

# Performance Comparison of Size-based Scheduling Policies in Clustered Web Servers

Victoria Ungureanu \*    Benjamin Melamed †    Michael Katehakis ‡

## Abstract

Web servers commonly use cluster-based architectures which combine good performance and low cost. 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, which in turn serve the requests according to some discipline.

It has been proven that giving priority to short job yields optimal performance for single (unclustered) servers. In this paper we are comparing the performance of two policies that apply this strategy to clustered Web servers. The first policy, called **CDA** (Class Dependent Assignment) calls for the dispatcher to assign 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. The second policy, called **SRPT** (Shortest Remaining Processing Time), calls for a back-end server to schedule short jobs (or with short remaining processing time) with higher priority than long jobs. In effect, **CDA** uses the *global* information available to the dispatcher, while **SRPT** uses the *local* information available to back-end servers. Therefore, an experimental comparison between these two policies would contrast the performance metrics achieved when using local vs. global information, and it would quantify the performance differences.

To gauge the performance of these policies, we exercised them on empirical data traces measured at Internet sites. The experimental results show that **CDA** outperforms **SRPT** by 40% with respect to the metrics of average waiting time and slow-down. These results confirm that using global information yields better performance than using local information and quantify the performance improvement.

**Keywords:** clustered Web server, scheduling policy, power-law distribution, simulation.

---

\*Department of MSIS, Rutgers University, 180 University Ave., Newark, NJ 07102, email: un-  
gurean@rbs.rutgers.edu

†Department of MSIS, Rutgers University, 94 Rockefeller Rd., Piscataway, NJ 08854, email:  
melamed@rbs.rutgers.edu

‡Department of MSIS, Rutgers University, 180 University Ave., Newark, NJ 07102, email:  
mnk@andromeda.rutgers.edu

# 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, key Quality of Service (QoS) performance metrics are waiting time and job slow-down. To improve performance 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. We have presented in [10] a policy for this type of architecture,

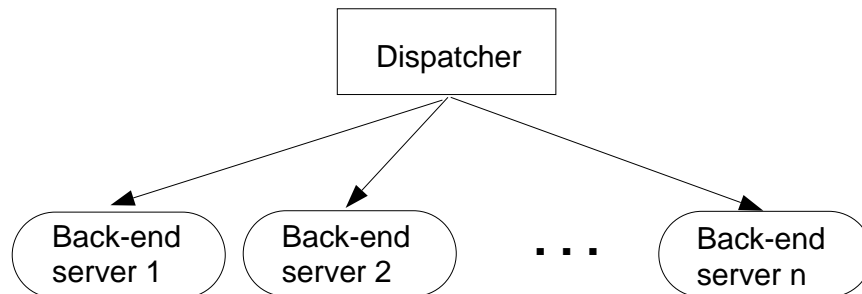


Figure 1: A clustered Web server.

called CDA (Class Dependent Assignment). 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. In broad outline, CDA operates as follows: 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.

The rationale for this policy is that giving priority to short job yields good performance. Indeed, it has been shown that such policies are optimal for single (unclustered) servers. In particular, Smith [8] shows that scheduling the shortest jobs first is optimal (i.e. yields minimal response times) when the jobs have (deterministic) fixed-sizes. In a similar vein, Rothkopf [7] shows that scheduling shortest job first yields minimal expected response times, for job sizes having arbitrary (known) distributions. The size-dependent behavior of CDA is further motivated by empirical evidence suggesting that the sizes of files traveling on

the Internet follow power-law distributions [2, 1, 3]. In power-law job-size distributions, *a relatively small fraction of jobs accounts for a relatively large fraction of the overall load*. Thus, CDA is based on the premise that the judicious assignment of (the relatively few) long jobs will lead to good performance.

In [10] we have compared the performance of CDA with Round-Robin and JSQ (Join Shortest Queue)<sup>1</sup>. 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. The latter is a significant result, since it has been proven that JSQ has optimal response-time statistics when the dispatcher uses immediate assignment, and job arrival and sizes follow certain distributions [12]. (Whitt [11] showed that there exist other job-size distributions for which JSQ is not optimal.)

In this paper we compare CDA performance with the SRPT (Shortest Remaining Processing Time) policy proposed recently by Harchol-Balter *et al.* [4]. Under SRPT, the back-end server schedules first short jobs (or with short remaining processing time). It has been experimentally shown in [4] that SRPT greatly outperforms traditional policies.

Both CDA and SRPT give priority to short jobs in order to improve response time performance. However, CDA uses the *global* information available to the dispatcher, while the SRPT uses the *local* information available to back-end servers. More specifically, under CDA, the dispatcher chooses jobs from the global pool of jobs, whereas under SRPT each back-end server chooses from the jobs assigned to it. We are interested in an experimental comparison between these two policies because it contrasts the performance metrics achieved when using local vs. global information, and it quantifies the performance differences.

To gauge the performance of CDA and SRPT, we exercised them 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. 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 shows 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 4% of the jobs are classified as long, then CDA outperforms SRPT by 40% with respect to both average waiting time and slow-down.

