Class-Dependent Assignment in Cluster-based Servers

Victoria Ungureanu[*]
Department of MSIS
Rutgers University
180 University Ave.
Newark, NJ 07102,
USA

Benjamin Melamed[†]
Department of MSIS
Rutgers University
94 Rockafeller Rd.
Piscataway, NJ
08854,USA

Phillip G. Bradford[‡]
Department of CS
The University of
Alabama
Box 870290
Tuscaloosa, AL
35487, USA

Michael Katehakis[§]
Department of MSIS
Rutgers University
180 University Ave.
Newark, NJ 07102,
USA

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 in Round-Robin manner 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 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.

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"

[*] Work supported in part by DIMACS under contract STC-91-19999, and by Information Technology and Electronic Commerce Clinic, Rutgers University. email:ungurean@rbs.rutgers.edu

[†] email:melamed@rbs.rutgers.edu

[‡] email:pgb@cs.ua.edu

[§] email:mnk@andromedarutgers.edu

SAC'04 March 14-17, 2004, Nicosia, Cyprus

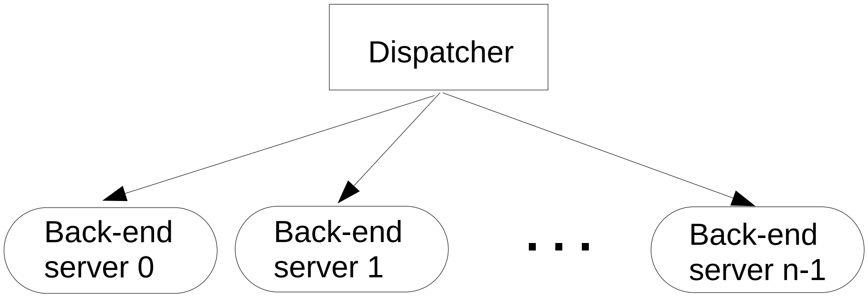


Figure 1: A cluster-based server

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 [10, 23]. However, the former are rarely, if at all, used in practice [15, 23] 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 [7, 2, 8] 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 [2] 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 workload. 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. Section 3 discusses the `CDA` policy and Section 4 provides a detailed performance analysis for it. Finally, Section 5 concludes the paper.

2. PREVIOUS WORK

The literature contains a considerable body of work on job scheduling (see [3, 4, 6, 14, 17, 18, 20] and references therein). Most of the analytical models assume that arrivals and service times follow exponential-type distributions. Smith [22] considers 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 [21] shows that this algorithm yields minimal expected response times for job sizes having arbitrary (known) distributions. Winston [25] 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 [24] 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, which consider job sizes that follow a power-law distribution. Harchol-Balter *et al.* [10] 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 [10]. Ciardo *et al.* [5] 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 [19] 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. THE CDA POLICY

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 job with the shortest estimated size in its buffer. While the back-end server processes a long request, it will not be assigned any other jobs.*

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. Furthermore, it has been shown that the time to service a request is well-approximated by the size of the requested file [11].

Secondly, the dispatcher can estimate job size only for *static* requests (dynamic files are created on the fly by the server in response to a request). Consequently, CDA implicitly treats dynamic requests as short. Even though this classification may be inaccurate for certain dynamic requests, the errors incurred do not affect substantially CDA performance. This is because evidence suggests that while the number of dynamic requests is growing, the majority of the requests at most web servers are static [1, 11, 13]. For example, it was shown in [2] that for World Cup traces only 0.02% of requests were for dynamic files.

