The use of memory in randomized load balancing

Devavrat Shah
Dept. of CS
Stanford University
e-mail: devavrat@stanford.edu

Balaji Prabhakar
Depts. of EE & CS
Stanford University
e-mail: balaji@stanford.edu

Abstract — **We consider the following canonical load balancing problem: Drop n balls into n bins so as to minimize the maximum loading. An execution of the well-known "load the least loaded bin" algorithm results in an optimal loading of one ball per bin. Azar *et. al.* (1994) consider an algorithm which assigns each ball to the least loaded of d randomly chosen bins and show that the maximum load is $\frac{\ln\ln n}{\ln d} + O(1)$ for $d \geq 2$, as compared to $\frac{\ln n}{\ln\ln n}(1 + o(1))$ for $d = 1$. A dynamic version of the load balancing problem involves jobs arriving as a rate $n\lambda$ Poisson process at n rate 1 exponential server queues is considered in Mitzenmacher (1996) and in Vvedenskaya *et. al.* (1996). They find similar exponential improvements in performance for $d \geq 2$ as compared with $d = 1$.**

In this paper, we consider a variation of randomized load balancing schemes which involve the use of memory. This is motivated by the observation that the loads do not change by much between iterations; hence remembering "good samples" from one iteration for use in future iterations ought to pay off significantly. We find this is indeed true, and quantify the improvement.

I. Introduction

Load balancing is a fundamental problem arising in different practical situations. Perhaps a typical example is the scheduling of n jobs on n homogeneous processors so as to minimize the maximum loading, or to minimize the make-span of the jobs. One model for this problem is that of allocating n balls to n bins such that the maximum load is minimized. Another model (called the supermarket model) involves the allocation of jobs arriving as a rate $n\lambda$ Poisson process at a bank of n exponential server queues with service rates $\mu_1 \leq \ldots \leq \mu_n$, $\sum_i \mu_i = n$, so as maximize the throughput and minimize the backlogs. A centralized allocator who has knowledge of the load at each bin (queue) would recursively assign the balls (jobs) to the least loaded bin (queue). But this exhaustive information of the loading can be very expensive to gather when n is large.

To simplify the implementation, Azar *et. al.* [1] consider a natural randomized algorithm for the ball/bin problem: Suppose that each ball is recursively assigned to the least loaded of d randomly chosen bins, where $d \ll n$. When $d \geq 2$ they show that, with a high probability, this strategy leads to a maximum load of $\frac{\ln\ln n}{\ln d} + O(1)$. Contrasting this with the case $d = 1$ when the maximum load is approximately $\frac{\ln n}{\ln\ln n}(1+o(1))$ with a high probability, they observe an exponential improvement in performance in going from $d = 1$ to $d \geq 2$. Similar results are obtained for the supermarket model by Mitzenmacher [2] and Vvedenskaya et. al. [3].

[1]This work was supported by an Alfred P. Sloan Foundation Fellowship.

II. Load Balancing with Memory

We consider the following randomized load balancing policy, called the "(d, m) policy": Suppose that $d \geq m$. At the beginning of the t^{th} iteration the (d, m) system has stored in its memory the identity of the m least loaded bins at the end of the $(t-1)^{th}$ iteration. During the t^{th} iteration it selects d new bins, uniformly and independently of all else, and assigns the ball/job to the least loaded of these d bins and the m retained in memory[1]. It concludes the t^{th} iteration by writing the identity of the m least loaded of the $d + m$ bins into memory. Of course, at the beginning of the first iteration, the (d, m) system takes d samples and retains m of these for the second iteration.

III. Theorems

Theorem 1 *Consider the supermarket model and suppose that the service rates μ_i are all not equal (but $\sum_i \mu_i = n$). Then the following statements hold:*
(i) When sampling is done with replacement, we have instability for $\lambda < 1$ under the $(d, 0)$ assignment policy, even when $d = O(n)$.
(ii) We have stability for all $\lambda < 1$ under the $(1, 1)$ policy.

Theorem 1 strongly demonstrates the benefit of using memory to obtain maximum throughput.

Theorem 2 *The following statements hold the ball-bin model after all the balls are loaded.*
(i) The size of the maximum loaded bin under the $(d, 1)$ policy $(d \geq 2)$ is bounded by $\frac{\ln\ln n}{\ln(2d-1)} + O(1)$ with high probability.
(ii) The maximum load of the (d, m) system when all n balls are loaded is bounded by $\frac{\ln\ln n}{\ln((d-m/2)(m+1))} + O(1)$ with a high probability.

Theorem 2 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.

References

[1] Y. Azar, A. Broder, A. Karlin and E. Upfal. Balanced Allocations. In *ACM STOC*, pp-593–602,1994.

[2] M. Mitzenmacher. The power of two choices in randomized load balancing, PhD thesis. University of California, Berkeley, 1996

[3] 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*, Vol. 32, No. 1, pp-15–27,-1996.

[1]Sampling can be with or without replacement, except when made explicit.

0-7803-7501-7/02/$17.00©2002 IEEE.