DIMACS Technical Report 2003-16
April 2003

Class-Dependent Assignment in Cluster-based Server

by

Victoria Ungureanu[1]
Department of MSIS
Rutgers Business School - Newark and New Brunswick
Rutgers University
180 University Ave., Newark, NJ 07102
email:ungurean@rbs.rutgers.edu

Phillip G. Bradford
Department of Computer Science
The University of Alabama,
Box 870290,
Tuscaloosa, AL 35487-0290,
email: pgb@cs.ua.edu

Michael Katehakis Department of MSIS
Rutgers Business School - Newark and New Brunswick
Rutgers University
180 University Ave., Newark, NJ 07102
email: mnk@andromeda.rutgers.edu

Benjamin Melamed
Department of MSIS
Rutgers Business School - Newark and New Brunswick
Rutgers University
94 Rockafeller Rd., Piscataway, NJ 08854
email: melamed@rbs.rutgers.edu

[1]Work supported in part by DIMACS under contract STC-91-19999, and Information Technology and Electronic Commerce Clinic, Rutgers University

---

DIMACS is a partnership of Rutgers University, Princeton University, AT&T Labs-Research, Bell Labs, NEC Research Institute and Telcordia Technologies (formerly Bellcore).

DIMACS was founded as an NSF Science and Technology Center, and also receives support from the New Jersey Commission on Science and Technology.

ABSTRACT

A cluster-based server consists of a front-end dispatcher and several back-end servers. The dispatcher receives incoming requests, and then assigns them to back-end servers for processing. Our goal is to devise an assignment policy that has good response time performance, and is practical to implement in that the amount of information used by the dispatcher is relatively small, so that the attendant computation and communication overheads are low. In contrast to extant assignment policies that apply the same assignment policy to all incoming jobs, our approach calls for the dispatcher to classify incoming jobs as long or short, and then use class-dependent assignment policies. Specifically, we propose a policy, called `CDA` (Class Dependent Assignment), where short jobs are assigned Round-Robin as soon as they arrive, while long jobs are deferred and assigned only when a back-end server becomes idle. Furthermore, when processing a long job, a back-end server is not assigned any other jobs.

Our approach is motivated by empirical evidence suggesting that the sizes of files traveling on the Internet follow power-law distributions, where long jobs constituting a small fraction of all incoming jobs actually account for a large fraction of the overall load. To gauge the performance of the proposed policy, we exercised it on empirical data traces measured at Internet sites serving the 1998 World Cup. Since the assignment of long jobs incurs computational overhead as well as extra communication overhead, we studied the performance of `CDA` as a function of the fraction of jobs classified as long. Our study shows that classification of even a small fraction of jobs as long can have a profound impact on overall response time performance. More specifically, our experimental results show that if less than 3% of the jobs are classified as long, then `CDA` outperforms traditional policies, such as Round-Robin, by two orders of magnitude. From an implementation viewpoint, these results support our contention that `CDA`-based assignment is a practical policy combining low overhead and greatly improved performance.

Keywords: cluster-based server, assignment policy, power-law distribution, simulation

1 Introduction

Web servers are becoming increasingly critical as the Internet assumes an ever more central role in the telecommunications infrastructure. The satisfactory execution of common business applications (e.g., Web, multimedia and e-commerce activities, to name a few) depends on the efficient performance of Web servers. In particular, from a customer standpoint, a key Quality of Service (QoS) performance metric is response time. To improve service response times, it is necessary to take into consideration server architecture and Internet traffic loads.

In this paper we consider a cluster-based architecture for Web servers that combines good performance and low cost. A cluster-based server consists of a front-end dispatcher and several back-end servers (see Figure 1). The dispatcher receives incoming requests, and then decides how to assign them to back-end servers, which in turn serve the requests according to some discipline.

