined by a triple (Π_t, A_t, θ) . For each time step $t \geq 1$, Π_t is a probability distribution for scheduling the n processes at time t , and A_t is the subset of *possibly active* processes at time step t . At time step $t \geq 1$, the distribution Π_t gives, for every $i \in \{1, \dots, n\}$ a probability p_i^t , with which process i is scheduled. The distribution Π_t may depend on arbitrary outside factors, such as the current state of the algorithm being scheduled. For every $t \geq 1$, the parameters ensure the following: (i) $\sum_{i=1}^n p_i^t = 1$; (ii) For every process $i \in A_t$, $p_i^t \geq \theta$; (iii) For every process $i \notin A_t$, $p_i^t = 0$; (iv) $A_{t+1} \subseteq A_t$. A scheduler (Π_t, A_t, θ) is *stochastic* if $\theta > 0$. We consider a static *non-uniform* scheduler, which picks the next process to schedule using a fixed distribution $\mathbf{p} = (p_1, p_2, \dots, p_n)$ at every time t .

Executions and Schedules. An execution is a sequence of operations performed by the processes. Since process steps are atomic, we can assume discrete time, where at every time unit a single process is scheduled. In a time unit, a process performs any number of local computations or coin flips, after which it issues a *step*, which consists of a single shared memory operation. Whenever a process is scheduled to take a new step by the scheduler, it performs its local computation and then executes its next step, as dictated by the algorithm. Thus, the *schedule* is a possibly infinite sequence of process identifiers. If process i is in position t in the sequence, then it is active at time t .

```

1 Shared: registers  $R_1, R_2, \dots, R_k$ 
2 procedure operation()
3   while true do
4      $v_1 \leftarrow R_1.\text{read}(); v_2 \leftarrow R_2.\text{read}(); \dots;$ 
5      $v_{k-1} \leftarrow R_{k-1}.\text{read}()$ 
6      $v_k \leftarrow R_k.\text{read}()$  /* Scan region */
7      $v' \leftarrow$  new proposed state based on
8      $v_1, v_2, \dots, v_k$ 
9      $\text{flag} \leftarrow \text{CAS}(R_k, v_k, v')$  /* Validation
    step */
10    if  $\text{flag} = \text{true}$  then
11      output success

```

Algorithm 1: The structure of the lock-free algorithms in *SCU*.

Progress Guarantees. An implementation is *lock-free* (*non-blocking*) if, for each time t in the execution, there exists a finite bound T_t such that *some* operation returns within the next T_t steps. An implementation is *wait-free* if, for every operation, there exists a finite bound T on the number of steps that the operation can take before returning.

3.2 Algorithmic Pattern and Complexity Measures

We now describe the *Single-CAS Universal* algorithmic pattern which is the focus of our analysis. This pattern can be used to provide a lock-free concurrent implementation of any sequential object, via a straightforward extension of Herlihy’s universal construction [13]. This pattern is also the basis for several efficient implementations of shared objects, such as counters, lists, stacks, queues, or skip-lists, e.g. [10, 15, 24].

An operation using the *SCU* pattern consists of a *scan-and-validate* loop consisting of k steps reading shared registers R_1, R_2, \dots, R_k , ending with a *validation* step which performs a *compare-and-swap* operation on the last read location R_k . (Please see Algorithm 1.) If this compare-and-swap operation succeeds, then the operation returns successfully. Otherwise, if the operation fails, and the process re-issues the operation.

Complexity Metrics. We address the following question: what is the performance of an arbitrary algorithm A in *SCU*, under an arbitrary non-uniform stochastic scheduler given by the distribution $\mathbf{p} = (p_1, p_2, \dots, p_n)$?

We consider two natural performance measures. The first is *system latency*, i.e., the expected number of time steps between two completed operations by the system. The second is *individual latency*, i.e., the expected number of time steps between two committed operations by a *specific* process i . Individual latency is the correspondent of the *step complexity* metric, which measures the number of individual steps taken by an operation in order to complete.

3.3 Markov Chain Formulation and Preliminary Results

In the following, we fix an algorithm A in *SCU*, whose structure is fixed, and follows the pattern in Algorithm 1. We assume that each process executes an infinite number of operations, and that each process executes its next operation as soon as it completes the current one.

Modeling the Algorithm. First, notice that we can reduce A to the following balls-into-bins game. Consider a system of n bins, representing the processes, each starting

with one ball. This state corresponds to the process being about to read the value of R_1 . In each scheduling round, the scheduler allocates a new ball (i.e., a step) into one of the n bins: bin i gets the ball with probability p_i . Bins continue to receive balls until reaching k balls, which corresponds to being about to perform the CAS operation in line 7 in Algorithm 1.

Following the algorithm, whenever the scheduler places a ball into a bin that already contains k balls, a new operation gets completed and we therefore record a *win* by the system and by the corresponding process. Crucially, on a win by process i , the bin corresponding to the process is reset to having one ball, whereas all the other bins which had exactly k balls before i ’s win will be reset to 0 balls.

Markov Chain Formulation. It is easy to see that the system can be modeled as a discrete-time Markov chain. At the arrival of the t -th ball, a bin $I(t)$ is selected at random by drawing a sample from the distribution $\mathbf{p} = (p_1, p_2, \dots, p_n)$. The state of the system is defined by the bin occupancies $X(t) = (X_1(t), X_2(t), \dots, X_n(t))$ before the arrival of the t -th ball. If the selected bin has less than k balls, i.e., $X_{I(t)}(t) \in \{0, \dots, k-1\}$, then we simply increment $X_{I(t)}$ by 1 and continue. Otherwise, if $X_{I(t)}(t) = k$, we reset $I(t)$ to one ball, i.e., $X_{I(t)}(t+1) = 1$. Further, we reset all other bins $j \neq I(t)$ with $X_j(t) = k$ so that $X_j(t+1) = 0$.³ The process of bin occupancies $\{X(t)\}_{t \geq 0}$ is a discrete-time Markov chain with the finite state space $\{0, 1, \dots, k\}^n$.

