Load Balancing with Memory

Michael Mitzenmacher* Balaji Prabhakar † Devavrat Shah ‡

Abstract

A standard load balancing model considers placing n balls into n bins by choosing d possible locations for each ball independently and uniformly at random and sequentially placing each in the least loaded of its chosen bins. It is well known that allowing just a small amount of choice ($d = 2$) greatly improves performance over random placement ($d = 1$). In this paper, we show that similar performance gains occur by introducing memory. We focus on the situation where each time a ball is placed, the least loaded of that ball's choices after placement is remembered and used as one of the possible choices for the next ball. For example, we show that when each ball gets just one random choice, but can also choose the best of the last ball's choices, the maximum number of balls in a bin is $\log\log n/2\log\tau + O(1)$ with high probability, where $\tau = (1+\sqrt{5})/2$ is the golden ratio. The asymptotic performance is therefore better with one random choice and one choice from memory than with two fresh random choices for each ball; the performance with memory asymptotically matches the asymmetric policy using two choices introduced by Vöcking. More generally, we find that a small amount of memory, like a small amount of choice, can dramatically improve the load balancing performance. We also investigate continuous time variations corresponding to queueing systems, where we find similar results.

1 Introduction

The idea of using small amounts of choice to improve load balancing schemes is now well understood, and is often expressed with the following simple example. Suppose that n balls are thrown into n bins, with each ball choosing a bin independently and uniformly at random. Then the *maximum load*, or the most balls in any bin, is approximately $\log n/\log\log n$ with high probability. Suppose instead that the balls are placed sequentially, and each ball is placed in the least loaded of $d \geq 2$ bins chosen independently and uniformly at random. Azar, Broder, Karlin, and Upfal showed that in this case, the maximum load is $\log\log n/\log d \pm \Theta(1)$ with high probability [1]. This result demonstrates the important power that choice can have in load balancing.

Following recent work in [11], in this paper we demonstrate that memory combined with choice can yield similar gains in performance. For example, one of our results concerns the following variation: n balls are thrown into n bins, with each ball choosing a bin independently and uniformly at random. When a ball is placed, the least loaded of that ball's choices after placement is remembered and used as one of the possible choices for the next ball. With just

*Harvard University, Department of Engineering and Applied Sciences. michaelm@eecs.harvard.edu.
†Stanford University, Departments of Electrical Engineering and Computer Science. balaji@Stanford.edu.
‡Stanford University, Department of Computer Science. devavrat@Stanford.edu.

one random choice and one choice from memory, the maximum load is $\log\log n/2\log\tau + O(1)$, where $\tau = (1+\sqrt{5})/2$ is the golden ratio. Hence a second choice from memory is asymptotically better than a second fresh random choice at each step. We obtain similar results for the case of general d; having d random choices and one choice from memory is slightly better than having $2d$ independent random choices, in terms of the asymptotic maximum load.

Besides being of theoretical interest, our work provides grounding for heuristics used in practice. For example, the idea of *sticky routing*, where a good route once found is re-utilized until it becomes congested, has been used and analyzed for telephone call networks [6]. More recently, randomized switch scheduling algorithms that use memory have been analyzed in [5, 13]. These algorithms considerably simplify the implementation while providing 100% throughput and nearly optimal backlogs. The use of memory in load balancing can be motivated by thinking of tasks as coming in bursts from a source. The source may remember and re-use good destinations. In the queueing setting, this model has been studied empirically by Dahlin [3]. Our analysis assumes that a new task always has some memory from the last task; this may be a good approximation if burst sizes are sufficiently large.

1.1 Background and Related Work

There is a substantial amount of related work, as related models have been the subject of a great deal of study. The survey by Mitzenmacher et al [8] provides details on most of the previous related work. We therefore recall only the relevant essentials.

The problem where n balls are sequentially placed into n bins, with each ball choosing from the best of d random bins, was introduced and analyzed by Azar et al [1]. Following [9], we call their policy the d-random policy. Among other things, they showed that the d-random policy yields a maximum load of $\frac{\log\log n}{\log d} \pm \Theta(1)$ with high probability.

Vöcking demonstrated how to use asymmetry to improve the result of Azar et al [14]. In Vöcking's framework, the bins are split into d groups of equal size. We may think of these groups as being laid out from left to right. Each ball chooses one bin from each group independently and uniformly at random. The ball is placed in the least loaded bin, with ties being broken in favor of the leftmost bin. The combination of the split and the tie-breaking policy yields smaller maximum loads. Indeed, the maximum load is $\frac{\log\log n}{d\log\phi_d} \pm \Theta(1)$, where ϕ_d is the growth exponent of the generalized dth order Fibonacci sequence. Following [9], we refer to Vöcking's approach as the d-left policy. Note that when $d = 2$, the maximum load for the 2-left policy is $\frac{\log\log n}{2\log\tau} \pm \Theta(1)$, matching our policy with one random choice and one choice from memory. The 2-left policy, however, requires more randomness resources, as well as an a priori agreement on the grouping of bins.

Mitzenmacher [7] develops an approach using the theory of large deviations for studying these load balancing problems by deriving differential equations for the limiting behavior as n grows large. This methodology is also useful for studying dynamic, queueing versions of the problem, where the balls correspond to tasks and the bins correspond to exponential server queues. Similar queueing systems were also analyzed independently by Vvedenskaya et al [15].

In this work, we make use of a non-trivial extension of large deviation theory that has not been used in the above load balancing work, as we describe below. The treatment we follow is from the text of Shwartz and Weiss [12]; the book by Ethier and Kurtz is also a useful reference [4].

