LUND UNIVERSITY

On overload control through queue length for web servers

Jianhua, Cao; Nyberg, Christian

Published in:
Sixteenth Nordic Teletraffic Seminar NTS 16 : Helsinki University of Technology, Espoo August 21-23, 2002 : proceedings (Report / Helsinki University of Technology, Networking Laboratory)

2002

Link to publication

Citation for published version (APA):
Jianhua, C., & Nyberg, C. (2002). On overload control through queue length for web servers. In P. Lassila, E. Nyberg, & J. Virtamo (Eds.), *Sixteenth Nordic Teletraffic Seminar NTS 16 : Helsinki University of Technology, Espoo August 21-23, 2002 : proceedings (Report / Helsinki University of Technology, Networking Laboratory)* Helsinki University of Technology, Networking Laboratory.

Total number of authors:
2

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

This is an author produced version of a paper presented at
Sixteenth Nordic Teletraffic Seminar NTS 16 : Helsinki University of
Technology, Espoo August 21-23, 2002.
This paper may not include the final publisher
proof-corrections or pagination.

Citation for the published paper:
Cao, Jianhua, Nyberg, Christian, 2002,
"On overload control through queue length for web servers",
Sixteenth Nordic Teletraffic Seminar NTS 16 : Helsinki University of Technology, Espoo August 21-23, 2002 : proceedings (Report / Helsinki University of Technology, Networking Laboratory, ISSN 1458-0322).
Publisher: Helsinki University of Technology, Networking Laboratory.

On Overload Control through Queue Length for Web Servers

Jianhua Cao and Christian Nyberg

Department of Communication Systems
Lund Institute of Technology, Sweden
e-mail: {jcao, cn}@telecom.lth.se

Abstract

We investigate the performance of overload control through queue length for two different web server architectures. Both of them use finite queue lengths to prevent servers from being overloaded. One architecture prioritizes requests from established sessions while the other treats all requests equally. First, we introduce queueing models for these two systems. Then, we define and explain a new web server performance metric that is a function of session throughput, error rate for connection within sessions and average request response time. Finally, we use simulation to evaluate the performance of these two types of web servers. The result suggests that the benefit of request prioritization is noticeable only when the capacities of the sub-systems match each other.

1 Introduction

The excessive delay of web services is more and more common as the number of Internet users increases everyday. Not only the customers will be unsatisfied but also the service provider will be hurt in the long term. There are two ways to alleviate the problem, namely, infrastructure upgrading and overload control. Infrastructure upgrading is an obvious approach and will bring more traffic and, probably, more profit to the operator. Overload control, however, assures a certain quality of service to a limited amount of customers and sacrifices the rest by throttling the incoming traffic.

Overload control may not be the best thing to do when infrastructure upgrading is possible, but it is still a necessary complement to upgrading approach for two reasons. First, one can not upgrade the system so often as to keep up with the ever increasing Internet traffic. Second, the large variance of Internet traffic will cause the web server overload from time to

time even if the web server is engineered to handle the average traffic. So how could overload control be done? A crude overload control mechanism will constantly monitor the server's CPU usage, connection response time, the number of current connections (jobs) and/or the number of queued requests. When certain parameters exceed some predetermined levels, the server starts to reject customers until monitored parameters are back to normal. However, this easy scheme has a small problem.

A customer visiting a web site tends to send several requests in sequel. A possible sequence could contain the following commands: browsing, searching, ordering, paying and exiting. We call such a sequence of requests a session. As one can see here, the customer will be very irritated if her paying request is rejected after filling in her credit card number. In general, requests within a session should not be rejected. When a overload control scheme is blind at sessions, it is called request-based overload control (RBOC) otherwise session-based overload control (SBOC).

Different overload control strategies for telephone switches have been studied by Nyberg [5]. The performance of SBOC and RBOC for e-commerce web sites has been studied by Kihl and Widell [2] recently. Several attempts [1, 4, 6] have been tried to model the web servers and have different level of success.

In this paper, we use simulation to study the effectiveness of SBOC and RBOC through queue length. Overload control through queue length means that the control decision is based on the number of queued requests. The result shows that the SBOC through queue length is not necessarily effective when the web server is not carefully configured.