Thirdly, to prevent *starvation* of large jobs, the dispatcher periodically updates estimated sizes of jobs in its queue. In essence, the estimated size of a job decreases with the time it has waited in the dispatcher queue. Consequently, even large jobs will eventually appear to the dispatcher as having small sizes, and thus accelerate their selection for assignment. This feature is implemented as follows: the dispatcher records for each job the time it has last updated its size. If the time since the last update exceeds a predefined interval $\mathtt{T_u}$, then the estimated size is divided by a predefined factor $\mathtt{F_u}$. (A long job is assigned to a back-end server either when it becomes the shortest long job in the buffer, or when its estimated size falls below c. In either case, while the back-end server processes the request no other jobs will be assigned to it.) The values for $\mathtt{T_u}$ and $\mathtt{F_u}$ used in the experimental study (presented in Section 4.2) are respectively, 100 ms and 2. (In effect, by using these values, the estimated size of a file is reduced by a factor of 1000 in 1s, and thus even very large files are not delayed too long.)

Finally, CDA requires the dispatcher to know when back-end servers become idle. The dispatcher can infer this type of information by monitoring its number of active connections to back-end servers. (The dispatcher is responsible for passing incoming data pertaining to a job from a client to a back-end server. So for each job in progress at a back-end server there is an open connection between the dispatcher and that server [15, 23].)

4. EXPERIMENTAL PERFORMANCE STUDY

We demonstrate the good performance of CDA by running a simulation driven by trace data from Internet sites serving the 1998 World-Cup. The data used was archived in an Internet repository (see [2] and http://ita.ee.lbl.gov/html/traces.html).

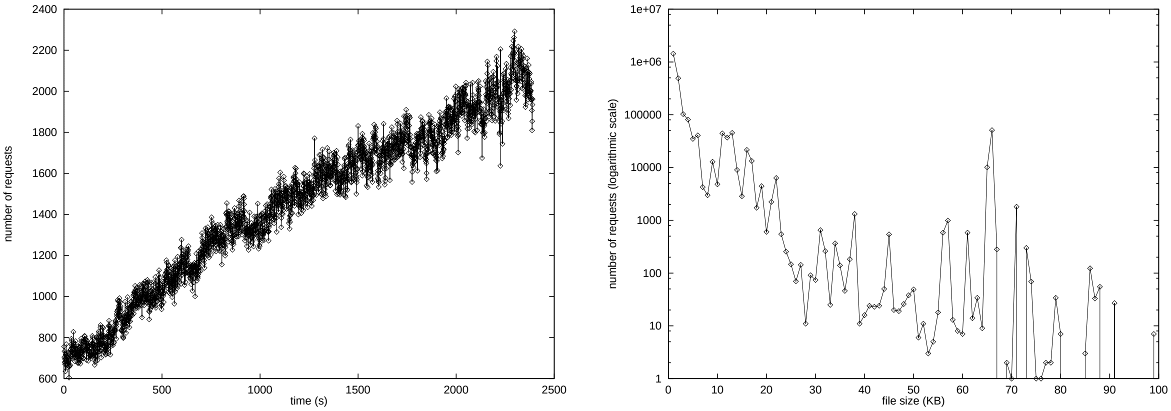
4.1 Simulation Data

The aforementioned 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 [2] 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 2(a) 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(b), 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.
- Files with sizes greater than 30KB, which make up less than 3% of the files requested, account for over 50% of the overall workload. 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



(a) Number of request arrivals per second;

(b)File size distribution of arriving requests (notice the logarithmic scale on the y axis)

Figure 2: Empirical request statistics from a World Cup trace.

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 [15, 23], and are as follows: the constant time to make a connection is set at 0.9 ms, and the transmission and processing time is set at 80μs/KB.

4.2 Simulation Experiments

The simulation study models a cluster with four MT (multi-threaded) back-end servers. In an MT back-end server, a thread performs all steps associated with a request before accepting a new one. The simulation uses non-preemptive (coroutine) scheduling for threads, where a thread is allowed to run until it finishes or blocks. The study makes the following standard, simplifying assumptions: (1) communication times between the dispatcher and back-end servers are negligible, and (2) the overhead incurred by the dispatcher to select (job, server) pairs is negligible.