Stationary Distribution and Ergodicity. Under the assumption $p_i > 0$ for each bin i , the transition matrix of the Markov chain $\{X(t)\}_{t \geq 0}$ is irreducible, and thus, has a unique stationary distribution π (see, e.g., [6]).

The chain $\{X(t)\}_{t \geq 0}$ is not aperiodic. To resolve this issue, we simply consider a perturbed version that is parameterized with $\delta \in (0, 1)$ such that at each time step a new ball arrives with probability $1 - \delta$ and is assigned to one of the bins according to the given distribution \mathbf{p} , and otherwise no ball arrives in the given time step and the occupancies of the bins remain unchanged. Such a perturbation results in a “lazy” Markov chain $\{X^\delta(t)\}_{t \geq 0}$ that has the same stationary distribution, but unlike the original chain, it is aperiodic, and hence, ergodic. The increase of the system and individual expected latencies is only for a factor $1/(1 - \delta)$, which can be made arbitrarily close to 1 by taking δ small enough. In what follows, we will tacitly assume that the given vector \mathbf{p} is already chosen so that the chain $\{X(t)\}_{t \geq 0}$ is ergodic, i.e., irreducible and aperiodic.

Win Points, Phases, and Latencies. For the Markov chain $\{X(t)\}_{t \geq 0}$, we say that a time-step t is a *win point* if at time-step t bin i is selected such that $X_i(t-1) = k$. The win points are technically reset points of the *point process* $T_0 \leq 0 < T_1 < T_2 < \dots$ where T_r is the time-step of the r -th win point.

A *phase* corresponds to the interval of time-steps starting at a win point and ending in the time-step just before the next win point. The phases are enumerated such that the 0-th phase corresponds to the interval of time-steps $[T_0, T_1)$, first phase corresponds to the interval of time-steps $[T_1, T_2)$ and so on. The length of the r -th phase is denoted by S_r . Since we are interested in the long-run average performance,

³The winning bin is reset to one ball since its next step is the first step in the next operation, whereas all other bins with $X_j(t) = k$ get reset to 0 since they must fail their validation (CAS operation) before being able to restart their operation.

we shall consider expected values with respect to the stationary distribution of the bin occupancies. We refer to a *random time* t to be an arbitrary time t under the assumption that the initial bin occupancies at time 0 are according to the stationary distribution. Under this assumption, the distribution of the bin occupancies at any given time $t \geq 0$ is the stationary distribution.

We shall also consider the state of the bin occupancies embedded at the win points. We refer to a *random phase* r to be an arbitrary phase r under assumption that the initial bin occupancies at time 0 are according to the stationary distribution of the bin occupancies at a win point. We are interested in characterizing system latency τ , defined as the expected number of time-steps between two consecutive win points, i.e., $\tau = \mathbf{E}[S_r]$. By the Palm inversion formula (see Lemma 1), the rate of win points λ satisfies $\lambda = 1/\tau$. The individual latency τ_i for bin i is the expected number of time steps between two consecutive win points in which bin i wins. Again, by the Palm inversion formula, the rate of wins of bin i , λ_i , satisfies $\lambda_i = 1/\tau_i$. The rates of system win points and individual win points satisfy $\lambda = \sum_{i=1}^n \lambda_i$. The winning probability of bin i in a random phase r is denoted with w_i . By the Palm inversion formula, $\lambda_i = \lambda w_i$, for every bin i . The rate of win points for a bin i satisfies $\lambda_i = p_i \cdot \Pr[X_i(t) = k]$, where t is a random time step.

Helper Results. We now state a couple of results which are used throughout the analysis. Their proofs are deferred to the full version of the paper [2].

The first is the Palm inversion formula. This is a well-known relation in the context of renewal and regenerative processes, which relates expectations of a stochastic process with respect to the stationary distribution of the state at an arbitrary time and the stationary distribution of the state at a special point in time. Please see e.g. [6] for further background and for a proof.

LEMMA 1 (PALM INVERSION FORMULA). *Let $X(t)$ be a regenerative process (e.g. see [6]) on the state space E with regeneration points $\dots < T_{-1} < T_0 = 0 < T_1 < \dots$ and the let $S_r = T_{r+1} - T_r$ for $r \in \mathbf{Z}$. Suppose that $f : E \rightarrow \mathbf{R}$ is such that*

$$\mathbf{E} \left[\sum_{t=0}^{S_0-1} |f(X(t))| \right] < \infty.$$

Suppose π is the stationary distribution of the process $X(t)$. Then, we have

$$\sum_{\mathbf{x} \in E} f(\mathbf{x}) \pi(\mathbf{x}) = \frac{\mathbf{E} \left[\sum_{t=0}^{S_0-1} f(X(t)) \right]}{\mathbf{E}[S_0]}.$$

If the distribution of S_0 is proper and non-lattice, then for any initial state $\mathbf{x} \in E$,

$$\lim_{t \rightarrow \infty} \mathbf{E}[f(X(t)) \mid X(0) = \mathbf{x}] = \sum_{\mathbf{x} \in E} f(\mathbf{x}) \pi(\mathbf{x}).$$

The second auxiliary result characterizes the stationary distribution of a simplified stochastic process.

LEMMA 2. *Let $Y(t) = (X_1(t) \bmod k, X_2(t) \bmod k, \dots, X_n(t) \bmod k)$. Then, $Y(t)$ is an ergodic, discrete-time Markov chain on the state space $\{0, 1, \dots, k-1\}^n$ with uniform sta-*

tionary distribution π , i.e.

$$\lim_{t \rightarrow \infty} \Pr[Y(t) = \mathbf{x} \mid Y(0) = \mathbf{y}] = \pi(\mathbf{x}) = 1/k^n,$$

for any $\mathbf{x}, \mathbf{y} \in \{0, 1, \dots, k-1\}^n$.

The second result characterizes the system latency and probability of winning a random phase for a simpler process, where all bins get reset to 1 on a win. Due to space limitations, the analysis of this process is given in the full version of this paper [2]. This result may be of independent interest, as it is related to non-uniform birthday processes, e.g. [20].