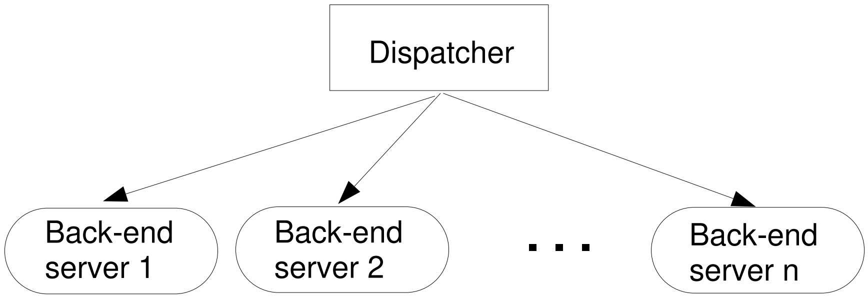

Figure 1: A cluster-based server

A number of dispatcher assignment policies have been proposed for this type of architecture, the most well-known of which are briefly described below. The `Round-Robin` policy assigns jobs to back-end servers in a cyclical manner, namely, the i-th job is assigned to server $i \bmod n$, where n is the number of back-end servers in the cluster. The `JSQ` (Join Shortest Queue) policy assigns each incoming job to the back-end server with the smallest amount of residual work. Finally, `Size-Range` policies are based on the observation that when "short" jobs are stuck behind "long" ones, then response time performance can degrade severely. Such policies avoid the pitfalls stemming from disparate job sizes at a given back-end server by assigning any back-end server only jobs of similar sizes. It has been shown that the response time performance of `JSQ` and `Size-Range` can be several orders of magnitude better than `Round-Robin` and its variants [8, 19]. However, the former are rarely, if at all, used in practice [16, 19] because they require the dispatcher to estimate the sizes of all incoming jobs and to know the status of back-end servers, thus incurring computation and communication overheads.

Our goal is to devise an assignment policy that has good response time performance and is practical to implement in that the amount of information used by the dispatcher is relatively small, so that the attendant computation and communication overheads are low. In contrast to extant assignment policies that apply *the same assignment policy to all incoming jobs*, our approach calls for the dispatcher to classify incoming jobs as long or short, and then use class-dependent assignment policies. Specifically, we propose a policy, called `CDA` (Class Dependent Assignment), where the dispatcher assigns short jobs as soon as they arrive in `Round-Robin` manner, while long jobs are assigned to back-end servers only when the servers become idle. Moreover, while a back-end server processes a long job, it will not be assigned any other jobs.

We argue that `CDA` achieves low response times for several reasons. First, `CDA` reduces the variance of job sizes at a given back-end server by assigning to it either long or short jobs, but not both, at any given time. Thus, it eliminates the problem of short jobs getting "stuck" behind long ones. Secondly, by deferring the assignment of long jobs, CDA effectively gives priority to short jobs. Finally, the assignment overhead in CDA is low, because the dispatcher need only estimate the size of long jobs and need only know when back-end servers become idle.

The size-dependent behavior of CDA is motivated by empirical evidence suggesting that the sizes of files traveling on the Internet follow power-law distributions [5, 1, 6] of the form

$$\mathbb{P}[X > x] \quad \sim \quad \frac{c}{x^{\alpha}},$$

where X is the random file size, $c > 0$ and $1 \leq \alpha \leq 2$. In power-law job-size distributions, *a relatively small fraction of jobs accounts for a relatively large fraction of the overall load.*

To gauge the performance of the CDA policy, we exercised it on empirical data traces measured at Internet sites serving the 1998 World Cup. We mention that Arlitt and Jin [1] shows that job sizes from these traces do indeed follow a power-law distribution with $\alpha = 1.37$. Moreover, for the traces considered in our experiments, less than 3% of the jobs account for more than 50% of the overall job sizes. Since the assignment of long jobs incurs more computational overhead as well as extra communication overhead, we studied the performance of CDA *as function of the fraction of jobs classified as long.* Our study demonstrates that the classification of even a small fraction of jobs as long can have a profound impact on overall response time performance. More specifically, our experimental results show that if less than 3% of the jobs are classified as long, then CDA outperforms Round-Robin, by two orders of magnitude. Furthermore, for the same fraction of long jobs, CDA yields response time metrics similar to JSQ. From an implementation viewpoint, these results support our contention that CDA is a practical policy combining low overhead and good response time performance.

The rest of the paper is organized as follows. Section 2 presents related work in the literature. Section 3 discusses the CDA policy and provides a detailed performance analysis for it. Section 4 discusses extensions of the CDA policy, and finally, Section 5 concludes the paper.