The paper is organized as follows. In Section 2, we introduce the queuing models of a web server. We then define and explain a performance metric for web servers in Section 3. Section 4 gives the simulation results of web servers using SBOC and RBOC with three different configurations. We conclude the paper in Section 4.

2 Queuing models of web servers

The basic queuing model used here is based on [4]. The model consists of three subsystems, TCP, HTTP and IO, in tandem. Fig. 1 shows the structure.

The TCP subsystem is modeled as a multi-server system with zero buffer. The number of TCP servers, m_{tcp} is equal to the maximum number of allowed concurrent TCP connections. The HTTP subsystem is modeled as a multi-server system with a finite buffer. The number of HTTP servers, m_{http}, is equal to the maximum number of allowed concurrent HTTP processes or threads. The IO subsystem is modeled as a processor sharing server with a finite buffer. The buffer size of HTTP subsystem, n_{http}, and the buffer size of the IO subsystem, m_{io}, along with m_{tcp} and m_{http}, are all configurable parameters.

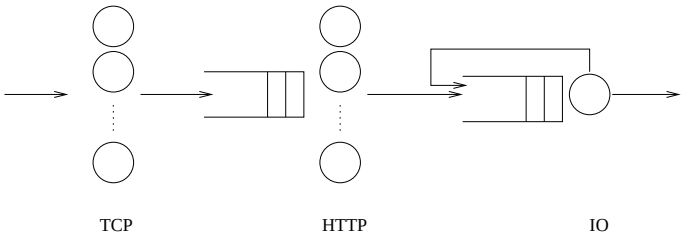


Figure 1: A basic queueing model of web server consists of three subsystems,TCP, HTTP and IO in tandem.

The model works as follows. When a HTTP request arrives, the TCP subsystem will establish a connection between the client and the web server by handshake. So the service time of the TCP subsystem, x_{tcp}, is equal to the the round trip time, t_{rtt}. Note that the TCP subsystem will allow at most m_{tcp} simultaneous connections .

After the connection is established, the request is forwarded to the HTTP subsystem and the TCP server will be released. The request will be processed immediately if there is a free HTTP server available otherwise it will be pushed into the FIFO queue. The service time of the HTTP subsystem, x_{http}, is the total time spending on parsing the HTTP requests and fetching/generating the HTML files. We could reasonably assume that x_{http} is proportional to the returned file size with coefficient, k(with unit s/KB). As soon as the returned files are compiled, the job will be transfered to the IO subsystem when there is an IO slot available, otherwise the HTTP server that processes the request will be hold. The HTTP server on holding state can not process more HTTP requests until it is freed.

The IO subsystem uses the TCP protocol to send the HTML files. The bandwidth of IO server is shared by all jobs in the queue. The IO server polls jobs in a round robin fashion. The service time of an IO job, x_{io}(with unit seconds), depends on the file size, s_{file}(with unit KB), the bandwidth of the server, w_{server}(with unit KB/s), and the client, w_{client}, the number of concurrent IO jobs, $n_{\text{io}}(\leq m_{\text{io}})$, and the packet loss rate of the connection, p_{loss}.

To simplify the derivation of x_{io}, we use the following four assumptions. First, all the clients use the same maximum segment size, s_{mss} (with unit KB). Second, all the clients have the same packet loss rate of zero, $p_{\text{loss}} = 0$. Third, all the connection have the same TCP flow/congestion control window size, s_{wnd}(with unit KB), all the time. Fourth, the bandwidth of client is less than that of server. So, the approximated IO service time is given by Eq.1.

$$x_{\text{io}} = \left\lceil \frac{s_{\text{file}}}{s_{\text{wnd}}} \right\rceil t_{\text{rtt}} + \frac{s_{\text{file}}}{w_{\text{client}}} + \left\lceil \frac{s_{\text{file}}}{s_{\text{mss}}} \right\rceil \frac{s_{\text{mss}}}{w_{\text{server}}}(n_{\text{io}} - 1) \tag{1}$$

The first item in the right hand side of the equation is the transmission

time of the file; the second item is the time for client to receive the file; the third item is the waiting time.

The model described above can be used to predict the throughput, the response time of the web server quite accurately. But it needs some modification when the requests are grouped in sessions, otherwise the arrival process is not a Poisson process any more.