THEOREM 2. *The length S_r of a random phase r and the probabilities w_i that bin i wins an arbitrary phase, in the process where all bins reset to one ball on a win, satisfy:*

$$\mathbf{E}[S_r] \in \left[\frac{(k!/2)^{1/k}}{2\|\mathbf{p}\|_k}, \frac{8ek^k}{\|\mathbf{p}\|_k} \right] \text{ and } w_i = \Theta\left(\frac{p_i^k}{\|\mathbf{p}\|_k^k}\right).$$

4. ANALYSIS OF SYSTEM LATENCY

Proof Strategy. We are interested in upper and lower bounds on system latency τ , assuming that the operation length k is a constant. Determining these quantities from the stationary distribution of the Markov chain described in the previous section would be sufficient. Unfortunately, we will not be able to determine such bounds directly, as the occupancies of individual bins are highly correlated. Instead, we take a more roundabout approach: we first consider the stationary distribution of the simplified random process described in Lemma 2, in which bin occupancies are taken modulo k . Analyzing such a process may appear useless at first, as it obscures the probability that a bin contains k balls, which is essential for determining λ . However, we show that a characterization of this simplified process is a clean $\sqrt{k}/\|\mathbf{p}\|_2$ lower bound on system latency τ for the complete stochastic process.

THEOREM 3. *The expected system latency satisfies*

$$\tau \geq \sqrt{k}/\|\mathbf{p}\|_2.$$

PROOF. Fix an arbitrary bin i . By the definition of the Markov chain $\{X(t)\}_{t \geq 0}$, we have that the probability that a bin i contains 0 balls at time $t+1$ is

$$\Pr[X_i(t+1) = 0] = (1 - p_i) \Pr[X_i(t) = 0] + \sum_{j \neq i} p_j \Pr[X_j(t) = k, X_i(t) = k],$$

since the bin's state either did not change, or some other bin $j \neq i$ won in the previous step, while i also contained k balls.

Hence, for the expected values with respect to the stationary distribution we have

$$p_i \Pr[X_i(t) = 0] = \sum_{j \neq i} p_j \Pr[X_j(t) = k, X_i(t) = k].$$

By Lemma 2, $\Pr[X_i(t) = 0] + \Pr[X_i(t) = k] = 1/k$. Therefore, we can rewrite the previous relation as

$$\sum_{j=1}^n p_j \Pr[X_i(t) = k, X_j(t) = k] = \frac{1}{k} p_i.$$

Using this identity, we obtain

$$\begin{aligned} \frac{1}{k} \sum_{i=1}^n p_i^2 &= \sum_{i=1}^n p_i \sum_{j=1}^n p_j \Pr[X_i(t) = k, X_j(t) = k] \\ &= \mathbf{E} \left[\left(\sum_{i=1}^n p_i \mathbf{1}_{X_i(t)=k} \right)^2 \right] \\ &\geq \left(\sum_{i=1}^n p_i \Pr[X_i(t) = k] \right)^2 = \left(\sum_{i=1}^n \lambda_i \right)^2 = \lambda^2, \end{aligned}$$

where the last inequality is by Jensen's inequality and we use the fact that for every bin i , $\lambda_i = p_i \Pr[X_i(t) = k]$. We have thus showed that $\lambda \leq \sqrt{\sum_{i=1}^n p_i^2 / \sqrt{k}} = \|\mathbf{p}\|_2 / \sqrt{k}$. The proof is completed by recalling that $\tau = 1/\lambda$. \square

The proof of the upper bound is more complex, and works as follows. We pick a random time t , and bound the *residual phase length* $S(t)$, i.e., the remaining number of steps until an operation completes. The second step is to bound the length of a phase depending on this quantity; in fact bounding the residual phase length gives a *stronger* measure of performance.

To bound the residual phase length, our starting tool is Lemma 2, which provides a handle on the occupancy distribution modulo k at time t . However, the main difficulty with bounding $S(t)$ is that the occupancy distribution at time t may be quite irregular depending on the “shape” of the vector \mathbf{p} . We will take into account these possibilities, merging the Hoeffding bounds at each occupancy level to show that the residual phase length is always $O(1/\|\mathbf{p}\|_2)$.

LEMMA 3. *For a random time t , the residual phase length $S(t)$ satisfies $\mathbf{E}[S(t)] = O(1/\|\mathbf{p}\|_2)$. Moreover, for constant $C > 0$, $\Pr[S(t) \leq C \cdot (\log n)^{1+\frac{\log k}{2}} / \|\mathbf{p}\|_2] \geq 1 - 1/n$.*

PROOF. We partition the bins into *levels* $1, 2, \dots, k+1$, where a bin is at level ℓ at time t if it needs to acquire ℓ balls in order to win, i.e., it contains $k+1-\ell$ balls at t . Since we are interested in upper bounding the phase length, we can assume that there are no bins at level 1 at time t , as those could only shorten the phase.

For each bin i and level $2 \leq \ell \leq k+1$, we define the indicator random variable $Z_{\ell,i}(t)$ to equal p_i^ℓ if bin i is at level ℓ at time t , and 0, otherwise. Further, define $Z_\ell(t) = \sum_{i=1}^n Z_{\ell,i}(t)$. By Lemma 2, we know that $\mathbf{E}[Z_\ell(t)] = \frac{1}{k} \|\mathbf{p}\|_\ell^\ell$ and that $Z_{\ell,1}(t), Z_{\ell,2}(t), \dots, Z_{\ell,n}(t)$ are independent events for every $2 \leq \ell \leq k+1$. Hence, for fixed ℓ , we apply Hoeffding's inequality to variables $(Z_{\ell,i}(t))_i$ to obtain the following bound

$$\begin{aligned} \Pr[Z_\ell(t) \leq \|\mathbf{p}\|_\ell^\ell / (2k)] &\leq \exp \left(-\frac{1}{2k^2} \frac{(\sum_{i=1}^n p_i^\ell)^2}{\sum_{i=1}^n p_i^{2\ell}} \right) = \\ &\exp \left(-\frac{1}{2k^2} \frac{\|\mathbf{p}\|_{2\ell}^{2\ell}}{\|\mathbf{p}\|_\ell^{2\ell}} \right). \quad (1) \end{aligned}$$

Recall that the phase ends as soon as, for some level ℓ , a bin that is at level ℓ at time t acquires ℓ balls. By Theorem 2, we have that, conditional on the values of $(Z_{\ell,i}(t))_{\ell,i}$ the expected residual phase length is

$$\mathbf{E}[S(t) \mid (Z_{\ell,i}(t))_{\ell,i}] = O \left(\min_{2 \leq \ell \leq k+1} \frac{1}{Z_\ell(t)^{1/\ell}} \right). \quad (2)$$

We now focus on bounding the left-hand side quantity in terms of $1/\|\mathbf{p}\|_2$.