1.1.1 The memory model

We consider the following problem: n balls are thrown into n bins. Under a (d, m)-memory policy, each time a ball is thrown d (a constant) bins are chosen independently and uniformly at random from the n bins. Also, m (a constant) bins are stored in memory from the past throw. (For the first ball, the memory is empty; this does not affect the analysis.) The ball is placed in the least loaded of the $d + m$ bins (with ties broken arbitrarily). After a ball is placed, the m least loaded of the $d + m$ bins are kept in memory. The case of no memory ($m = 0$) is the now classic problem analyzed in [1]. For ease of exposition we focus on the case $m = 1$ for the remainder of the paper, although our analysis is easily extended to general m, and more details for the general case will appear in the full paper. We note that one could also analyze variations of the asymmetric process suggested by Vöcking with memory; results for this system will also appear in the full paper.

In the dynamic setting, bins correspond to queues. Arrivals occur according to a rate $n\lambda$ ($\lambda < 1$) Poisson process at a bank of n independent rate μ_i exponential servers. We will assume that $\sum_i \mu_i = n$, so that the net service rate is larger than the net arrival rate. We refer to this general queueing setup as the *supermarket model*, following [7]. As in the balls and bins problem, for each arrival d queues are chosen independently and uniformly at random, and m queues are stored in memory. The arrival is placed in the least loaded of the $d+m$ queues, and the m least loaded of the $d + m$ queues are kept in memory. The case of no memory ($m = 0$) has been analyzed in [9] for the case when $\mu_i = 1$ for all i. Here again we focus on the case $m = 1$.

Shah and Prabhakar [11] analyzed both the discrete and continuous memory models. We briefly recall their results and motivate the results of this paper.

They consider the supermarket model when the service rates μ_i are not all equal (but $\sum_i \mu_i = n$), and show that when sampling is done *with replacement*, the backlog in the system can increase without bound under the d-random policy, *even when* $d = O(n)$. However, they show that the $(1, 1)$-memory policy keeps the queue lengths finite (even in expectation) for all $\lambda < 1$. Thus, whereas the d-random policy is unable to detect slower servers from faster ones and cannot keep the queues bounded, the use of memory in the $(1, 1)$-memory policy helps achieve stability. This result strongly demonstrates the benefit of using memory when the servers in the system run at different speeds.

For the balls and bins model, [11] uses large deviations theory and coupling arguments to show that the maximum load under the $(d, 1)$ policy ($d \geq 2$) is bounded above by $\frac{\ln \ln n}{\ln(2d-1)} + O(1)$ with high probability. This demonstrates that the effect of the samples stored in memory is "multiplicative" rather than just "additive." That is, each sample in memory counts as $O(d)$ new random samples, as opposed to just one.

1.1.2 A Preliminary Sketch of our Methods and Results

In this paper we analyze the discrete and continuous versions of the problem and obtain exact bounds. Our main results make use of a non-trivial extension of large deviation theory (following the treatment in the text of Shwartz and Weiss [12]). The use of memory complicates the analysis as follows. Consider the (1,1)-memory policy for the supermarket model. Under Poisson arrivals and exponential services, the load of the queues in the system and of the queue in memory form a Markov chain. The load of the system is generally represented by a vector with the fraction of queues with i customers for each i. The complexity of this Markov chain is that it has modes evolving at two different time scales: the load of the queue in memory evolves much faster than the load of the queues in the entire system, since it can change at each arrival while the system state is affected by $O(n)$ arrivals. It is therefore not possible to analyze the memory system using large deviations methods as in Mitzenmacher [7]. The rather technical way out is to decompose the overall system into two Markov chains: one for the memory and the other for the servers.

Since the Markov chain for the memory evolves much faster than the chain for the load, for large enough n its transitions appear at any time to be in equilibrium given the state of the queue loads. The state of the memory then in turn influences the loading and hence the state of the rest of the system. What is needed is a proper pasting of the two different chains.

We shall execute this procedure, tailoring the methods in [12] to our needs, in the coming sections. Specifically, after reviewing the appropriate results from large deviations theory, we first formally consider the discrete system and determine exact orders. We then rigorously analyze the continuous supermarket model. We also provide simulation comparisons of the $(1,1)$-memory policy with the 2-random and 2-left policies.

2 Large Deviations Results

In this section, we first review results from large deviations theory that we later use. Our treatment is necessarily a brief summary, based on the more extensive treatment be found in [12]. Readers uninterested in the fine technical details may skip this section on a first reading, and accept the more intuitive explanations in the appropriate sections.

We begin with a definition.

Definition 1. A *finite-level finite-dimensional jump Markov process* with D dimensions and L levels has the state space $\mathbb{R}^D \times \{0, 1, \ldots, L-1\}$. The state is represented by the $D+L$ tuple $(\bar{x}; m) = (x_0, \ldots, x_{D-1}; 0, \ldots, 1, \ldots, 0)$, where a 1 in position $D+m, 1 \le m \le L$ represents that the system is in level m. When in state $(\bar{x}; m)$ the system can make $\zeta(m)$ possible different jumps: it jumps to state $(\bar{x} + \bar{e}_i(m); \tilde{m}(m,i)) = (\bar{x} + \bar{e}_i(m); 0, \ldots, 1-\alpha, 0, \ldots, \alpha, \ldots, 0)$ with rate $\nu_i(\bar{x}; m)$, for $1 \le i \le \zeta(m)$, and $\alpha \in \{0,1\}$. Here $\bar{e}_i(m)$ is a unit vector in one of the D dimensions, and, depending on value of α, the level may change or remain as it is.

The idea behind this definition is that we have an underlying finite-dimensional jump Markov process (such as that used in [7]), along with an associated "level" Markov chain on the state space $\{0, \ldots, L\}$; the state of the second Markov chain may affect the transition rates of the first.