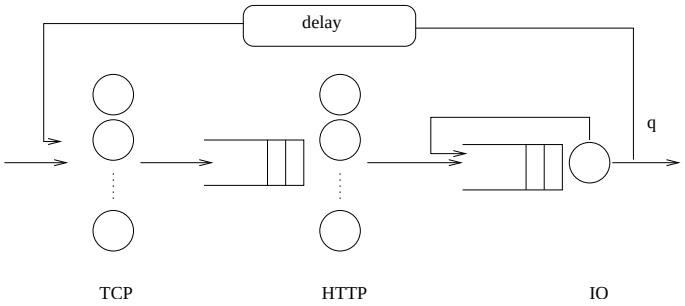


Figure 2: The session is modeled by Bernoulli feedback. If the feedback probability is q, then average number requests in a session is $1/(1-q)$.

In our case, we model the sessions by adding a Bernoulli feedback from the output to the input of the model above, as shown in Fig. 2. So it implies that the number of requests in a session is geometrically distributed. This fact is justified by the statistical analysis of [3]. Since the transient analysis is not our major interests, the delay between requests is not modeled explicitly here.

The HTTP server with FIFO queue of finite length is in fact an implementation of RBOC through queue length. Therefore the model above can be used to investigate the performance of RBOC. The HTTP subsystem of SBOC is a bit different from that of RBOC, as illustrated in Fig. 3. Let us call the customers from the feedback the old customers and the others new customers. The SBOC uses two separate queues in the HTTP subsystem: one for new customers and one for old customers. The queue for old customers is assigned with a higher priority in order to break as little number of sessions as possible.

3 Web Server Performance Metrics

Three metrics, throughput, H, response time, T, and connection error rate,E are widely used to evaluate the performance of web servers. Throughput is defined as the average number of completed requests (or sessions) per second. Typically, the throughput will reach some limit when the arrival intensity, λ, i.e. the number of arrived customers per second, exceeds some threshold, λ^*. Below that threshold, the throughput will increase linearly with λ. The

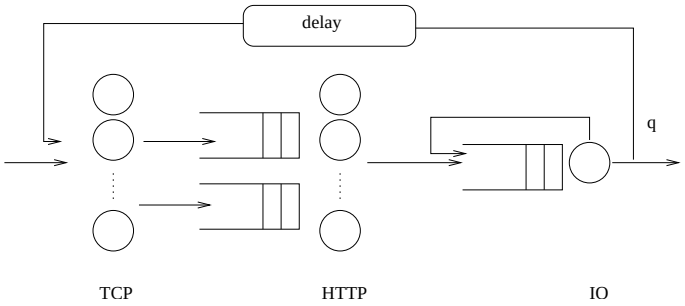


Figure 3: The model of web server using SBOC scheme. Two queues are used for old customers and new customers. The queue for old customers has higher priority than the other one.

response time measures the interval between when request is sent and when the requested files are received by the customer. It should be a small constant when $\lambda < \lambda^*$ and increase when λ is close to λ^*. If the web server uses finite buffers, the response time will be a large constant when $\lambda > \lambda^*$. The connection error rate reflects the percentage of requests (or sessions) that are rejected (or broken) due to the limited capacity of the web server. It should be approximately zero when $\lambda < \lambda^*$, and approach one when λ goes to infinite.

In order to compare overall performance of RBOC and SBOC, we need a unified metric. In analogy with the definition of network power which is throughput divided by delay, we propose a metric called the power of web server, denoted by P. It is a function of session throughput, error rate for sessions and average request response time:

$$P = \frac{H_{\text{session}}}{T_{\text{request}}}(1 - E_{\text{session}}) \tag{2}$$

where H_{session} is the number of completed sessions per second, T_{request} is the average response time for each accepted request, E_{session} is the probability that a session is being broken because one of its requests is rejected by the server.

Since H_{session}, T_{request} and E_{session} are all functions of the arrival intensity λ, and so is the server power, P. We would expect the power of an ideal server to have the following properties: First, it should be proportional to λ when $\lambda < \lambda^*$. Second, it should be able to maintain the maximum power, $P(\lambda^*)$, when $\lambda > \lambda^*$. As we can see later, the RBOC cannot maintain the second property and the power of web server using RBOC will drop. So can SBOC maintain maximum power constantly even if the server is heavily loaded? Our answer is "it depends" as the simulation shows.