For $2 \leq \ell \leq k$, let \mathcal{E}_ℓ be the event that $Z_\ell(t) \leq \|\mathbf{p}\|_\ell^\ell / (2k)$. Let j be the index of the first event \mathcal{E}_ℓ that holds when we consider the events $\mathcal{E}_2, \mathcal{E}_3, \dots, \mathcal{E}_k$ in increasing order of parameter ℓ if any exists, and let j be equal to $k+1$, otherwise. In other words, the norm bounds for all levels $< j$ have failed, and the bound holds at j . Combining the resulting bound on the norm with Equation (2), it follows that $\mathbf{E}[S(t) \mid \overline{\mathcal{E}_2} \wedge \overline{\mathcal{E}_3} \wedge \dots \wedge \overline{\mathcal{E}_{j-1}} \wedge \mathcal{E}_j] = O(1/\|\mathbf{p}\|_j)$.

Applying the law of total expectation, it follows that there exists a constant $C > 0$ such that

$$\mathbf{E}[S(t)] \leq C' \left(\frac{1}{\|\mathbf{p}\|_2} + \sum_{\ell=2}^k \Pr[\overline{\mathcal{E}_2} \wedge \overline{\mathcal{E}_3} \wedge \dots \wedge \overline{\mathcal{E}_\ell}] \frac{1}{\|\mathbf{p}\|_{\ell+1}} \right). \quad (3)$$

Bounding the probability term for fixed ℓ and applying the bound (1) we obtain

$$\begin{aligned} \Pr[\overline{\mathcal{E}_2} \wedge \overline{\mathcal{E}_3} \wedge \dots \wedge \overline{\mathcal{E}_\ell}] &\leq \min_{j=2}^\ell \left\{ \Pr[\overline{\mathcal{E}_j}] \right\} \\ &\leq \left(\prod_{j=2}^\ell \Pr[\overline{\mathcal{E}_j}] \right)^{1/(\ell-1)} \leq \prod_{j=2}^\ell \exp \left(-\frac{1}{2k^2(\ell-1)} \frac{\|\mathbf{p}\|_j^{2j}}{\|\mathbf{p}\|_{2j}^{2j}} \right). \end{aligned}$$

Bounding the last term using the fact that $\exp(-x) \leq 1/x$ for $x > 0$, and $\|\mathbf{p}\|_j / \|\mathbf{p}\|_{j+1} \geq 1$, we have

$$\begin{aligned} \prod_{j=2}^\ell \exp \left(-\frac{1}{2k^2(\ell-1)} \frac{\|\mathbf{p}\|_j^{2j}}{\|\mathbf{p}\|_{2j}^{2j}} \right) &\leq \prod_{j=2}^\ell 2(\ell-1)k^2 \frac{\|\mathbf{p}\|_{j+1}}{\|\mathbf{p}\|_j} \\ &\leq (2(\ell-1)k^2)^{\ell-1} \frac{\|\mathbf{p}\|_{\ell+1}}{\|\mathbf{p}\|_2}. \end{aligned}$$

Combining with Equation (3), it follows that there exists a constant $C' > 0$ such that

$$\begin{aligned} \mathbf{E}[S(t)] &\leq C' \left(\frac{1}{\|\mathbf{p}\|_2} + \sum_{\ell=2}^k \frac{\|\mathbf{p}\|_{\ell+1}}{\|\mathbf{p}\|_2} \cdot \frac{1}{\|\mathbf{p}\|_{\ell+1}} \right) \\ &\leq C' \cdot \frac{k}{\|\mathbf{p}\|_2} = O \left(\frac{1}{\|\mathbf{p}\|_2} \right). \end{aligned}$$

The proof of the high probability bound is similar, and can be found in the full version of the paper [2]. \square

Final Step. To obtain the upper bounds on the phase length, we relate the residual phase length $S(t)$ at a random time t to the expected phase length S_r of a random phase r .

LEMMA 4. *Let S_r be the length of a random phase r , and let $S(t)$ be the residual length of the phase at a random time t . Then the following relations hold: $\mathbf{E}[S_r] \leq 2\mathbf{E}[S(t)]$ and $\Pr[S_r > 2\theta] \leq 2\Pr[S(t) > \theta]$, for $\theta \geq 0$.*

PROOF. The first claim is showed as follows. By the Palm inversion formula, we have

$$\begin{aligned} \mathbf{E}[S(t)] &= \frac{\mathbf{E}[\sum_{s=0}^{S_r-1} (S_r - s)]}{\mathbf{E}[S_r]} = \frac{\mathbf{E}[S_r(S_r + 1)]}{2\mathbf{E}[S_r]} \\ &\geq \frac{\mathbf{E}[S_r^2]}{2\mathbf{E}[S_r]} \geq \frac{1}{2} \mathbf{E}[S_r] \end{aligned}$$

where the last inequality is by Jensen's inequality.