These results further show that using the global information (available to the dispatcher) vs. the local information (available to back-end servers) yields markedly improved performance. The results are all the more remarkable given that CDA uses less information than SRPT in the sense that CDA requires the dispatcher to know only the sizes of large jobs (4% of the total number of requests), whereas SRPT requires a back-end server to know the sizes of all jobs.

The rest of the paper is organized as follows. Section 2 discusses in more detail the CDA policy. Section 3 provides a detailed performance analysis for CDA and SRPT. Finally, Section 4 concludes the paper.

---

<sup>1</sup>JSQ assigns each incoming job to the back-end server with the smallest amount of residual work.

## 2 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, to prevent *starvation* of large jobs, the dispatcher periodically updates the 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.

Secondly, 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 [5, 9].)

## 3 Performance Study

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, (2) the overhead incurred by the dispatcher to select (job, server) pairs is negligible, and (3) the overhead incurred by a back-end server to select a job for execution is negligible.

The study runs a simulation driven by trace data from Internet sites serving the 1998 World-Cup. The data used was archived in an Internet repository (see [1] and <http://ita.ee.lbl.gov/html/traces.html>).

### 3.1 The Workload

The aforementioned repository provides detailed information about the 1.3 billion requests received by the 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 these traces follow a power-law distribution.

From this repository, we selected a trace covering the first 600 minutes of June 26th and containing over 11 million requests. Figure 2 depicts the number of requests received by the server in minute intervals. The long tail exhibited by the data can be discerned in Figure 3, 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. Since no information was recorded regarding the service time, our simulation experiments posited a service time proportional to the size of the requested document. We point out that this is an assumption whose reasonableness has been argued elsewhere [6, 4].

### 3.2 The Simulation

When simulating CDA we chose the cutoff-point between short and long jobs to be 20K. This cutoff value was chosen, because it leaves less than 4% of the jobs designated as “long”. When simulating SRPT we assumed that the dispatcher assigns jobs immediately on arrival to back-end servers in Round-Robin manner, and each server processes the shortest job in its queue first.

In order to evaluate the relative efficacy of CDA and SRPT, we compared their performance with respect to the following statistics:

1. Average waiting time (excluding service time);
2. Average slow-down (the ratio of a job waiting time to its service time – file size; in our case)

Figure 4 displays the average slow-down for the CDA and SRPT policies, over successive 5-minute intervals. The figure shows that the CDA policy yields a substantial lower slow-down than SRPT in all time intervals considered.

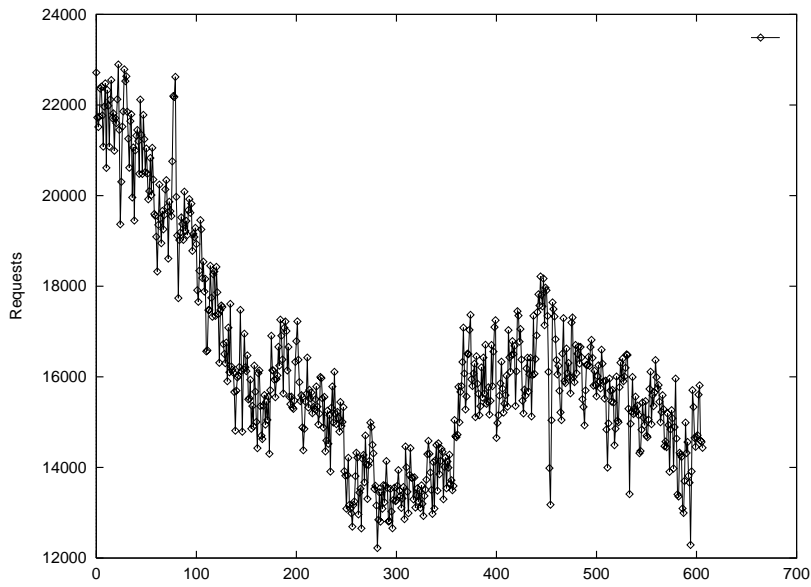


Figure 2: Number of request arrivals per minute.

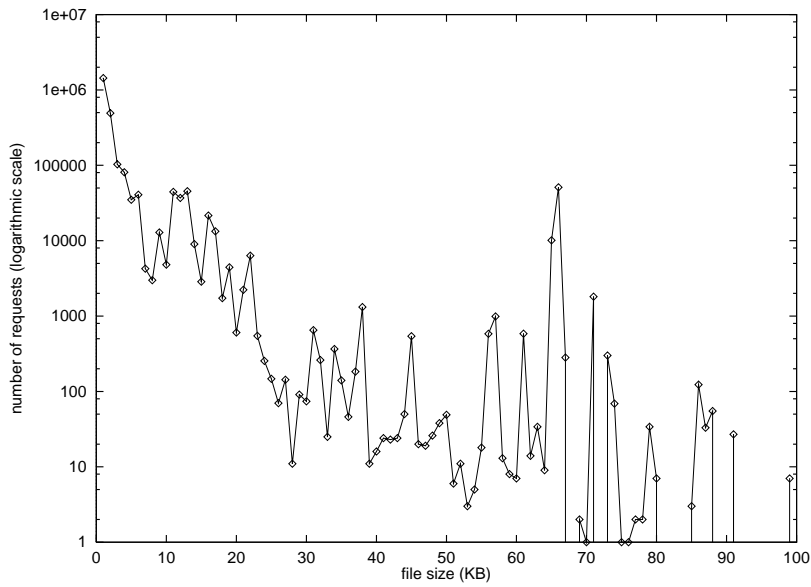


Figure 3: File size distribution of arriving requests (the y axis is in logarithmic scale).