4 Simulation Results

We use discrete event simulation to investigate the performance of web servers using RBOC and SBOC.

To avoid unnecessary complication, we limit the sources of randomness in our simulation to be two: the customer arrival process assumed to be Poissonian and the session length which is geometrically distributed with mean $1/(1-q) = 6.0$. The following parameters are assumed to be constants: the size of file that clients request, $s_{\text{file}} = 10\text{KB}$; the speed of file feteching, $k = 0.01$ s/KB; the client bandwidth, $w_{\text{client}} = 5.5\text{KB/s}$; the round trip time between client and server, $t_{\text{rtt}} = 0.1\text{s}$; the maximum segment size, $s_{\text{mss}} = 0.5\text{KB}$.

The following parameters in both models are configurable: the number of TCP servers, m_{tcp}; the number of HTTP servers, m_{http}; the maximum queue length of HTTP subsystem, n_{http}; the maximum buffer size of IO subsystem, m_{io}; and the bandwidth of the web server, w_{server}.

Let C denote the capacity of a system, i.e. the maximum number of requests that can be proceeded in one second. The capacities of TCP, HTTP, IO subsystems are then given by:

$$C_{\text{tcp}} = \frac{m_{\text{tcp}}}{x_{\text{tcp}}} = \frac{m_{\text{tcp}}}{t_{\text{rtt}}} \tag{3}$$

$$C_{\text{http}} = \frac{m_{\text{http}}}{x_{\text{http}}} = \frac{m_{\text{http}}}{k \cdot s_{\text{file}}} \tag{4}$$

$$C_{\text{io}} = \frac{m_{\text{io}}}{x_{\text{io}}|_{n_{\text{io}}=m_{\text{io}}}} \tag{5}$$

Given that k, s_{file} and t_{rtt} are constants, the capacities of the sub-systems, C_{tcp}, C_{http}, C_{io}, are then fully determined by the system configuration.

In most configurations, it will usually be the case that $C_{\text{tcp}} > C_{\text{http}}$ and $C_{\text{tcp}} > C_{\text{io}}$. So we limit our investigation to the following three cases: $C_{\text{http}} > C_{\text{io}}$ (case A), $C_{\text{http}} = C_{\text{io}}$(case B), $C_{\text{http}} < C_{\text{io}}$ (case C).

Further we fix the configurations: $m_{\text{tcp}} = 102$, $n_{\text{http}} = 100$, $m_{\text{io}} = 507$ and $w_{\text{server}} = 100\text{Mbits/s}$, to make C_{io} a constant. The variation of C_{http} is then achieved by adjusting the number of HTTP servers, m_{http}.

We show the different configurations of m_{http} and corresponding subsystem capacities in three cases in Table 1. The simulation results are shown in Fig. 4 and Fig. 5.

In Fig. 4, we plot the session throughput, average request response time and session error rate for systems using RBOC and SBOC in three different cases. The sub-figure (a) and (b) show that RBOC and SBOC give almost identical session throughput. But SBOC gives lower response than RBOC in all three cases as sub-figures (c) and (d) indicate. The sub-figure (e) gives expected behavior of session error rate of RBOC that approaching to one when arrival intensity exceeds some threshold. However the session error rate of SBOC in case A shows the the same pattern. Recall that in case

	case A	case B	case C
m_{http}	24	12	6
C_{tcp}	10240	10240	10240
C_{http}	240	120	60
C_{io}	120.057	120.057	120.057

Table 1: Different configuration of m_{http} and the capacities of subsystems in different cases. case A: $C_{\text{http}} > C_{\text{io}}$; case B: $C_{\text{http}} = C_{\text{io}}$; case C: $C_{\text{http}} < C_{\text{io}}$.

A, the capacity of HTTP subsystem is greater than that of IO subsystem, therefore HTTP servers tend to be held when a job completes. When there are some HTTP servers are held, the effective number of servers will decrease and queues of HTTP subsystem will tend to pile up.