The second claim is showed as follows. Again, by the Palm inversion formula we have

$$\begin{aligned}\Pr[S(t) > \theta] &= \frac{\mathbf{E}[\sum_{s=0}^{S_r-1} \mathbf{1}_{S_r-s>\theta}]}{\mathbf{E}[S_r]} = \\ \frac{\mathbf{E}[\max\{S_r - \theta, 0\}]}{\mathbf{E}[S_r]} &\geq \frac{\mathbf{E}[\max\{S_r - \theta, 0\} \mathbf{1}_{S_r>2\theta}]}{\mathbf{E}[S_r]} = \\ \frac{\mathbf{E}[S_r \mathbf{1}_{S_r>2\theta}]}{2 \mathbf{E}[S_r]} &\geq \frac{\mathbf{E}[S_r] \mathbf{E}[\mathbf{1}_{S_r>2\theta}]}{2 \mathbf{E}[S_r]} = \frac{1}{2} \Pr[S_r > 2\theta]\end{aligned}$$

where the last inequality follows by Harris' inequality, as S_r is a real-valued random variable and

$$\mathbf{E}[S_r \mathbf{1}_{S_r>2\theta}] = \mathbf{E}[f(S_r)g(S_r)]$$

for two non-decreasing functions f and g defined by $f(x) = x$ and $g(x) = \mathbf{1}_{x>2\theta}$ for $x \in \mathbf{R}$. \square

The proof of the phase upper bound therefore follows by combining Lemma 3 and Lemma 4.

THEOREM 4. *The expected system latency satisfies $\tau = O(1/\|\mathbf{p}\|_2)$.*

5. ANALYSIS OF INDIVIDUAL LATENCIES

5.1 Upper Bound for Winning Probabilities

We prove the following bound on the probability that bin i wins a phase, which will imply an individual latency bound of $O(\|\mathbf{p}\|_2/p_i^2)$ for bin i .

THEOREM 5. *For every bin i , the winning probability satisfies $w_i = O(p_i^2/\|\mathbf{p}\|_2^2)$.*

Proof Outline. The proof of this result is quite involved, so we first provide an outline for the case $k = 2$. Fix an arbitrary bin i . Let W_r be the event that in a random phase r bin i wins, so $w_i = \Pr[W_r]$. A time step t is defined to be bad if $\sum_{i=1}^n p_i \mathbf{1}_{Y_i(t)=1} \leq \|\mathbf{p}\|_2^2/16$. Similarly, a phase r is defined to be bad if its first time step is bad. Intuitively, if a phase is not bad, then there is “enough competition” which limits the winning probability of i . Let B_r denote the event that phase r is bad. By a simple conditioning, we have $\Pr[W_r] = \Pr[\bar{B}_r] \cdot \Pr[W_r | \bar{B}_r] + \Pr[B_r] \cdot \Pr[W_r | B_r]$. We shall establish the following three claims:

$$\Pr[B_r] \leq \exp\left(-C \frac{\|\mathbf{p}\|_2^2}{\|\mathbf{p}\|_3^2}\right), \text{ for a constant } C > 0, \quad (4)$$

$$\begin{aligned}\Pr[W_r | \bar{B}_r] &= O\left(\frac{p_i^2}{\|\mathbf{p}\|_2^2}\right) \text{ and} \\ \Pr[W_r | B_r] &= O\left(\frac{p_i^2}{\|\mathbf{p}\|_3^2}\right).\end{aligned} \quad (5)$$

The asserted upper bound follows then by combining these bounds and the fact that $\exp(-x) \leq 1/x$ for $x > 0$,

$$\Pr[W_r] \leq \exp\left(-C \frac{\|\mathbf{p}\|_2^2}{\|\mathbf{p}\|_3^2}\right) \cdot \frac{p_i^2}{\|\mathbf{p}\|_2^2} + \frac{p_i^2}{\|\mathbf{p}\|_2^2} = O\left(\frac{p_i^2}{\|\mathbf{p}\|_2^2}\right).$$

The inequality in (4) is obtained by showing first that the probability that a random time t is bad is at most $\exp(-C'\|\mathbf{p}\|_2/\|\mathbf{p}\|_3)$ for a constant $C' > 0$, and then showing that the probability that a random phase r is bad is

upper bounded by a similar bound. The first upper bound in (5) is obtained by conservatively assuming that at the beginning of phase r , bin i has one ball and exploiting that the probability mass on the bins having one ball is large enough, and then applying Theorem 2. The second upper bound in (5) is obtained by assuming that bin i has one ball and all other bins have zero balls, and again applying Theorem 2.

For $k \geq 3$, we need a more careful analysis, as the (conditional) winning probabilities could lie between any of the values $p_i^2/\|\mathbf{p}\|_j^2$ for $2 \leq j \leq k+1$. Corresponding to these values, we define different gradients of “badness”, and derive bounds as in (4) and (5) for each gradient separately. Then all these bounds are charged against each other and the proof is completed in a similar way as in Lemma 3. We give the full proof of Theorem 5 in the full version of our paper [2], and focus here on establishing the key step, Eq. (4), and its version for $k \geq 3$.

Probability of a Bad Phase. Our goal is to show that the probability that a random phase is bad is small. In order to capture the general case $k \geq 3$, we first have to formalize the notion of ℓ -badness of a time step and introduce some further pieces of notation. Similar to the proof of Lemma 3, we partition bins into levels $1, 2, \dots, k+1$, where bins at level ℓ need to acquire ℓ balls in order to win. For each bin i and level $2 \leq \ell \leq k+1$, we define the indicator random variable $Z_{\ell,i}(t)$ to equal p_i^ℓ if bin i is at level ℓ at time t , and 0, otherwise. Further, define $Z_\ell(t) = \sum_{i=1}^n Z_{\ell,i}(t)$.

DEFINITION 1. *For every level $2 \leq \ell \leq k$, a time step t is (i) ℓ -bad if $Z_\ell(t) < \|\mathbf{p}\|_\ell^\ell/(8k)$, (ii) ℓ -good if $Z_\ell(t) \geq \|\mathbf{p}\|_\ell^\ell/(8k)$, (iii) ℓ -perfect if all time steps in $[t, t+1/(32k\|\mathbf{p}\|_2)]$ are ℓ -good.*

As before, a phase is ℓ -bad, ℓ -good or ℓ -perfect if the first time step of that phase is ℓ -bad, ℓ -good or ℓ -perfect. The idea behind the definition of ℓ -perfect is that it enforces a particular structure on the time steps; every block of ℓ -perfect time steps is followed up with exactly $1/(32k\|\mathbf{p}\|_2)$ rounds which are ℓ -good (but not ℓ -perfect), after which there is at least one ℓ -bad round. The next lemma is the key ingredient of the proof.