The generator A of this Markov process, which operates on real valued functions $f : \mathbb{R}^{D+L} \to \mathbb{R}$ is defined as:

$$Af(\bar{x}; m) \quad = \quad \sum_{i=1}^{\zeta(m)} \nu_i(\bar{x}; m)[f(\bar{x} + \bar{e}_i(m); \tilde{m}(m; i)) - f(\bar{x}; m)] \qquad (1)$$

Our interest is in the scaled version of this process with scaling parameter n. The scaling is done as follows: the rate of each transition is scaled up by n, while the jump magnitude is scaled down by n. The state of this scaled system will be represented by $(\bar{s}_n; m) =$

$(s_0, \ldots, s_{D-1}; 0, \ldots, \frac{1}{n}, \ldots, 0)$. The jump vector will be $(\frac{\bar{e}_i}{n}; 0, \ldots, -\frac{\alpha}{n}, \ldots, \frac{\alpha}{n}, \ldots, 0)$, and the corresponding rates are $n\nu_i(\bar{s}_n; m)$ for $1 \le i \le \zeta(m)$. The generator for the scaled Markov process is:

$$\begin{aligned} A_n f(\bar{s}_n; m) &= \sum_{i=1}^{\zeta(m)} n\nu_i(\bar{s}_n; m)\{f[\bar{s}_n + \frac{\bar{e}_i(m)}{n}; 0, .., \frac{1}{n} - \frac{\alpha(m,i)}{n}, 0, .., \frac{\alpha(m,i)}{n}, .., 0] \\ &\qquad - f(\bar{s}_n; m)\}. \end{aligned} \tag{2}$$

The following theorem (Theorem 8.15 from [12]) describes the evolution of the typical path of the scaled Markov process in the limit as n grows large. The idea behind the theorem is that because the finite-level Markov chain reaches equilibrium in some finite time, for large enough n the approximation that the finite-level Markov chain is in equilibrium is sufficient to obtain Chernoff-like bounds.

Theorem 1. *Under Conditions 1 and 2 below, for any given T and $\epsilon > 0$, there exist positive constants $C_1, C_2(\epsilon)$ and n_0 such that for all initial positions $\bar{s}^0 \in \mathbb{R}^D$, any initial level $m \in \{0, 1, \ldots, L-1\}$, and any $n \ge n_0$,*

$$\Pr_{\bar{s}^0, m}\left(\sup_{0\le t\le T} |\bar{s}_n(t) - \bar{s}_\infty(t)| > \epsilon\right) \le C_1 \exp(-nC_2(\epsilon)). \tag{3}$$

where, $\bar{s}_\infty(t)$ satisfies the following:

$$\begin{aligned} \frac{d}{dt}\bar{s}_\infty(t) &= \sum_{l=0}^{L} \Pr(m(t) = l) \sum_{i=1}^{\zeta(l)} \nu_i(\bar{s}_\infty; l)\bar{e}_i(l) \\ \bar{s}_\infty(0) &= \bar{s}^0 \end{aligned} \tag{4}$$

where $\Pr(m(t) = l)$ is the equilibrium probability of the level-process being in level l given the state $\bar{s}_\infty(t)$.

Condition 1. *For any fixed value of $\bar{x} \in \mathbb{R}^D$, the Markov process evolving over the levels $\{0, 1, \ldots, L-1\}$ with transition rate $\nu_i(\bar{x}; m)$ of going to level $\tilde{m}(m, i)$ from level m, is ergodic.*

Condition 2. *The functions $\log \nu_i(\bar{x}; y)$ are bounded and Lipschitz continuous in $\bar{x}$ for every y (where continuity is in all the D co-ordinates).*

Note: The above conditions are to be checked for the unscaled process $(\bar{x}; m)$ and not for the scaled process $(\bar{s}; m)$.

3 The Discrete System

We now study the case where n balls are dropped into n bins sequentially using the $(1,1)$-memory policy. (Our analysis easily generalizes to cn balls being dropped into n bins for any constant c; we follow standard practice by focusing on the case where the average load is 1.) We denote by time t the time before the tth ball is dropped. Let $s_i(t)$ be the fraction of bins with load at least i at time t; and similarly, let $p_i(t)$ be the probability that the bin in memory has load at least i at time t. Then it follows that

$$E[s_i(t+1) - s_i(t)|s(t)] = [s_{i-1}(t)p_{i-1}(t) - s_i(t)p_i(t)]/n.$$

That is, s_i increases by $1/n$ when both the randomly selected bin and the bin from memory both have load at least $i-1$, but both do not have load at least i.

If we re-scale time so that it runs from 0 to 1, and let $\Delta t = 1/n$, then we have the following equation

$$\frac{E[s_i(t+\Delta t) - s_i(t)|s(t)]}{\Delta t} = [s_{i-1}(t)p_{i-1}(t) - s_i(t)p_i(t)].$$

More formally, in the large deviation framework, we have that in the limiting process the system follows the path

$$\frac{ds_i}{dt} = s_{i-1}p_{i-1} - s_i p_i.$$

Here the load of the bin in memory is the external Markov process referred to in Theorem 1; p_i represents the equilibrium probability (at the time t) that the bin in memory has load at least i.

Note that the p_i's are governed by the following equation:

$$p_i(t+1) = (p_{i-1}(t) - p_i(t))s_i(t) + p_i(t)(s_{i-1}(t) - s_i(t)) + p_i(t)s_i(t).$$

The first term corresponds to the case where the bin in memory obtains a ball and reaches load i; the second term corresponds to the randomly chosen bin obtaining a ball and reaching load i; and the third term corresponds to the case where both choices have load at least i. Recall the discussion from the section on large deviations; the chain governing the p_i values runs at a much faster rate than the chain for the s_i values. Hence, in considering the p_i, we take the equilibrium distribution for the p_i values given a fixed set of values for the s_i. In this equilibrium, we have

$$p_i = p_{i-1}s_i + p_i(s_{i-1} - s_i),$$

or equivalently

$$p_i = \frac{p_{i-1}s_i}{1 - s_{i-1} + s_i}.$$

Given these equations for p_i, we can substitute so that all equations are in terms of the s_i. The differential equations can then by solved numerically.

