Web Server Performance Modeling using an M/G/1/K*PS Queue

Cao, Jianhua; Andersson, Mikael; Nyberg, Christian; Kihl, Maria

Published in:
ICT'2003 : 10th international conference on telecommunications. Volume 2

DOI:
10.1109/ICTEL.2003.1191656

2003

Web Server Performance Modeling Using an M/G/1/K*PS Queue

Jianhua Cao, Mikael Andersson, Christian Nyberg and Maria Kihl
Department of Communication Systems, Lund Institute of Technology
Box 118, SE-221 00 Lund, Sweden
Email: {jcao, mike, cn, maria}@telecom.lth.se

Abstract— **Performance modeling is an important topic in capacity planning and overload control for web servers. We present an M/G/1/K*PS queueing model of a web server. The arrival process of HTTP requests is assumed to be Poissonian and the service discipline is processor sharing. The total number of requests that can be processed at one time is limited to K. We obtain closed form expressions for web server performance metrics such as average response time, throughput and blocking probability. The average of the service time requirement and the limit of the number of requests being served are model parameters. The parameters are estimated by maximizing the log-likelihood function of the measured average response time. Compared to other models, our model is conceptually simple and it is easy to estimate model parameters. The model has been validated through measurements in our lab. The performance metrics predicted by the model fit well to the experimental outcome.**

I. Introduction

Performance modeling is an important part of the research area of web servers. Without a correct model of a web server it is difficult to give an accurate prediction of performance metrics. A validated model is the basis of web server capacity planning, where models are used to predict performance in different settings, see Hu et al. [1] or Menascé and Almeida [2].

Today a web site can receive millions of hits per day and it may become overloaded as the arrival rate exceeds the server capacity. To cope with this, overload control can be used, which means that some requests are allowed to be served by the web server and some are rejected. In this way the web server can achieve reasonable service times for the accepted requests. In overload control investigations for web servers, performance models predict improvements when using a certain overload control strategy, see Widell [3] or Cao and Nyberg [4]. Overload control is a research area of its own, but it is depending on performance models that are valid in the overloaded work region.

Several attempts have been made to create performance models for web servers. Van der Mei et al. [5] modeled the web server as a tandem queuing network. The model was used to predict web server performance metrics and was validated through measurements and simulations. Wells et al. [6] made a performance analysis of web servers using colored Petri nets. Their model is divided into three layers, where each layer models a certain aspect of the system. The model has several parameters, some of which are known. Unknown parameters are determined by simulations. Dilley et al. [7] used layered queuing models in their performance studies. Cherkasova and Phaal [8] used a model similar to the one presented in this paper, but with assumptions of deterministic service times and session-based workload. Beckers et al. [9] proposed a generalized processor sharing performance model for Internet access lines which include web servers. Their model describes the flow-level characteristics of the traffic carried. They established simple relations between the capacity, the utilization of the access line and download times of Internet objects.

However, several of the previous models are complicated. It lacks a simple model that is still valid in the overloaded work region. A simple model renders a smaller parameter space thus easier to estimate, while a complicated model usually contains parameters that are difficult to obtain.

A simple model, like the M/M/1/K or M/D/1/K with a First-Come-First-Served (FCFS) service discipline can predict web server performance quite well. But conceptually it is difficult to assume that the service time distribution is exponential or deterministic and that the service discipline is always FCFS.

In this paper we describe a web server model that consists of a processor sharing node with a queue attached to it. The total number of jobs in the system is limited. The arrival process to the server is assumed to be Poissonian, whereas the service time distribution is arbitrary. A system like this is called an M/G/1/K*PS queue. The average service time and the maximum number of jobs are parameters that can be determined through a maximum likelihood estimation. We also derived closed form expressions for web server performance metrics such as throughput, average response time and blocking probability.

Our validation environment consists of a server and two computers representing clients connected through a switch. The measurements validate the model. Results show that the model can predict the performance metrics at both lighter loaded and overloaded regions.

The rest of the paper is organized as follows: The next section gives an overview of how a web server works and

0-7803-7661-7/03/$17.00 ©2003 IEEE

introduces M/G/1/K*PS queue. In section III we describe our new web server model and derive expressions for the performance metrics. We develop the maximum likelihood estimation of the model parameters in Section IV. Our model is validated through experiments in Section V. Section V shows the results and the discussion. The last section concludes our work.

II. Preliminaries

This section describes how web servers work and gives a background on the theory of an M/G/1/K*PS queue.

A. Web servers

A web server contains software that offers access to documents stored on the server. Clients can browse the documents in a web browser. The documents can be for example static Hypertext Markup Language (HTML) files, image files or various script files, such as Common Gateway Interface (CGI), Java script or Perl files. The communication between clients and server is based on HTTP [10].

A HTTP transaction consists of three steps: TCP connection setup, HTTP layer processing and network processing. The TCP connection setup is performed through a three-way handshake, where the client and the server exchange TCP SYN, TCP SYN/ACK and TCP ACK messages. Once the connection has been established, a document request can be issued with a HTTP GET message to the server. The server then replies with a HTTP GET REPLY message. Finally, the TCP connection is closed by sending TCP FIN and TCP ACK messages in both directions.