LEMMA 5 (KEY LEMMA). *For every $2 \leq \ell \leq k$, a random phase is ℓ -bad w. p. $\leq \exp(-\Omega((\frac{\|\mathbf{p}\|_\ell}{\|\mathbf{p}\|_{\ell+1}})^4))$.*

PROOF. Let $B^\ell(T)$ be the number of ℓ -bad phases and let $N(T)$ be the number of phases in the time interval $[0, T)$. By the ergodicity of the Markov chain of bin occupancies and Theorem 4,

$$\begin{aligned}\Pr[\text{phase } r \text{ is } \ell\text{-bad}] &= \lim_{T \rightarrow \infty} B^\ell(T)/N(T) \\ \text{and} \quad \lim_{T \rightarrow \infty} N(T)/T &= \Omega(\|\mathbf{p}\|_2),\end{aligned}$$

where the limits hold with probability 1.

Let $L^\ell(T)$ be the number of non-perfect times steps in the time interval $[0, T)$. From Lemma 7 below it follows that $\lim_{T \rightarrow \infty} B^\ell(T)/L^\ell(T) = O(\|\mathbf{p}\|_2)$ with probability 1. Combining these relations with the relation below which comes from Lemma 6 completes the proof:

$$\lim_{T \rightarrow \infty} \frac{L^\ell(T)}{T} \leq \exp\left(-\Omega\left(\left(\frac{\|\mathbf{p}\|_\ell}{\|\mathbf{p}\|_{\ell+1}}\right)^4\right)\right) \text{ w.p. } 1$$

\square

LEMMA 6. For every level $2 \leq \ell \leq k$, a random time t is ℓ -perfect w. p. at least $1 - \exp\left(-\Omega\left(\left(\frac{\|\mathbf{p}\|_\ell}{\|\mathbf{p}\|_{\ell+1}}\right)^4\right)\right)$.

The proof of Lemma 6 follows by first exploiting that for a random time t , the occupancy distribution modulo k is uniform, which allows us to apply Hoeffding's bound to lower bound $Z_\ell(t)$. Then we lower bound the set of bins at level ℓ at time step t which do not receive any ball until time $t+1/(32k\|\mathbf{p}\|_2)$. Combining these two parts yields the statement of the lemma, whose complete proof is deferred to the full version of this paper [2].

LEMMA 7. For every level $2 \leq \ell \leq k$ and random time t ,

$$\Pr[\text{time step } t \text{ is beginning of a } \ell\text{-bad phase}] \leq O(\|\mathbf{p}\|_2) \cdot \Pr[\text{time step } t \text{ is not } \ell\text{-perfect}].$$

The proof of this lemma relies on a careful comparison between the point process associated to ℓ -bad phases, and the point process associated to ℓ -(non-)perfect time steps. Intuitively, we will show that, on average, every bad phase can be charged $\Omega(1/\|\mathbf{p}\|_2)$ non-perfect time steps, which then implies Lemma 7.

Proof of Lemma 7. We will relate ℓ -non-perfect time steps and ℓ -bad phases (since $2 \leq \ell \leq k-1$ is fixed throughout this part, we will often simply say non-perfect and bad). To this end we will consider a specific process which in parallel counts the number of bad phases and lower bounds the number of non-perfect time steps. This process will use a (non-complete) partitioning of phases into set of phases, called epochs.

DEFINITION 2. An epoch is defined to be an interval of consecutive phases that starts at the beginning of a bad phase and ends at the end of the first occurrence of a phase that contains at least one perfect time step.

Starting from a fixed perfect time step (w.l.o.g. time 0), we can enumerate all epochs by E_1, E_2, \dots . The j -th phase of the e -th epoch is denoted by $E_{e,j}$, whose length is $|E_{e,j}|$. Finally, $|E_e| \geq 1$ denotes the size of epoch e , i.e., the number of phases it contains. An illustration of these definitions can be found in Figure 1.

Counting the Ratio Phase-by-Phase. Figure 1 suggests a link between the phase lengths and the number of non-perfect time steps. In particular, every phase of an epoch except for the last one contains only non-perfect time steps. To elaborate on this link, we next describe a process which follows the partitioning into epochs to update two counters. First, a counter B adds up all phases which are part of the e -th epoch. Simultaneously, a second counter L adds up all phase lengths $|E_{e,j}|$ except the last one as well as the $1/(32k\|\mathbf{p}\|_2)$ good but non-perfect time steps that prelude any new epoch. The precise description can be found in Process 2.

The next lemma shows that the counters B and L are indeed related to the number of bad phases and the number of non-perfect rounds. It basically follows from the definition of an epoch.

LEMMA 8. Consider Process 2 running until the f -th epoch is completed, and let $B = B(f)$ and $L = L(f)$ be the values of the counters B and L at this time. Then the following two statements hold. First, $B(f)$ counts exactly the total number of bad phases that occur until the last time step of the

```

1  e = 0, j = 0 /* counters for epoch and phase
   */
2  L = 0, B = 0 /* counters for non-perfect
   rounds and bad phases */
3  while true do
4      Jump to the next bad phase, e = e + 1, j = 0
5      repeat
6          j = j + 1
7          Allocate balls until phase j of epoch i
8          ends
9          if j = 1 then L = L + 1/(32k||p||_2),
10             B = B + 1
11             if j ≥ 2 then L = L + |Ee,j-1|,
12                 B = B + 1
13             until phase j contained a perfect time step

```

Process 2: The original counting process and its phase-based updates.

f -th epoch. Second, the expression $L(f) = \sum_{e=1}^f \left(\frac{1}{32k\|\mathbf{p}\|_2} + \sum_{j=1}^{|E_e|-1} |E_{e,j}|\right)$ lower bounds the number of non-perfect time steps which belong to a phase of the first f epochs.