Formally, using the large deviation theory, we may state the following:

Theorem 2. *For any fixed constant K, for $1 \le i \le K$ let $s_i(1)$ be the solution for the s_i at time 1 in the differential equations above. For $1 \le i \le K$, let X_i be the random variable denoting the fraction of bins with i or more balls when we throw n balls into n bins using the $(1,1)$-memory policy. Then for sufficiently large n*

$$\Pr(|X_i - s_i(1)| > \epsilon) \quad \le \quad C_1 \exp(-nC_2(\epsilon)), \tag{5}$$

where C_1 is a constant that depends on K and $C_2(\epsilon)$ is a constant that depends on K and ϵ.

Proof. This follows from Theorem 1. To be clear, we emphasize how we restrict the system to be finite dimensional. This is easily accomplished by ignoring loads above K. That is, we restrict ourselves to examining $s_1, s_2, \dots, s_K$; since the evolution of the s_i for $i \leq K$ only depend on $s_1, s_2, \dots, s_K$, we do not lose anything by failing to track higher dimensions. The conditions of Theorem 1 are easily checked, and the Chernoff-like bound of the theorem yields the results. $\square$

Unfortunately, Theorem 1 as given by Shwartz and Weiss only applies to finite dimensional systems, and hence we cannot directly use it to prove (rigorously) that the maximum load is $O(\log \log n)$. We can, however, use this framework to gain the proper insight into the problem, and then use a layered induction technique (of [12], as described in [7]) to get rigorous bounds. This approach is described for example in [9] and [8]; here we simply use the differential equations.

Theorem 3. *When throwing n balls into n bins using a $(1,1)$-memory policy, the maximum load is $\log \log n / 2 \log \tau + O(1)$ with high probability.*

Proof. We sketch the proof using the differential equations above, and note that only minor additional work is necessary to obtain a formal layered induction proof. Let us temporarily assume that we had $p_i = p_{i-1}s_i$. Then,

$$\frac{ds_i}{dt} \leq s_{i-1}p_{i-1} = s_{i-1}^2 p_{i-2} = s_{i-1}^2 s_{i-2} s_{i-3} \dots s_1.$$

This holds for all $t \leq 1$. Hence

$$s_i(1) \leq s_{i-1}^2(1) s_{i-2}(1) s_{i-3}(1) \dots s_1(1).$$

Inductively, we find that $s_i(1) \leq s_1(1)^{F_{2i-1}}$, where F_n is the nth Fibonacci number. (Here $F_1 = F_2 = 1$.) To see this another way, let us suppose $s_i(1) \leq s_1(1)^{G_i}$. Then we find that G_i should satisfy the recurrence $G_i \geq 2G_{i-1} + \sum_{j=1}^{i-2} G_j$; we easily find that $G_i \geq F_{2i-1}$ by induction. When $s_i(1)$ falls below $1/n$, the expected number of bins with load greater than or equal to i falls below 1. Hence from the above we can conclude that the maximum load grows like $\frac{\log \log n}{2 \log \tau} + O(1)$ by finding where $s_i(1)$ falls below $1/n$. To revise the argument to take into account the actual formula $p_i = p_{i-1}s_i/(1 - s_{i-1} + s_i)$, we note that for sufficiently large i the denominator becomes arbitrarily close to 1. In particular, for any constant $\gamma > 0$ there exists a constant j so that for all $i \geq j$, $p_i \leq (1+\gamma)p_{i-1}s_i$. Hence for all suitably large i

$$s_i(1) \leq s_{i-1}^2(1) s_{i-2}(1) s_{i-3}(1) \dots s_j(1) p_{j-1} (1+\gamma)^{j-i}.$$

Choose j to be a large enough constant so that $s_j(1) \leq c/(1+\gamma)$ for some constant $c < 1$. Inductively, we find that for $i > j$, $s_i(1) \leq c^{F_{2(i-j)+1}}/(1+\gamma)$, from which we may again conclude that the maximum load grows like $\frac{\log \log n}{2 \log \tau} + O(1)$. $\square$

We note that similarly a lower bound for the $(1,1)$-memory policy could be proven, to show that in fact the maximum load is $\frac{\log \log n}{2 \log \tau} + \Theta(1)$.

Following the same framework, for the general case of a $(d,1)$ system where $d > 1$, the relevant equations are:

$$\frac{ds_i}{dt} = s_{i-1}^d p_{i-1} - s_i^d p_i$$

and

$$p_i = \frac{p_{i-1}s_i^d}{1 - d(s_{i-1} - s_i)s_{i-1}^{d-1}}.$$

Following the same line of reasoning as in Theorem 3, we find that for sufficiently large i and a suitably large constant j, $s_i(1)$ falls like $c^{G_{(j-i)+1}}/(1+\gamma)$, where here G satisfies the recurrence $G_i \geq 2dG_{i-1} + d\sum_{j=1}^{i-2} G_j$. This recurrence is easily solved to find

$$G_i \sim \left(\frac{2d+1+\sqrt{4d^2+1}}{2}\right)^i.$$

Let us define

$$f(d) = (2d + 1 + \sqrt{4d^2+1})/2. \tag{6}$$

Note $f(d) \in (2d, 2d+1)$; for large d, $f(d)$ is slightly more than $2d + 1/2$. Then, we have that the maximum load for a $(d, 1)$ system is $\frac{\log\log n}{\log f(d)} + O(1)$ with high probability. Hence the tails of a $(d, 1)$-memory system behave like a D-random system where D is between $2d$ and $2d + 1$. Phrased more intuitively, the value of the one spot of memory is worth somewhere between d and $d + 1$ additional choices.