Table 1 displays the performance of CDA and SRPT for the overall simulation horizon. The results show again that the CDA policy performs substantially better than SRPT. More specifically, both the average waiting time and slow-down resulting from CDA are approximately 40% lower than those resulting from SRPT.

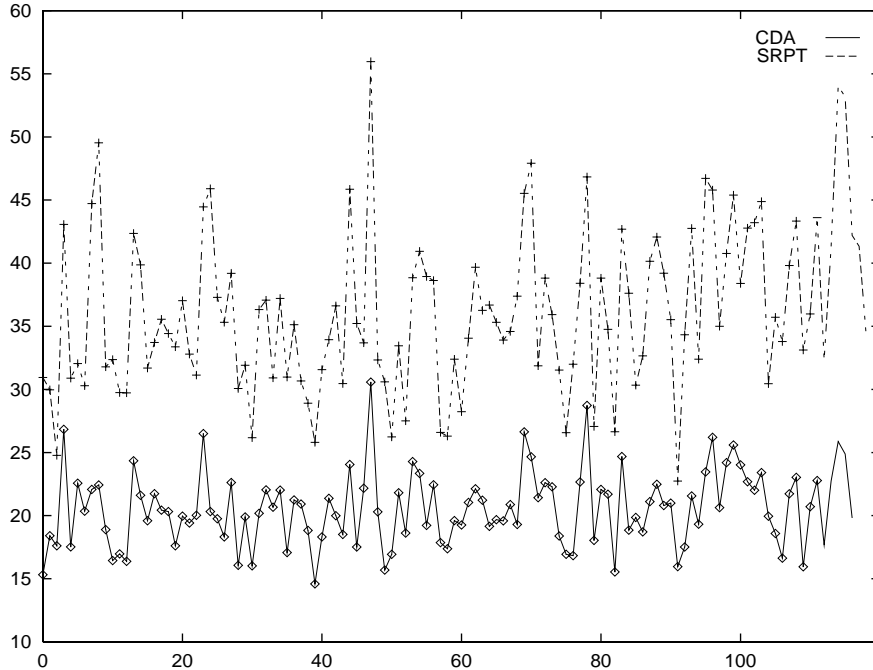


Figure 4: Average slow-down of CDA and SRPT as function of time (time unit is 5 min.)

Policy	Average Waiting Time (ms)	Average Slow-down
SRPT	57	36
CDA	35	22

Table 1: Comparison of statistics for the SRPT and CDA policies.

## 4 Conclusion

In this paper we have compared the performance of two size-based policies: CDA and SRPT. The former calls for the dispatcher to assign short jobs in Round-Robin manner as soon as they arrive, while long jobs are assigned to back-end servers only when the servers become idle. The latter calls for a back-end server to schedule short jobs with higher priority than long jobs. We have experimentally shown that CDA outperforms SRPT by 40% when run on a Web clustered server using a multi-threaded architecture. These results further show that using the global information (available to the dispatcher) vs. the local information (available to back-end servers) yields markedly improved performance.

These results were obtained by modeling a cluster using MT (multi-threaded) back-end servers. Another common architecture for back-end servers is MP (multi-processor). In MP-architecture a back-end server employs multiple processes. A process serves a request to completion before accepting a new one. Each process runs for a predefined time interval (quantum) or until it blocks, after which another process is selected to run. Simulating CDA

and SRPT on a clustered server using the MP architecture is part of our plans for future work.

## References

- [1] Arlitt, M. and Jin, T. Workload characterization of the 1998 World Cup Web Site. *IEEE Network*, 14(3):30-37, May/June 2000.
- [2] 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.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] Rothkopf, M.H. Scheduling with random service times. *Management Science*, 12;703-713, 1966.
- [8] Smith, W.E. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59-66, 1956.
- [9] 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.
- [10] Ungureanu, V., Melamed, B., Bradford, P. and Katehakis, M. “Class-Dependent Assignment in Cluster-based Servers”, In *19th ACM Symposium on Applied Computing: Special Track on Parallel and Distributed Systems*, March 2004.
- [11] Whitt, W. “Deciding which Queue to Join: Some Counter Examples,” *Operations Research*, Vol. 34, No. 1, 55-62, 1986.
- [12] Winston, W. “Optimality of the Shortest Line Discipline,” *Journal of Applied Probability*, 14:181–189, 1977.