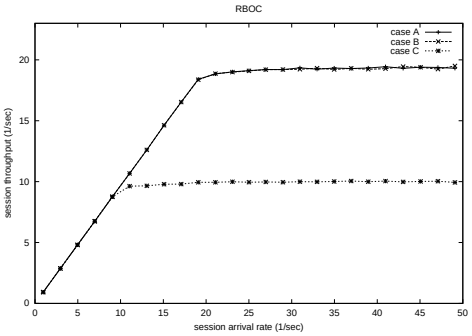
In Fig. 4, we show the power of systems using RBOC and SBOC. Now it is much more straightforward to compare RBOC and SBOC in three different configurations. Let us first consider the case B. It is clear from the simulation results that the power of the system using SBOC does not drop even when the arrival intensity exceeds the threshold λ^*, while it is not the case for the system using RBOC. However in case A, the power of two systems using RBOC and SBOC is the same. The reason is mainly due to that in both systems the session error rate increases as the arrival intensity increases. In case C, we see some advantages of SBOC over RBOC but the gain is not that much as in case B. We also notice that the power of the two systems in case C is the half of that in case B. If checking the Table 1, we will find out that the number of HTTP servers in case C is half of that in case B. Obviously the configuration of case C makes the web server under utilized.

So finally, we can reach a conclusion that SBOC will have its largest advantage over RBOC when the web server is configured in such a way that the capacities of subsystems match each other.

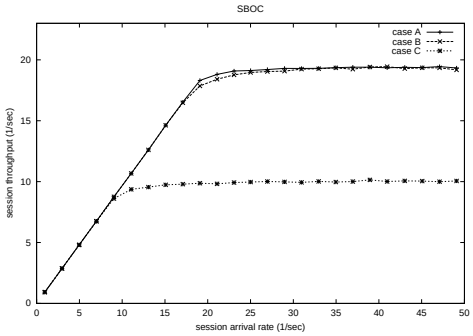
5 Conclusion Remarks

In this paper, we first introduce the queuing model of web servers with SBOC and RBOC respectively. To facilitate the investigation, we define a new performance metric called the power of web server. We use simulation to investigate the performance of two overload control schemes in the cases that the capacity of the HTTP subsystem does/does not match that of the IO subsystem. The result suggests that SBOC is effective when the web server is properly configured so that the capacities of the subsystems match each other.

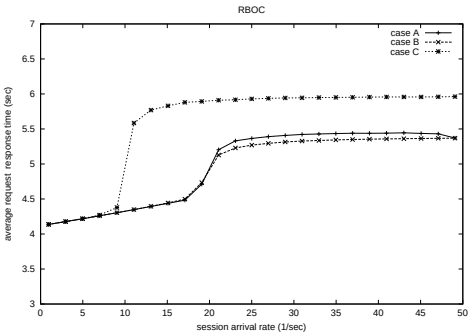
For the future work, first, it is preferred to use analytical model over



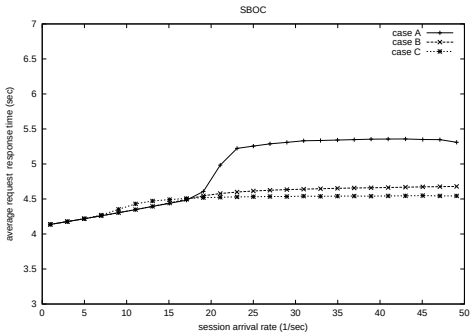
(a) Throughput (RBOC)



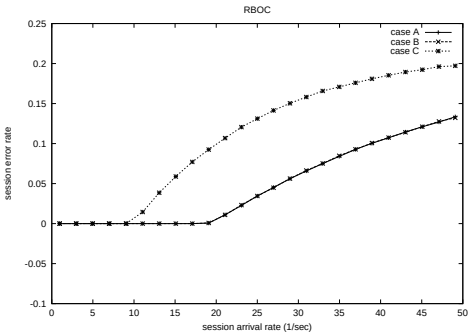
(b) Throughput (SBOC)