Counting the Ratio Epoch-by-Epoch. The problem with analyzing Process 2 is that we do not have a good control on its actual length in terms of phases or time steps. We could think of it as facing an online adversary who is allowed to see the evolution of all phase lengths and then able to end the epoch after an arbitrary phase. However, we know that every phase will take $\Omega(1/\|\mathbf{p}\|_2)$ with constant prob. > 0 , regardless of the occupancy distribution at the beginning of that phase. This motivates the definition of a modified process, where the epoch length is the maximum number x so that more than half of its phase lengths are less than some threshold $1/(16k\|\mathbf{p}\|_2)$. If $x > 0$, then we conservatively assume that the epoch runs for x phases all of which have length 0. If $x = 0$, then we assume that the epoch consists of one phase of length $1/(128k\|\mathbf{p}\|_2)$.

For the modified process we generate for the e -th epoch with phase j a separate infinite sequence of bin samples denoted by $M_{e,j} \in \{1, \dots, n\}^{\mathbb{N}}$. Here, $M_{e,j}(t)$ is the bin where the t -th ball of phase j in epoch E_e is placed, where for any bin i , $\Pr[M_{e,j}(t) = i] = p_i$. Further, for every pair e, j , we use an indicator variable $Z_{e,j}$ which is one iff the bin samples $M_{e,j}$ ensure that no bin gets two balls within $1/(16k\|\mathbf{p}\|_2)$ time steps.

```

1  e = 0,  $\tilde{L} = 0$ ,  $\tilde{B} = 0$  /* counters for epoch,
   non-perfect rounds and bad phases */
2   $M_{e,j} \in \{1, \dots, n\}^{\mathbb{N}}$  /* random sequence for
   bin samples in phase j of the e-th
   epoch */
3  while true do
4      e = e + 1
5       $\tilde{S}_e = \max\{x \geq 1 : \sum_{j=1}^x Z_{e,j} < x/2\}$ 
6      /* defined by  $M_{e,j}$  for  $j = 1, 2, \dots$  */
7      if  $\tilde{S}_e = 0$  then  $\tilde{L} = \tilde{L} + 1/(128k||p||_2)$ ,
9           $\tilde{B} = \tilde{B} + 1$ 
10         if  $\tilde{S}_e \neq 0$  then  $\tilde{L} = \tilde{L} + 0$ ,  $\tilde{B} = \tilde{B} + \tilde{S}_e$ 

```

Process 3: The modified process and its epoch-based updates.

We proceed to establish a coupling which implies that L/B majorises \tilde{L}/\tilde{B} . A random variable X is stochastically

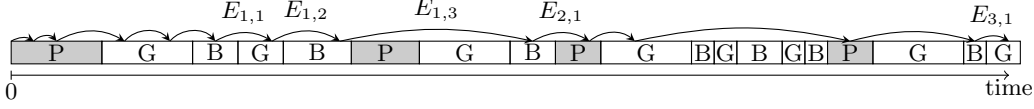


Figure 1: Illustration of phases viewed as a point process in time, which can be of three different types: P (perfect), G (good but not perfect) and B (bad). Arrows with a label $E_{e,j}$ form phase j in epoch e . Phases without a label do not belong to any epoch.

smaller (is majorized by) Y , in symbols $X \preceq Y$, if $\Pr[X \geq \theta] \leq \Pr[Y \geq \theta]$ for every θ .

LEMMA 9. Consider the original process and modified process both running until epoch f is completed, and let $B(f)$, $L(f)$, $\tilde{B}(f)$, $\tilde{L}(f)$ be the value of the four counters at this time. Then $L(f)/B(f) \succeq \tilde{L}(f)/\tilde{B}(f)$.

The proof of this lemma is based on a coupling where the original process uses the same bin samples $M_{e,j}$ as the modified process. Finally, we study the distribution of the counters \tilde{B} and \tilde{L} .

LEMMA 10. For the modified process (Process 3), the following statements holds.

- For any $\theta \geq 1$ and epoch e , $\Pr[\tilde{S}_e \geq \theta] \leq (1/4)2^{-\theta}$, thus, in particular, $\Pr[\tilde{S}_e < \infty] = 1$.
- Let $\tilde{B} = \tilde{B}(f)$ and $\tilde{L} = \tilde{L}(f)$ be the values of the counters \tilde{B} and \tilde{L} until epoch f is completed. Then, $\tilde{L}(f) \succeq \text{Bin}(f, 1/2)/(128k\|\mathbf{p}\|_2)$, and $\tilde{B}(f) \preceq \text{Geo}(1/2) \cdot f$.

PROOF COMPLETING THE PROOF OF LEMMA 7. First, we consider $\tilde{L}(f)/\tilde{B}(f)$ for a large value of epoch f . By Lemma 10, $\mathbf{E}[\tilde{L}(f)] \geq (f/2)/(128k\|\mathbf{p}\|_2)$ and $\mathbf{E}[\tilde{B}(f)] \leq 2f$. Hence by ergodicity, the following limits hold w.p. 1:

$$\lim_{f \rightarrow \infty} \tilde{L}(f) \geq f \cdot \frac{1}{192k\|\mathbf{p}\|_2} \text{ and } \lim_{f \rightarrow \infty} \tilde{B}(f) \leq 3f,$$

$$\text{implying } \lim_{f \rightarrow \infty} \frac{L(f)}{B(f)} \geq \lim_{f \rightarrow \infty} \frac{\tilde{L}(f)}{\tilde{B}(f)} \geq \frac{1}{576k\|\mathbf{p}\|_2},$$

where the first inequality follows from Lemma 9. By Lemma 8, we know that $L(f)$ is a lower bound on the number of non-perfect time steps until the end of epoch f , and $B(f)$ counts the number of bad phases until the end of epoch f . Hence considering the beginning steps of bad phases as a separate process, we have

$$\Pr[\text{time step } t \text{ is beginning of a bad phase}] \leq 576k \cdot \|\mathbf{p}\|_2 \cdot \Pr[\text{time step } t \text{ is not perfect}].$$

□

Lower Bound on Winning Probabilities. Finally, we prove a lower bound on the winning probability exploiting the fact that at a random time t , any bin i has $k-2$ balls with probability $1/k$.