2 Previous Work

The literature contains a considerable body of work on job scheduling (see [2, 4, 10, 11, 12, 14] and references therein). Most of the analytical models assume that arrivals and service times follow exponential-type distributions. Smith [18] considered scheduling with fixed-size (deterministic) jobs on a single server. The paper shows that in this case, scheduling the shortest jobs first is optimal in that the algorithm yields minimal response times. In a similar vein, Rothkopf [15] shows that this algorithm yields minimal expected response times, for job sizes having arbitrary (known) distributions. Winston [21] considered a cluster-based server with the first-come first-serve (FCFS) discipline at each server queue, exponential job-size distributions, and Poisson arrivals. The paper proves that under these assumptions, the JSQ policy is optimal (yields minimal expected response times). However, Whitt [20] showed that there exist other job-size distributions for which JSQ is not optimal.

Next we proceed with a review of Size-Range scheduling policies that consider job sizes that follow a power-law distribution. Harchol-Balter *et al.* [8] introduce a policy called “Size Interval Task Assignment with Equal Load” (SITA-E). The SITA-E policy fits job-size ranges (intervals) to bounded-Pareto distributions, and then equalizes the expected work. That is, given n back-end servers, then n size ranges are determined off-line, such that each range contains approximately the same amount of work. Under certain job-size variance assumptions, they show that SITA-E outperforms JSQ [8]. Ciardo *et al.* [3] presents a load-balancing algorithm, called EquiLoad, similar to SITE-E in that it also uses predefined ranges. The paper shows that EquiLoad performs well on World Cup data traces. We mention that the main drawback of SITA-E and EquiLoad is that they assume a priori knowledge of the job-size distribution. Another algorithm, called AdaptLoad, is proposed in [13] as an adaptive, on-line version of EquiLoad. Again, AdaptLoad assigns each back-end server to a job-size range, but these ranges are continually re-evaluated based on the most recent history window of requested jobs. The paper shows empirically that for World Cup data traces, AdaptLoad performs similarly to JSQ.

3 Performance Study of `CDA`

Under the `CDA` policy, the dispatcher has a cutoff point parameter for classifying arriving jobs as long or short. Denoting by `CDA(c)` the `CDA` policy with cutoff `c`, following are its rules of operation.

1. *The dispatcher classifies incoming jobs into long or short relative to* `c`.
2. *The dispatcher assigns short jobs in Round-Robin manner as soon as they arrive. Back-end servers process jobs in their queues in the order of their arrival.*
3. *The dispatcher does not assign long jobs as soon as they arrive, but rather, holds them in its queue. When a back-end server becomes idle, the dispatcher assigns it the shortest of the (long) jobs in its queue. While the back-end server processes a long request, it will not be assigned any other jobs.*
4. *Any back-end server notifies the dispatcher when it becomes idle.*

We draw the reader's attention to the following points. First, the classification into long and short jobs is based on the *size of requested documents*. Although *job service time* is a more relevant classification metric pro forma, we nevertheless chose *size*, because the dispatcher has this type of information readily available. In contrast, it is very difficult, if at all possible, for the dispatcher to estimate service times with sufficient accuracy. This is so, because the service time for a given document depends on whether or not the document is cached, the status of the network and other imponderables. In a similar vein, it is difficult for the dispatcher to estimate the current workload at a back-end server, and thus to determine precisely when it becomes idle. This is why item 4 of the `CDA` policy calls for a back-end server to explicitly notify the dispatcher when it becomes idle.

Secondly, requiring explicit notification adds communication overhead. However, communication overhead has only a minor impact on system performance regardless of load (for specific experimental results see Section 4). To see that, note that under light load, job processing is not delayed due to back-end server notifications, while under heavy load such notifications are infrequent. It is worth noting that `JSQ` and `Size-Range` policies require the dispatcher to know the status of back-end servers at all times, thus incurring a considerably higher communication overhead than `CDA`.

3.1 Simulation Data

Our study used trace data from Internet sites serving the 1998 World Cup. The data used are archived in an Internet repository (see [1] and http://ita.ee.lbl.gov/html/traces.html). The repository provides detailed information about the 1.3 billion requests received by World-Cup sites over 92 days – from April 26, 1998 to July 26, 1998. We mention again that Arlitt and Jin [1] have shown that job sizes from World Cup traces follow a power-law distribution with $\alpha = 1.37$.