4 The Continuous System

We wish to use Theorem 1 to analyze the supermarket model operating under the $(d, 1)$-memory policy. The state of the supermarket model, however, is infinite dimensional, as the load at the queue can be any positive integer. Similarly, the memory (which is the level process) can also take countably infinite values. Obviously, we need to truncate the supermarket model to be able to use Theorem 1, as we did for the discrete case. Hence we define auxiliary systems in which the truncation occurs naturally and which approximate the supermarket model.

More precisely, we shall define two systems—S_L^b and S_L^w— which perform, respectively, better and worse than the supermarket model under the $(d, 1)$-memory policy. These systems are analyzable as finite-dimensional finite-level jump Markov processes. We first obtain the defining equations for these systems as at (4) and then let L go to ∞. Now, for each L, the load of the systems S_L^b and S_L^w sandwich the load of the supermarket model. When L goes to ∞, the sandwich closes, yielding the performance of the supermarket model. There are subtleties in truncating and in taking the limit $L \to \infty$, as described below.

4.1 Analysis

Consider the supermarket model under the $(d, 1)$-memory policy with n servers and Poisson arrivals of rate $n\lambda$. Following [7], let the state at time t be $s(t) = (s_i(t))_{i\geq 0}$, where $s_i(t)$ denotes the fraction of queues with load at least i (we drop reference to scale parameter n in representing the state $\bar{s}$ as it is clear by the context). Let $m(t)$ denote the load of the queue in memory at time t. The overall state is $(s(t), m(t))$. We now define the better and worse systems: S_L^b and S_L^w.

Definition 2. **S^b_L:** Consider the supermarket model under the$(d,1)$-memory policy with the modification that the buffers are truncated at level L. Accordingly, any arrival occurring to a queue with size L is dropped. The state of the system[1] is truncated to $(s_0,\dots,s_L)$, and the load of the memory queue can be in $\{0,1,\dots,L\}$.

We formalize the notion under which S^b_L is better than the original system with the following definition.

Definition 3. A vector $u=(u_1,\dots,u_n)$ is majorized by $v=(v_1,\dots,v_n)$ if for $i \le i \le n$ we have $\sum_{1\le j\le i} u_{\pi(j)} \le \sum_{1\le j\le i} v_{\sigma(j)}$, where π and σ are permutations such that $u_{\pi(1)} \ge \dots u_{\pi(n)}$ and $v_{\sigma(1)} \ge \dots u_{\sigma(n)}$. For two allocation schemes X and Y on n queues, we say X is majorized by Y if $u(t)=(u_1(t),\dots,u_n(t))$ represent the loads of X at time t, $v(t)=(v_1(t),\dots,v_n(t))$ represent the loads of Y at time t, and there is a coupling between the two schemes so that $u(t)$ is majorized by $v(t)$ at all times t.

It is easy to see through straightforward coupling arguments that the load in S^b_L is majorized by the load in the original $(d,1)$ system. Upon any arrival, if any of the $d+1$ possible destination queues has load at most L, then there will be no overflows and the two systems behave similarly. If all of the $d+1$ queues have load equal to L, then whereas the load in the original system increases, the load of S^b_L does not.

The infinitesimal generator of this Markov process is defined as follows: For $f:\mathbb{R}^{2L+2}\to\mathbb{R}$, f continuous,

$$\begin{aligned} A^b_n f(s;m) \;&=\; \sum_{j\le m} \lambda(s^d_{j-1}-s^d_j)(f(s+e_j/n)-f(s)) + s^d_m(f(s+e_m/n)-f(s)) \\ &\quad + \sum_{0\le k\le L} (s_k-s_{k+1})(f(s-e_k/n)-f(s)) \end{aligned}$$

The S^b_L is finite-dimensional finite-level process. Hence, we can apply Theorem 1 after checking conditions 1 and 2. We would like to remind the reader that the conditions need to be checked for the unscaled version of the process. The scaling in S^b_L comes due to the size of the system. Hence, for any finite-sized version $n=n_0$ of the system S^b_L can be considered as the unscaled version of the system. As before, let the state be $(\bar{x};m)$, where $\bar{x}=(x_0,\dots,x_L)$ with x_i represents the fraction of queues with load at least i. The x_is are of the form $q/n_0, q\in\mathbb{N}, q\le n_0$. The conditions are checked as follows:
Checking condition 1: The memory process is an aperiodic, irreducible, finite-state Markov process for any state of loading $\bar{x}$, and is therefore ergodic.
Checking condition 2: This condition is related to checking the bound and continuity of transition rates. The infinitesimal generator of the process S^b_L suggests that the transition rates $\nu_i(\bar{x};m)$ due to arrival or departure are finite size polynomials of load vector $\bar{x}$. This implies that $\log\nu_i(\bar{x};m)$ is Lipschitz continuous in co-ordinates of $\bar{x}$. The bound $x_i\le 1$ gives an upper bound. If because one or more $x_i=0$ the transition rate $\nu_i(\bar{x};m)=0$, then it means the transition is absent and we neglect it. Otherwise $x_i\ge 1/n_0$, giving the lower bound on the $\log\nu_i(\bar{x};m)$. This completes the check of condition 2.

[1]In order not to introduce extra symbols, we use $s(t)$ to denote the state of the truncated systems as well.