THEOREM 6. For any bin i , the winning probability satisfies $w_i = \Omega(p_i^2/\|\mathbf{p}\|_2^2)$.

Due to space limitations, we only present an outline of the proof. We know that for a random time t the probability that bin i has $k-2$ balls is exactly $1/k$. We show that this condition continues to hold from time t until the end

of the current phase with a constant probability. This will follow by considering the additional events that (i) bin i does not receive a ball in the time interval $[t, t+s]$ where $s = \Theta(1/\|\mathbf{p}\|_2)$ and (ii) the event that the next phase starts in that interval. By reasoning about the point process, we can deduce that for a constant fraction of phases, bin i has $k-2$ balls, and the claim then follows by Theorem 2 which shows that bin i has a probability of $\Omega(p_i^2/\|\mathbf{p}\|_2^2)$ for winning such a phase.

6. APPLICATIONS AND FUTURE DIRECTIONS

Our characterization has two non-trivial applications. The first is given by the following practical question: given a set of lower bound constraints on the performance of individual threads (individual latency upper bounds), what is the scheduling distribution which minimizes system latency? Modifying the scheduling distribution is commonly done through back-offs. By Theorem 1, this now corresponds to the simple optimization problem of maximizing the 2-norm of the scheduling probabilities \mathbf{p} , subject to a set of lower bounds on the scheduling probabilities.

The second application comes by noticing that the simple process where all bins are reset to 1 on a win models *obstruction-free* concurrent algorithms, which guarantee progress only in the absence of contention, and may abort operations otherwise. Theorem 2 implies that the system latency of an obstruction-free algorithm is $\Theta(1/\|\mathbf{p}\|_k)$, where k is the operation length. This provides a complexity separation between obstruction-free and lock-free algorithms, under stochastic schedulers, for most values of the scheduling distribution \mathbf{p} .

Summing up, we have given general framework for competitive non-uniform balls-into-bins processes. While we focused on a particular application, lock-free algorithms, we believe that our techniques can be used to model a broader class of algorithms and processes related to scheduling contended applications. Hence, a natural but challenging question would be to find a general characterization, which parameterizes the winning probabilities and phase lengths in terms of the reset policies. Further, it would be interesting if a similar model applies to *transactions* (either in hardware or in software), and if similar performance bounds can be derived.

7. REFERENCES

- [1] Dan Alistarh, Keren Censor-Hillel, and Nir Shavit. Are lock-free concurrent algorithms practically wait-free? In *Proceedings of the 46th ACM Symposium on Theory of Computing*, STOC'14, pages 714–723, 2014. Full version containing measurements available at: <http://arxiv.org/abs/1311.3200>.
- [2] Dan Alistarh, Thomas Sauerwald, and Milan Vojnovic. Lock-free algorithms under stochastic schedulers. Technical Report MSR-TR-2015-41, May 2015.

- [3] James Aspnes. Fast deterministic consensus in a noisy environment. *J. Algorithms*, 45(1):16–39, 2002.
- [4] Hagit Attiya, Rachid Guerraoui, Danny Hendler, and Petr Kuznetsov. The complexity of obstruction-free implementations. *J. ACM*, 56(4):24:1–24:33, July 2009.
- [5] Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999.
- [6] Pierre Bremaud. *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues (Text in Applied Mathematics)*. Springer, 2nd edition, 2001.
- [7] Fang Chen, László Lovász, and Igor Pak. Lifting markov chains to speed up mixing. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*, STOC’99, pages 275–281.
- [8] Dave Dice, Virendra J. Marathe, and Nir Shavit. Brief announcement: Persistent unfairness arising from cache residency imbalance. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA’14, pages 82–83, 2014.
- [9] Faith Ellen, Panagiota Fatourou, Joanna Helga, and Eric Ruppert. The amortized complexity of non-blocking binary search trees. In *Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing*, PODC ’14, pages 332–340, New York, NY, USA, 2014. ACM.
- [10] Keir Fraser and Timothy L. Harris. Concurrent programming without locks. *ACM Trans. Comput. Syst.*, 25(2), 2007.
- [11] Brighten Godfrey. Balls and bins with structure: balanced allocations on hypergraphs. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, SODA’08, pages 511–517, 2008.
- [12] Thomas P. Hayes and Alistair Sinclair. Liftings of tree-structured markov chains. In *Proceedings of the 13th International Conference on Approximation, and 14th International Conference on Randomization, and Combinatorial Optimization: Algorithms and Techniques*, APPROX/RANDOM’10, pages 602–616, 2010.
- [13] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):123–149, January 1991.
- [14] Maurice Herlihy, Victor Luchangco, and Mark Moir. Obstruction-free synchronization: Double-ended queues as an example. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, ICDCS’03, pages 522–529, 2003.
- [15] Maurice Herlihy and Nir Shavit. *The art of multiprocessor programming*. Morgan Kaufmann, 2008.
- [16] Maurice Herlihy and Nir Shavit. On the nature of progress. In *Proceedings of the 15th International Conference on Principles of Distributed Systems*, OPODIS’11, pages 313–328, 2011.
- [17] Christoph Lameter. Numa (non-uniform memory access): An overview. *Queue*, 11(7):40:40–40:51, July 2013.
- [18] Maged M. Michael and Michael L. Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *Proceedings of the 15th ACM Symposium on Principles of Distributed Computing*, PODC’96, pages 267–275, 1996.
- [19] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, New York, NY, USA, 2005.
- [20] Thomas S. Nunnikhoven. A birthday problem solution for nonuniform birth frequencies. *The American Statistician*, 46(4):270–274, 1992.
- [21] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill, Inc., New York, NY, USA, 3rd edition, 2003.
- [22] Michael L. Scott. *Shared-Memory Synchronization*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2013.
- [23] Kunal Talwar and Udi Wieder. Balanced allocations: the weighted case. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, STOC’07, pages 256–265, 2007.
- [24] R. K. Treiber. Systems programming: Coping with parallelism. Technical Report RJ 5118, IBM Almaden Research Center, 1986.