From these data, we selected a trace covering approximately 40 minutes from June 26, and containing over 3.5 million requests. Figure 3 depicts the number of requests received by the World Cup cluster in one-second intervals. The long tail exhibited by the data can be discerned in Figure 2, which depicts the number of requests made for selected file sizes in the trace considered (note the logarithmic scale on the y axis). It is worth pointing out the following aspects of the aforementioned trace.

- Approximately 75% of the files requested have sizes of less than 2KB, which account for less than 12% of the transferred data.

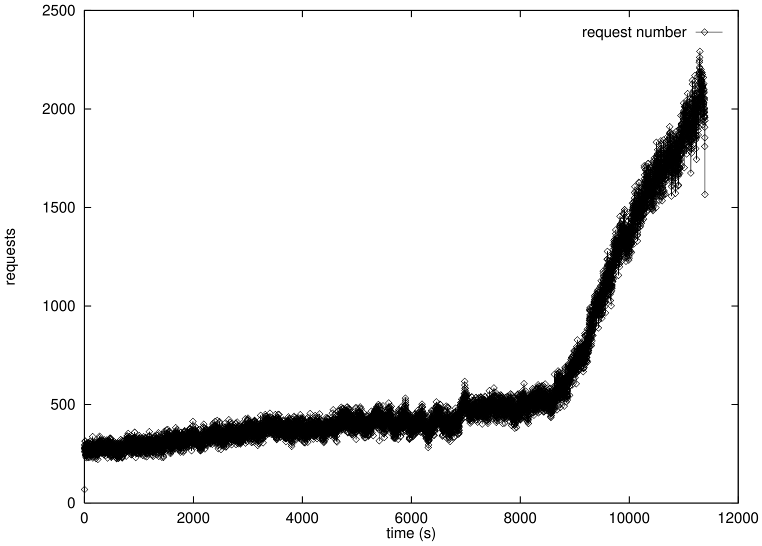

Figure 2: Number of request arrivals per second

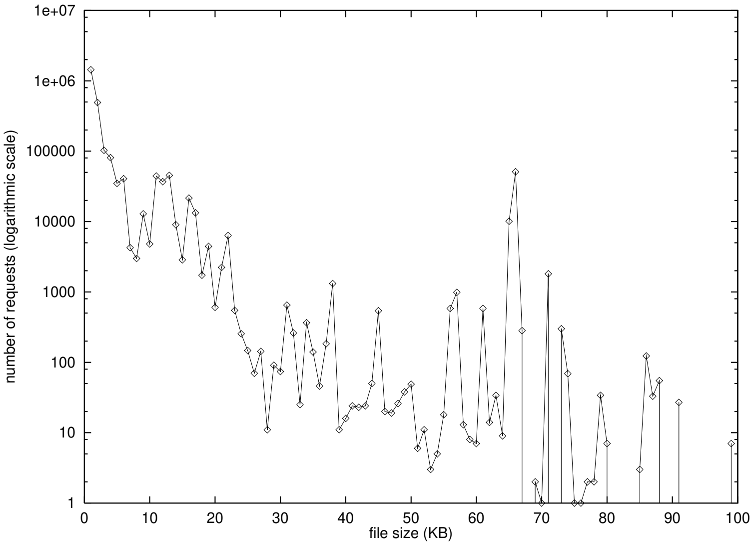

Figure 3: File size distribution of arriving requests (the y axis is in logarithmic scale)

- Files with sizes greater than 30KB, which make up less than 3% of the files requested, account for over 50% of the overall load. Even more striking, files in excess of 100KB, which make up less than 0.04% of all files requested, account for 7% of the transferred data.

From each trace record, we extracted only the request arrival time and the size of the requested file. We mention that the recorded time stamps are in integer seconds, with arrivals on the order of several hundreds of requests per second. Consequently, we have distributed request arrivals uniformly over each second. Since no information was recorded regarding service times, our simulation estimates these times as the sum of the (constant) time to establish and close a connection, and the (variable, size-dependent) time required to retrieve and transfer a file. The values used for these parameters have been determined experimentally in [16, 19], and are as follows: the constant time to make a connection is set at 0.9 ms, and the variable transmission and processing time is computed using 80μs/KB. We mention that the latter value applies only to static, cached files.