The policies compared are `CDA`, `Round-Robin` and `JSQ`. We excluded from the study `Size-Range` policies, because it was shown in [19] 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 3 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`, whose average waiting time is 826μs. 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. 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, and the status of back-end servers in a limited number of cases. Furthermore, `CDA` enjoys the flexibility of trading performance for information through a choice of the cutoff point value, `c`.

Figure 4 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 modest informational requirements: 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. We conclude by pointing out that the results above ignore the communication overhead incurred by `JSQ` in order to determine the status of back-end servers. Consequently, we expect the actual performance of `CDA` relative to `JSQ` to be even better.

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. The experimental study shows 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,

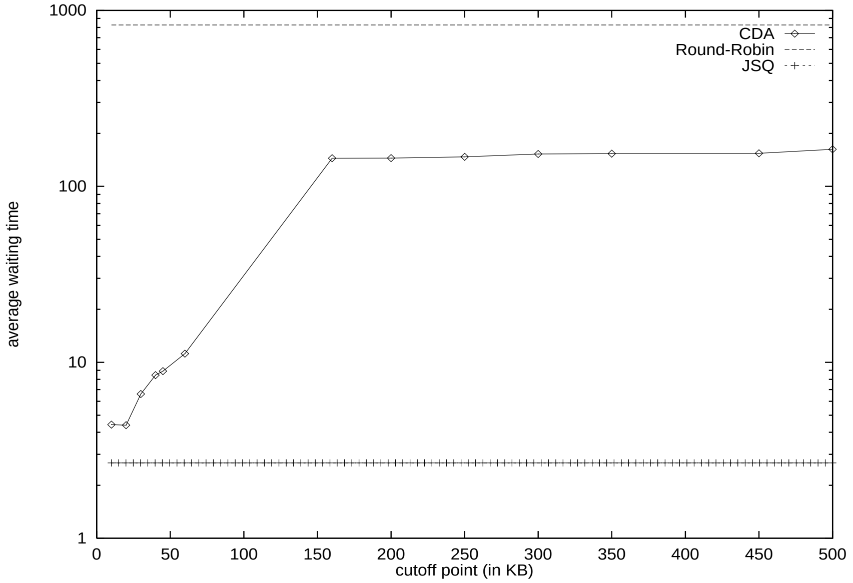


Figure 3: Average waiting time of `Round-Robin`, `JSQ` and `CDA` as function of cutoff point (the y axis is in logarithmic scale).

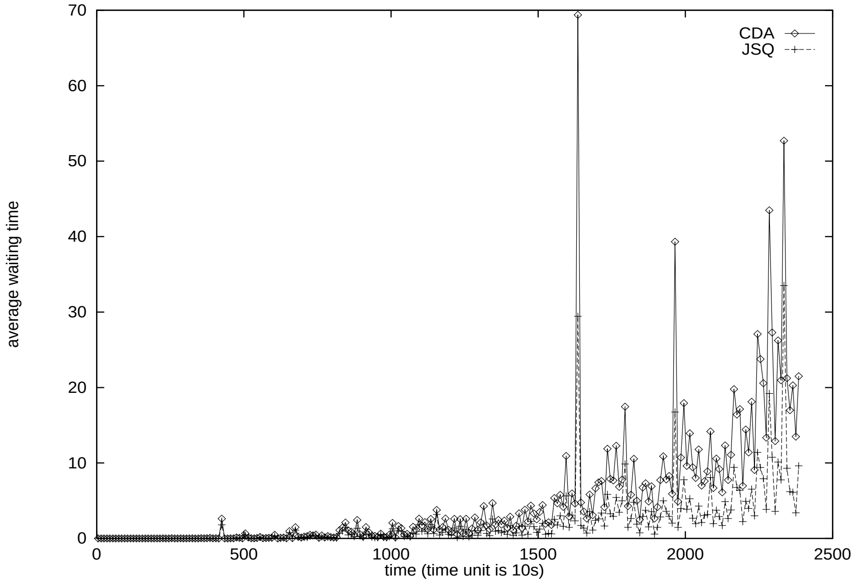


Figure 4: Average waiting times of `CDA(30)` and `JSQ` over successive 10-second intervals.

and incurs less communication overhead. All things considered, it is possible that when communication overhead is taken into account, `CDA` would perform better than `JSQ`.