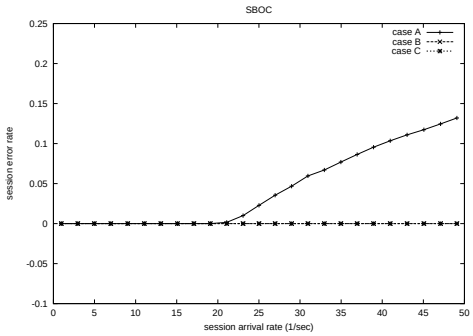
(c) Average Response Time (RBOC)



(d) Average Response Time (SBOC)

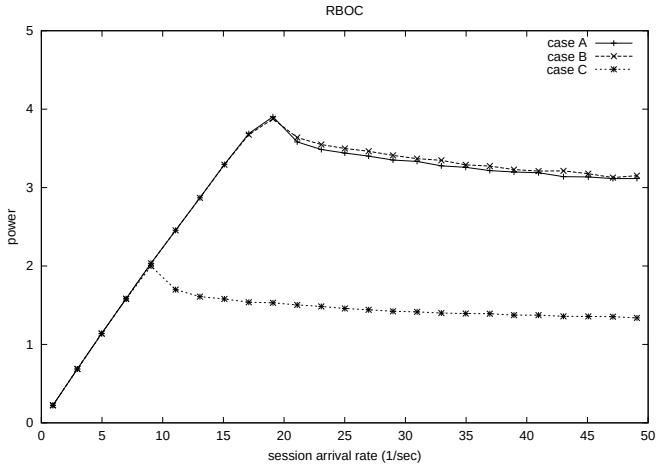


(e) Session Error Rate (RBOC)

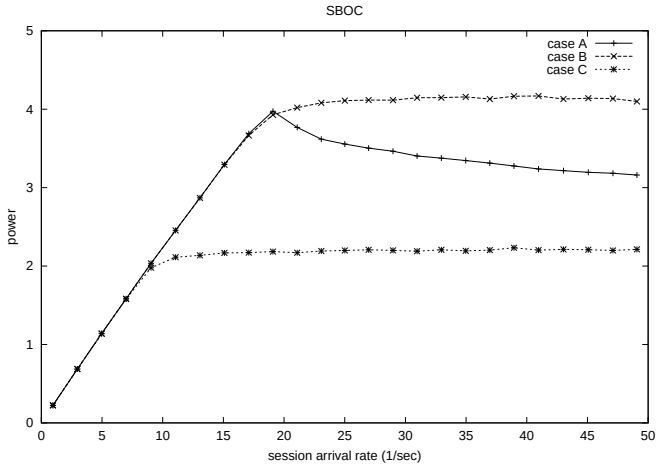


(f) Session Error Rate (SBOC)

Figure 4: The throughput, average request response time and session error rate of web servers using RBOC and SBOC in three different cases.



(a) Power (RBOC)



(b) Power (SBOC)

Figure 5: The power of web servers using RBOC and SBOC in three different cases.

simulation model in order to find out the region of the system parameters that renders the different overload control schemes effective/ineffective. Second, a more realistic model of the web server is needed. In order to simplify both simulation and analysis, the current web server model ignores the fact that the tasks like HTTP command parsing, network and file IO operation are all sharing resources such as CPU, network bandwidth and hard-disk. Hence a processor sharing based queueing network model could reveal more interesting dynamics of the web server.

References

[1] John Heidemann, Katia Orbraczka, and Joe Touch. Modeling the performance of http over several transport protocols. *IEEE/ACM Transactions on Networking*, 5(5), October 1997.

[2] Maria Kihl, Niklas Widell, and Christian Nyberg. Performance modeling of distributed e-commerce sites. In *Networking*, 2002.

[3] Zhen Liu, Nicolas Niclasusse, and Cesar Jalpa-Villanueva. Traffic model and performance evaluation of web servers. *Performance Evaluation*, 46, 2001.

[4] R. D. Van Der Mei, R. Hariharan, and P. K. Reeser. Web server performance modeling. *Telecommunication Systems*, 16(3,4):361–378, 2001.

[5] Christian Nyberg. *On Overload Control in Telecommunication Systems.* PhD thesis, Lund University, 1992.

[6] P. K. Reeser, R. D. van der Mei, and R. Hariharan. An analytical model of a web server. In *ITC16*, pages 1199–1280, 1999.