3.2 Simulation Experiments

The simulation study considers a cluster with four back-end servers, and makes the following simplifying assumptions:

1. Communication times between the dispatcher and back-end servers are negligible.
2. The overhead incurred by the dispatcher to select (job, server) pairs is negligible.
3. Request processing at a back-end server does not require disk access. This simplifying assumption is supported, for example, by [9, 19], which argue that essentially all files requested are cached during stable operation (following warm-up). We discuss how this and the other assumptions may be relaxed in Section 4.

The assignment policies compared are `CDA`, `Round-Robin` and `JSQ`. We excluded from the study `Size-Range` policies, because it was shown in [13] that their performance is similar to the `JSQ` policy for the trace used. In order to evaluate the relative efficacy of these policies, we compared their performance with respect to *waiting time*, defined as the time interval from the moment a request arrives at the dispatcher and up until it starts processing at the corresponding back-end server.

Figure 4 displays the average waiting time yielded by these policies for cutoff points, `c`, varying between 10KB and 500KB. The Figure shows that `CDA` improves over `Round-Robin` by a factor of four, even for a cutoff point of 500KB, for which long files account for less than 0.02% of the total number of requests. Moreover, this improvement increases greatly for smaller values of `c`. For example, for `c` = 30K (less than 3% of requested files are classified as long), the average waiting time of `CDA` is 6.6μs—two orders of magnitude lower than `Round-Robin`. On the other hand, `JSQ` is the best performer over all cutoff points, but is comparable to `CDA` for small values of `c` — the average waiting time of `JSQ` is 2.68μs as compared with 6.6μs for `CDA(30)`.

The results clearly show the trade-off between performance and the amount of information used by the dispatcher (with the attendant computational and communication overhead). Indeed, the `Round-Robin` policy (where the dispatcher does not use service time or size information) performs by far the worst, yielding an average waiting time of 826μs. In contrast, `JSQ` yields the best performance, but requires the dispatcher to know the status of *all* back-end servers at *all* times. Finally, `CDA` is in-between `Round-Robin` and `JSQ`, both in terms of waiting time performance and the amount of information required by the dispatcher. Specifically, `CDA` requires the dispatcher to know the sizes of just a fraction of the files. Furthermore, `CDA` enjoys the flexibility of trading performance for information through a choice of the cutoff point value, `c`.

Figure 5 displays the average waiting time of `JSQ` and `CDA(30)` over successive 10-second intervals spanning the entire simulation time horizon. The results for the `Round-Robin` policy are not plotted, because it is substantially outperformed by the other two policies. The results show that `CDA`'s performance is remarkably close to that of `JSQ`. This performance is all the more remarkable in view of `CDA`'s selectivity: it requires only long file information, while `JSQ` requires information on all files. In fact, `CDA` is designed to take advantage of power-law workload distributions. To wit, recall that 3% of requests are for long files, which account for more than 50% of total transferred data, while 75% of requests are for short files, which account for less than 12% of total transferred data. We conclude by pointing out the results above ignore communication

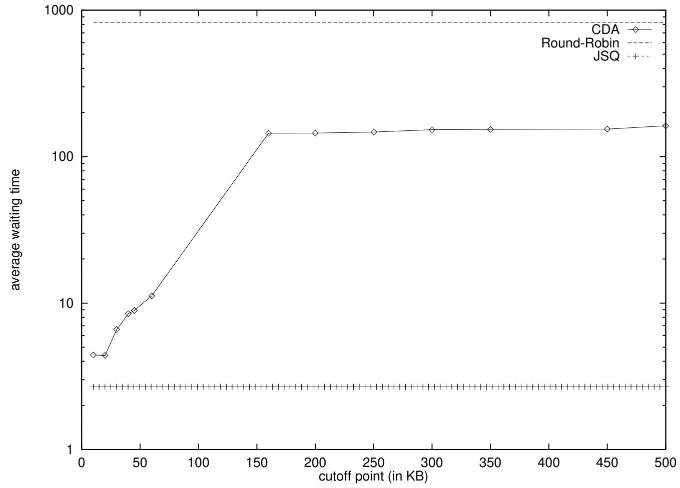

Figure 4: Average waiting time of `Round-Robin`, `JSQ` and `CDA` as function of cutoff point (the y axis is in logarithmic scale)