6. REFERENCES

[1] Arlitt, M., Friedrich, R. and Jin, T. Workload Characterization of a Web proxy in a cable modem environment. *ACM Performance Evaluation Review*, 27(2):25-36, 1999.

[2] Arlitt, M. and Jin, T. Workload characterization of the 1998 World Cup Web Site. *IEEE Network*, 14(3):30-37, May/June 2000.

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

[4] Cardellini, V., Casalicchio, E., Colajanni, M. and Yu, P. S.. The state of the art in locally distributed Web-server systems. *ACM Computing Surveys*, 34(2):263-311, June 2002.

[5] Ciardo, G., Riska, A. and Smirni, E. EquiLoad: a load balancing policy for clustered Web servers. In *Performance Evaluation* 46(2-3): 101-124, 2001.

[6] Colajanni, M., Yu, P. S. and Dias, D. M. Analysis of task assignment policies in scalable distributed Web-server systems. *IEEE Transactions on Parallel and Distributed Systems*, 9(6), 1998.

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

[8] Faloutsos, M., Faloutsos, P. and Faloutsos, C. On power-law relationships of the Internet topology. In Proceedings of *ACM SIGCOMM '99*, 251-262, Aug. 1999.

[9] Harchol-Balter, M. Task assignment with unknown duration. *Journal of the ACM*, 49(2):260-288, March 2002.

[10] Harchol-Balter M., Crovella, M.E. and Murta, C.D. On choosing a talk assignment policy for a distributed server system. In *Proceedings of Performance Tools '98*, Lecture Notes in Computer Science, 1468:231-242, 1998.

[11] Harchol-Balter M., Schroeder, B., Bansal, N. and Agrawal, M. Size-based scheduling to improve Web performance. *ACM Transactions on Computer Systems*, 21(2), May 2003.

[12] Hu, Y., Nanda, A. and Yang, Q. Measurement, analysis and performance improvement of the Apache Web server. *The International Journal of Computers and their Applications*, 8(4), 217-231, 2001.

[13] Krishnamurthy, B. and Rexford, J. Web Protocols and Practice : HTTP 1.1, Networking Protocols, Caching, and Traffic Measurement. Addison-Wesley, 2001.

[14] Katehakis, M. and Melolidakis, C. On the optimal maintenance of systems and control of arrivals in queues. *Stochastic Analysis and Applications*, 8(2):12-25, 1994.

[15] Pai, V.S., Aron, M., Banga, G., Svendsen, M., Druschel, P. Zwaenepoel, W. and Nahum, E. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, 1998.

[16] Pai, V.S., Druschel, P. and Zwaenepoel, W. Flash: an efficient and portable Web server, In *Proceedings of the USENIX 1999 Annual Technical Conference*, 1999.

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

[18] Righter, R. Scheduling in multiclass networks with deterministic service times. *Queuing Systems* 41(4): 305-319, 2002.

[19] Riska, A., Sun, W., Smirni, E. and Ciardo, G. AdaptLoad: effective balancing in clustered Web servers under transient load conditions. In *22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

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

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

[22] Smith, W.E. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59-66, 1956.

[23] Teo, Y.M. and Ayani, R. Comparison of load balancing strategies on cluster-based Web servers. *Simulation, The Journal of the Society for Modeling and Simulation International*, 77(5-6):185-195, November-December 2001.

[24] Whitt, W. Deciding which queue to join: some counter examples. *Operations Research*, 34(1):55-62, 1986.

[25] Winston, W. Optimality of the shortest line discipline. *Journal of Applied Probability*, 14:181–189, 1977.