Now we can apply Theorem 1 to S_L^b to obtain the behavior of system as $n \to \infty$, described by the following set of equations:

$$\begin{aligned}
\frac{ds_i(t)}{dt} &= \lambda[p_{i-1}(t)s_{i-1}(t)^d - p_i(t)s_i(t)^d] - [s_i(t) - s_{i+1}(t)], \quad \text{for} \quad i < L && (7)\\
\frac{ds_L(t)}{dt} &= \lambda[p_{L-1}(t)s_{L-1}(t)^d - p_L(t)s_L(t)^d] - [s_L(t)], \quad \text{and} && (8)\\
s_{L+1}(t) &= 0. && (9)\\
s_0(t) &= 1, s_0(0) = 1, s_i(0) = 0, 1 \leq i \leq L.
\end{aligned}$$

and,

$$\begin{aligned}
p_0(t) &= 1,\\
p_i(t) &= p_{i-1}(t)s_i(t)^d + d(s_{i-1}(t) - s_i(t))s_i(t)^{d-1}p_i(t), \quad \text{for } i \leq L && (10)\\
p_{L+1}(t) &= 0. && (11)
\end{aligned}$$

Next, we consider the worse system.

Definition 4. S_L^w: In the supermarket model, modify the $(d,1)$-memory policy as follows: at the arrival of a customer, if any of the d random queues and 1 memory queue has load smaller than L, follow usual $(d,1)$ policy. Otherwise, ignore the memory and follow the d-random policy. (The memory is restored when a future customer has a choice with load smaller than L.) Thus, if the queue in memory has load larger than L, it is irrelevant. Let us denote the state of the system by $(s_0(t), \ldots, s_L(t), F(t))$, where $F(t)$ represents the fraction of queues with load higher than L. The memory takes various values $\{0, 1, \ldots, L, L+1\}$, where $L+1$ represents memory taking any value higher than L.

We have used the same notation to represent the states for S_L^b and S_L^w, although the meaing should be clear by context. Note that we have truncated the system so that all load larger than L are merged into a single group. Again, it is easy to see via a simple coupling argument that S_L^w majorizes the original system, as the use of memory only helps.

We can similarly write the infinitesimal generator for the S_L^w. The S_L^w, as considered, is finite-dimensional finite-level Markov process. Hence we can apply Theorem 1. The conditions 1 and 2 can be checked in a similar way as done for S_L^b. For $n \to \infty$, for S_L^w, we obtain that the limiting behavior is given by the following equations:

$$\begin{aligned}
\frac{ds_i(t)}{dt} &= \lambda[p_{i-1}(t)s_{i-1}(t)^d - p_i(t)s_i(t)^d] - [s_i(t) - s_{i+1}(t)], \quad \text{for} \quad i < L && (12)\\
\frac{ds_L(t)}{dt} &= \lambda[p_{L-1}(t)s_{l-1}(t)^d - p_L(t)s_L(t)^d] - [s_L(t) - F(t)], \quad \text{and} && (13)\\
F(t) &\leq \lambda^{L+1}. && (14)\\
s_0(t) &= 1, s_0(0) = 1,\ s_i(0) = 0,\ 1 \leq i \leq L,\ F(0) = 0.
\end{aligned}$$

and,

$$\begin{aligned}
p_0(t) &= 1,\\
p_i(t) &= p_{i-1}(t)s_i(t)^d + d(s_{i-1}(t) - s_i(t))s_i(t)^{d-1}p_i(t), \quad \text{for } i \leq L && (15)\\
p_{L+1}(t) &= p_L(t)F(t)^d + d(s_L(t) - F(t))F(t)^{d-1}p_{L+1}(t). && (16)
\end{aligned}$$

The justification for (14) is as follows: The system S_L^w is majorized by the system where each incoming task chooses a single random queue and queues there. For such a system each queue behaves like an M/M/1 queue with arrival rate λ and service rate 1. The queue-size distribution for the M/M/1 queue suggests that $Pr(Q \geq i) \leq \lambda^i$, with an inequality when $i < 1$ and the system starts empty. In the limiting system as n to infinity, the fraction of queues with size $i \geq 1$ is therefore always smaller than λ^i if the system starts empty. This yields the bound of (14).

We would like to take the limit $L \to \infty$ for both S_L^b and S_L^w to obtain the performance of their corresponding limiting systems with respect to L. It is important to show that the limits exist, and that they are equal.

For $L_1 < L_2$, the loading of $S_{L_1}^b$ is majorized by the loading of $S_{L_2}^b$ as the dropping rate in the former is higher than that in the latter. Further, co-ordinate wise, these systems are bounded above by the 1-random system—which is a set of parallel M/M/1 queues. Thus, in each coordinate S_L^b is a bounded and monotonically increasing system in L and hence in each coordinate it converges to a limit. Similarly, in each coordinate S_L^w is bounded below and monotonically decreasing as L increases and hence in each coordinate a limit exists.

The next step is to show that these two limits are same and hence give us the exact performance for the original system with $(d,1)$-memory policy. The equations (7)-(11) for S_L^b and (12)-(16) for S_L^w demonstrates that the infinitesimal generators of these deterministic processes differ only in their tail co-ordinate (the $s_{L+1}(t) = 0$ of S_L^b and $F(t) \leq \lambda^{L+1}$ of S_L^w). The explicit bound of (14) ensures that, for any $\epsilon > 0$, there exists $L \geq L(\epsilon)$ such that the difference in these generators is bounded absolutely by ϵ. Hence, these systems converge to the same limit, as the generator of uniquely determines the limit. Thus, the limit of these two systems can be characterized by the following set of evolution equations, which also describes the evolution of the system with the $(d,1)$ policy:

$$\begin{aligned} \frac{ds_i(t)}{dt} &= \lambda[p_{i-1}(t)s_{i-1}(t)^d - p_i(t)s_i(t)^d] - [s_i(t) - s_{i+1}(t)], && (17) \\ s_0(t) &= 1, \ s_i(0) = 0, \quad i \geq 1. && (18) \end{aligned}$$

and,

$$\begin{aligned} p_0(t) &= 1, \\ p_i(t) &= p_{i-1}(t)s_i(t)^d + d(s_{i-1}(t) - s_i(t))s_i(t)^{d-1}p_i(t), \quad \text{for all } i && (19) \end{aligned}$$

Thus we conclude the following theorem.