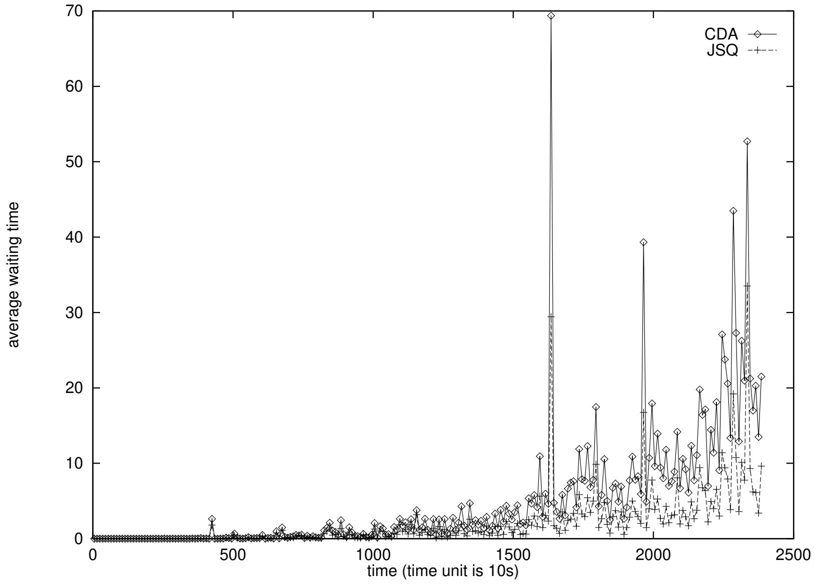

Figure 5: Average waiting times of `CDA(30)` and `JSQ` over successive 10-second intervals

overhead, so we expect the actual performance of `CDA`, relative to `JSQ`, to be even better. This issue if further discussed in the next section.

4 CDA Extensions

This section discusses how to relax some of the simplifying assumptions made in the simulation study of Section 3.2, and several issues pertaining to a practical implementation of `CDA`.

First, the simulation assumes that the communication overhead between the dispatcher and back-end servers is negligible. This is a reasonable assumption when the compared policies incur similar overheads. However, `CDA` calls for back-end servers to notify the dispatcher when they become idle, and thus require more dispatcher-server communication than `Round-Robin`. To study the effect of this extra communications, we have added a communication overhead of 100μs to the service time of long jobs (short jobs are assigned immediately, and thus are not delayed because of server notifications). If the fraction of jobs classified as long is very small, the impact is insignificant. For example, the average waiting time of `CDA(500)` is 162.53μs when taking into account communication overhead, as compared with 162.52μs when communication overhead was ignored. However, the impact of communication overhead becomes noticeable as the fraction of jobs classified as long increases. For example, the average waiting time of `CDA(30)` increases from 6.6μs to 7.66μs.

Secondly, the simulation study assumes that all requests are for static files, whose content, and therefore sizes, are fixed. In contrast, dynamic files are created on the fly by the server in response to a request. Dynamic files require considerably more processing time than static files [9]. A straightforward way of dealing with dynamic files is to classify the CGI programs creating this type of files as long. We note that treating requests for dynamic documents as long jobs does not change substantially the number of jobs considered as long: it was shown in [1] that for World Cup traces only 0.02% of requests were for dynamic files.

Finally, the simulation study assumes that all files are cached. This simplifying assumption was used to make the simulation easier, but it is not intrinsic to `CDA`. In fact, `CDA` can be extended to take into account cache misses and the resulting disk access. To this end, the back-end servers are required to report to the dispatcher whenever a long file cannot be served from the cache. The dispatcher, in turn, will assign short requests to that back-end server for the duration of the disk access, thereby utilizing its idle CPU cycles.