Theorem 4. *Consider the supermarket system with n queues starting empty ($\bar{s}^0 = \bar{0}$) under $(d,1)$-memory policy. Let $\bar{s}_n(t)$ represent the tails of the load at time t, and let $\bar{s}_\infty(t)$ represent the tails as determined by the limiting differential equations above. Then there exist constants $C_1, C_2(\epsilon)$ and n_0 such that*

$$\Pr_{\bar{s}^0,m}\left(\sup_{0 \leq t \leq T} |\bar{s}_n(t) - \bar{s}_\infty(t)| > \epsilon\right) \leq C_1 \exp(-nC_2(\epsilon)). \qquad (20)$$

4.2 Approximate behavior

Theorem 4 gives the limiting evolution for the system employing the $(d, 1)$-memory policy. We can gain more insight by considering the fixed point of this system, where the derivatives are all 0, following the tack of [7]. (Note that [7] proves that the d-random system converges to its fixed point; we do not yet have an equivalent proof for this system, although all experimental evidence suggests convergence happens.) By comparing the fixed point with that of the d-random policy, we can qualitatively compare the load balancing policies.

Let $s^* = (s_0^*, \ldots, s_k^*, \ldots)$ be the fixed point of equation (17), and let $p^* = (p_0^*, \ldots)$ be the corresponding solutions for (19). Then,

$$s_i^* = s_{i-1}^* + \lambda p_{i-1}^* (s_{i-1}^*)^d - \lambda p_{i-2}^* (s_{i-2}^*)^d \tag{21}$$

$$p_i^* = \frac{p_{i-1}^* (s_i^*)^d}{1 - d(s_{i-1}^* - s_i^*)(s_i^*)^{d-1}} \tag{22}$$

Note the similarity between these equations and the equations obtained for the $(d, 1)$-memory policy in the discrete balls and bins model. Recall also the boundary conditions $s_0^*(t) = 1$, $p_0^*(t) = 1$, and $s_1^*(t) = \lambda$. (The last corresponds to the fact that the arrival rate must equal the departure rate.) From these we can iteratively determine the s_i^* and p_i^*; the fixed point exists and is in fact unique. A simple induction based on equation (21) reveals that $s_i^* = \lambda p_{i-1}^* (s_{i-1}^*)^d$. Hence, following the argument used in section 3, we obtain

$$s_i^* \leq \frac{\lambda^{\frac{f(d)^{i-c}-1}{f(d)-1}}}{1+\gamma} \tag{23}$$

for all $i \geq c$ for some suitable constants c and γ. As noted before, $f(d) \in (2d, 2d + 1)$. Hence, past some point, the tails of the memory system using $(d, 1)$-memory policy are bounded above by tails of the system with $2d$-random policy and bounded below by the tails of $(2d+1)$-random policy at the fixed point.

Theorem 5. *For a given λ, there exists a constant c so that the tails of the fixed point distribution beyond the cth coordinate of the supermarket model with the $(d, 1)$-memory policy are dominated by the tails at fixed point distribution of the model under the $2d$-random policy, and they dominate the tails of the model under the $2d + 1$-random policy.*

5 Experiments and Numerical Calculations

Discrete system: To provide more insight into the behavior of the load balancing policies with memory, we numerically calculate the fraction of bins with load at least i for the $(1, 1)$-memory policy when n balls are thrown into n bins and compare them to the corresponding numbers for the 2-random and 2-left policies. Results are shown in Table 1. Terms that are less than 10^{-100} are not shown. As can be seen, the tails for $(1, 1)$-memory policy initially decrease more slowly than the 2-random, but then the decrease accelerates in a fashion similar to the 2-left policy. This behavior is what one would expect from Theorem 3.

These numerical results accurately match simulation results. For example, we ran 1,000 trials of all three policies, each with 100,000 bins and balls. For the 2-memory system, there

Discrete system			
i	2-random	$(1,1)$-memory	2-left
1	7.6e-1	6.3e-1	7.7e-1
2	2.3e-1	3.3e-1	2.2e-1
3	8.9e-3	3.8e-2	4.4e-3
4	6.0e-6	8.4e-5	5.2e-8
5	1.3e-12	6.1e-12	1.2e-21
6	3.2e-26	1.0e-30	5.3e-58
7	8.9e-54	6.8e-80	

Continuous system $\lambda = 0.9$			
i	2-random	$(1,1)$-memory	2-left
1	9.0e-1	9.0e-1	9.0e-1
2	7.3e-1	8.1e-1	7.3e-1
3	4.8e-1	6.5e-1	4.8e-1
4	2.1e-1	4.0e-1	2.0e-1
5	3.8e-2	1.3e-1	3.2e-2
6	1.3e-3	8.0e-3	5.8e-4
7	1.5e-6	4.4e-6	2.5e-8
8	2.1e-12	1.1e-14	9.9e-20
9	4.1e-24	2.9e-37	1.4e-49

Table 1: Results from numerically solving the differential equations, for 2-random, $(1,1)$-memory, and 2-left.

were 8,390 bins of load four over all trials, closely matching the results from the differential equations. Over the 1,000 trials, 2-left had a maximum load of four for only seven trials, and a maximum load of three in the remaining trials; 2-random had a maximum load of four in 462 trials, and a maximum load of three in the remaining trials; and 2-memory had a maximum load of four for all 1,000 trials. Hence, despite the fact that 2-memory is asymptotically better than 2-random, it performs slightly worse in this experiment, as the numerical results of Table 1 would suggest.

The key point of these results is that the $O(1)$ constants are important to the actual behavior for these variations on realistic sizes, and hence the ability to accurately predict performance from the differential equations is quite useful.

Continuous system: As with the discrete case, in the queueing scenario we can numerically compare the $(1,1)$-memory policy with the 2-random and 2-left policies for specific values of λ. Here, we compute the behavior at the fixed point for comparison. For example, Table 1 presents the tails s_i for all three policies at $\lambda = 0.9$, along with the average time in the system as computed at the fixed point.

Table 1 shows similar behaviors as in the discrete case. In particular, because the $(1,1)$-memory policy has slowly decreasing tails initially, the average time in the system is slightly greater than for the 2-random and 2-left policies. Still, the $(1,1)$-memory policy performs dramatically better than a system with just one random choice.

Again, simulation results match the above results quite well. For example, we ran 20 trials of all three policies with 100 queues over 50,000 units of time at $\lambda = 0.9$. We record the time spent in the system for all tasks after time 5,000; the initial time allows the system to reach the equilibrium state. For the 2-memory system, the average time in the system for our simulations was 3.229, compared to 2.646 for the 2-random system and 2.634 for the 2-left system. All of these results closely match the predicted value for the average time in system as calculated using the values from the table above. Again, the $O(1)$ constants are important for the actual behavior for these variations on realistic sizes, and hence the ability to accurately predict performance from the differential equations is quite useful.

6 Conclusions

We have been motivated to examine load balancing systems with memory by several recent results. For example, it has been shown in asymmetric cases, where queues serve at different rates, using random choices is not enough. The system can be unstable when the net arrival rate is less than the net service capacity, even with $O(n)$ choices per incoming customer. Using just one sample with one unit of memory, however, ensures stability [11]. This demonstrates that a policy using a small amount of memory can “learn” things about a system that simply using many choices cannot accomplish.

Here, we have focused on obtaining exact bounds for limiting cases of both the discrete balls and bins problem and the continuous queuing problem under uniform service rates. We have demonstrated that asymptotically choice plus memory yields more rapidly decreasing tails than an equal number of choices alone. Specifically, the $(d, 1)$-memory policy behaves somewhere between a $2d$-random and a $(2d + 1)$-random policy.

A further contribution of this paper is the application of the finite-dimensional finte-level jump Markov process framework to analyze load balancing systems with memory. We believe these methods will apply more widely, both for other load balancing variations such as asymmetric systems as well as other problems.

References

[1] Y. Azar, A. Broder, A. Karlin, and E. Upfal. Balanced Allocations. *SIAM Journal of Computing*, 29:1, pp. 180–200, 1999.

[2] A. Czumaj and V. Stemann. Randomized Allocation Processes. In *Proceeding of the 38th FOCS*, pp. 194–203, 1997.

[3] M. Dahlin. Interpreting Stale Load Information. *IEEE Transactions on Parallel and Distributed Systems*, 11(10), pp. 1033–1047, October 2000.

[4] S. N. Ethier and T. G. Kurtz. **Markov Processes : Characterization and Convergence.** John Wiley & Sons, New York, 1986.

[5] P. Giaccone, B. Prabhakar, and D. Shah. Towards Simple High-Performance Scheduler for Input Queued Switches. To appear at IEEE INFOCOM 2002.

[6] R. J. Gibbons, F. P. Kelly, and P. B. Key. Dynamic Alternative Routing - Modelling and Behavior, In *Proceedings of the 12 International Teletraffic Congress*, June 1988.

[7] M. Mitzenmacher. The Power of Two Choices in Randomized Load Balancing, PhD thesis. University of California, Berkeley, 1996. Journal article appears as: The Power of Two Choices in Randomized Load Balancing, *IEEE Transactions on Parallel and Distributed Systems*, 12:10, pp. 1094-1104, October 2001.

[8] M. Mitzenmacher, A. Richa, and R. Sitaraman. The Power of Two Random Choices: a Survey of Techniques and Results. In the **Handbook of Randomized Computing**, vol. 1, pp. 255-312, edited by P. Pardalos, S. Rajasekaran, J. Reif, and J. Rolim, Kluwer Press.

[9] M. Mitzenmacher, and B. Vöcking. The Asymptotics of Selecting the Shortest of Two, Improved. In *Proceedings of 27th Annual Allerton Conference on Communication, Control, and Computing*, pp. 326-327, 1999. To appear in a memorial volume for F. Karpelevich.

[10] K. Psounis, B. Prabhakar and D. Engler. A Randomized Cache Replacement Scheme for Approximating LRU. In *Proceedings of the* 34th *Annual Conf. of Information Sciences and Systems*, Princeton, March 2000.

[11] D. Shah and B. Prabhakar. The Use of Memory in Randomized Load Balancing. To appear at ISIT 2002.

[12] A. Shwartz and A. Weiss. **Large Deviations for Performance Analysis: Queues, Communication, and Computing.** Stochastic Modelling Series, Chapman & Hall, London, 1995.

[13] L. Tassiulas. Linear Complexity Algorithms for Maximum Throughput in Radio Networks and Input Queued Switches. In *Proceedings IEEE INFOCOM 1998*, vol. 2, pp. 533-539, New York, NY.

[14] B. Vöcking. How Asymmetry Helps Load Balancing. In *Proceeding of* 40th *IEEE-FOCS*, New York, 1999; pp. 131-140.

[15] N. D. Vvedenskaya, R. L. Dobrushin and F. I. Karpelevich. Queueing System with Selection of the Shortest of Two Queues : An Asymptotic Approach. *Problems of Information Transmission*, 32(1):15-29, 1996.