5 Conclusion

In this paper, we propose a novel approach to the job assignment problem in cluster-based servers, called `CDA`. In contrast to extant policies that use the same assignment method for all incoming jobs, our approach calls for the dispatcher to classify incoming jobs as long or short, and then use class-dependent assignment policies. We have studied the waiting time performance of `CDA`, and shown experimentally that `CDA` performs far better than `Round-Robin`. Furthermore, `CDA` performs similarly to `JSQ` even though the former requires considerably less information than the latter, and thus incurs less communication overhead. All things considered, it is possible that when communication overhead is taken into account, `CDA` would perform better than `JSQ`.

References

[1] Arlitt, M. and T. Jin. “Workload Characterization of the 1998 World Cup Web Site,” *IEEE Network*, Vol. 14, No. 3, 30-37, May/June 2000. Extended version: Tech Report H PL-1999-35R1, Hewlett-Packard Laboratories, September 1999.

[2] Bruckner, P. *Scheduling Algorithms*, Third Edition, Springer-Verlag, 2001.

[3] Ciardo, G., A. Riska, and E. Smirni. “EquiLoad: a Load Balancing Policy for Clustered Web Servers”, In *Performance Evaluation* 46(2-3): 101-124, 2001.

[4] Colajanni, M., P. S. Yu and D. M. Dias. ”Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems”, In *IEEE Transactions on Parallel and Distributed Systems*, 9(6), 1998.

[5] Crovella, M.E., M.S. Taqqu and A. Bestavros. "Heavy-tailed Probability Distributions in the World Wide Web," In *A Practical Guide To Heavy Tails*, Chapman Hall, New York, pp. 3–26, 1998.

[6] Faloutsos, M., P. Faloutsos and C. Faloutsos. "On Power-Law Relationships of the Internet Topology," In Proceedings of *ACM SIGCOMM '99*, 251-262, Aug. 1999.

[7] Harchol-Balter, M. "Task Assignment with Unknown Duration," *Journal of the ACM*, Vol. 49, No. 2, 260-288, March 2002. (Extended Abstract in *20th International Conference on Distributed Computing Systems (ICDCS '00), Taipei, Taiwan, April 2000.*)

[8] Harchol-Balter M., M.E. Crovella and C.D. Murta. "On Choosing a Talk Assignment Policy for a Distributed Server System," in *Proceedings of Performance Tools '98*, Lecture Notes in Computer Science, Vol. 1468, 231-242, 1998.

[9] Hu, Y., A. Nanda and Q. Yang. "Measurement, Analysis and Performance Improvement of the Apache Web Server", in *The International Journal of Computers and Their Applications*, 8(4), pp. 217-231, 2001.

[10] Katehakis, M. and C. Melolidakis. "On The Optimal Maintenance of Systems and Control of Arrivals in Queues", in *Stochastic Analysis and Applications*, 8 (2) 12-25, 1994.

[11] Pinedo, M. *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, 2002.

[12] Righter, R. "Scheduling in Multiclass Networks with Deterministic Service Times". In *Queuing Systems* 41(4): 305-319, 2002.

[13] Riska, A., W. Sun, E. Smirni and G. Ciardo. "AdaptLoad: Effective Balancing in Clustered Web Servers Under Transient Load Conditions," in *22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

[14] Ross, S.M. *Probability Models for Computer Science*, Academic Press, 2002.

[15] Rothkopf, M.H. "Scheduling with random service times" , in *Management Science*, 12;703-713, 1966.

[16] Pai, V.S., M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel and E. Nahum. "Locality-aware Request Distribution in Cluster-based Network Servers", in the *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, 1998.

[17] Pai, V.S., P. Druschel and W. Zwaenepoel. "Flash: An Efficient and Portable Web Server", in the *Proceedings of the USENIX 1999 Annual Technical Conference*, 1999.

[18] Smith, W.E. "Various optimizers for Single-Stage Production." *Naval Research Logistics Quarterly*, Vol. 3, 59-66, 1956.

[19] Teo Y.M. and R. Ayani, "Comparison of Load Balancing Strategies on Cluster-based Web Servers", In *Simulation, The Journal of the Society for Modeling and Simulation International*, pp. 185-195, Vol. 77, No. 5-6, November-December 2001.

[20] Whitt, W. "Deciding which Queue to Join: Some Counter Examples," *Operations Research*, Vol. 34, No. 1, 55-62, 1986.

[21] Winston, W. "Optimality of the Shortest Line Discipline," *Journal of Applied Probability*, 14:181–189, 1977.