Apache [11], which is a well-known web server and widely used, is multi-threaded. This means that a request is handled by its own thread or process throughout the life cycle of the request. Other types of web servers e.g. event-driven ones also exist [12]. However, in this paper we consider only the Apache web server. Apache also puts a limit on the number of processes allowed at one time in the server.

B. M/G/1/K*PS queue

Consider an M/G/1/K queue with processor sharing service discipline. The arrival of jobs is according to a Poisson process with rate λ. The service time requirements have a general distribution with mean $\bar{x}$. An arrival will be blocked if the total number of jobs in the system has reached a predetermined value K. A job in the queue receives a small quantum of service and is then suspended until every other job has received an identical quantum of service in a round-robin fashion. When a job has received the amount of service required, it leaves the queue. Such a system can also be viewed as a queueing network with one node [13].

The probability mass function (pmf) of the total number of jobs in the system has the following expression,

$$P[N=n] = \frac{(1-\rho)\rho^n}{(1-\rho^{K+1})}, \tag{1}$$

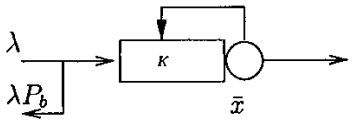

Fig. 1. An M/G/1/K-PS model of web servers

where ρ is the offered traffic and is equal to $\lambda\bar{x}$. We note that an M/M/1/K*FCFS queue has the same pmf [14], [15]. However the service time distribution of the M/M/1/K*FCFS queue must be exponential and its service discipline must be FCFS.

III. Web Server Model

We model the web server using an M/G/1/K*PS queue as Fig. 1 shows. The requests arrive according to a Poisson process with rate λ. The average service requirement of each request is $\bar{x}$. The service can handle at most K requests at a time. A request will be blocked if the number has been reached. The probability of blocking is denoted as P_b. Therefore the rate of blocked requests is given by λP_b.

From (1) we can derive the following three performance metrics, average response time, throughput and blocking probability.

The blocking probability P_b is equal to the probability that there are K jobs in the system, i.e. the system is full,

$$P_b = P[N=K] = \frac{(1-\rho)\rho^K}{(1-\rho^{K+1})}. \tag{2}$$

The throughput H is the rate of completed requests. When web server reaches equilibrium, H is equal to the rate of accepted requests,

$$H = \lambda(1-P_b). \tag{3}$$

The average response time T is the expected sojourn time of a job. Following the Little's law, we have that

$$T = \frac{E[N]}{H} = \frac{\rho^{K+1}(K\rho - K - 1) + \rho}{\lambda(1-\rho^K)(1-\rho)} \tag{4}$$

IV. Parameter Estimation

There are two parameters, $\bar{x}$ and K, in our model. We assume that the average response time for a certain arrival rate can be estimated from measurements. The estimations, $\hat{\bar{x}}$ and $\hat{K}$, are obtained by maximizing the likelihood function of the observed average response time.

Let T_i be the average response time predicted from the model and $\hat{T}_i$ be the average response time estimated from the measurements when the arrival intensity is $\lambda_i, i = 1 \ldots m$. Since the estimated response time $\hat{T}$ is the mean of samples, it is approximately a normal distributed random variable with mean T and variance σ_T^2/n when the number of samples n is very large. Hence, the model parameter pair

$(\bar{x}, K)$ can be estimated by maximizing the log-likelihood function

$$\log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma_i^2/n_i}} \exp\left[\frac{\left(\hat{T}_i - T_i\right)^2}{2\sigma_i^2/n_i}\right]. \quad (5)$$

Maximizing the log-likelihood function above is equivalent to minimize the weighted sum of square errors as follows,

$$\sum_{i=1}^{m} \frac{\left(\hat{T}_i - T_i\right)^2}{\sigma_i^2/n_i}. \quad (6)$$

As an approximation, the estimated variance of response time, $\hat{\sigma_i}^2$, can be used instead of σ_i^2.

Now, the problem of parameter estimation becomes a question of optimization,

$$(\hat{\bar{x}}, \hat{K}) = \underset{(\bar{x},K)}{\arg\min} \sum_{i=1}^{m} \frac{\left(\hat{T}_i - T_i\right)^2}{\hat{\sigma_i}^2/n_i} \quad (7)$$

The optimization can be solved in various ways, such as steepest decent, conjugate gradient, truncated Newton and even brute force searching. In this paper, we used a brute force approach. The optimum parameter is selected by examining every point of the discretized parameter space.

V. Experiments

Our validation experiments used one server computer and two client computers connected through a 100 Mbits/s Ethernet switch. The server was a PC Pentium III 1700 MHz with 512 MB RAM. The two clients were both PC Pentium III 700 with 256 MB RAM.

All computers used RedHat Linux 7.3 as operating system. Apache 1.3.9 [11] was installed in the server. We used the default configuration of the Apache, except for the maximum number of connections. The client computers were installed with a HTTP load generator, which was a modified version of S-Client [16]. The S-Client is able to generate high request rates even with few client computers by aborting TCP connection attempts that take too long time. The original version of S-Client uses deterministic waiting time between requests. We used exponential distributed waiting time instead. This makes the arrival process Poissonian [17].

The clients were programmed to request dynamically generated HTML files from the server. The CGI script was written in Perl. It generates a fix number, N_r, of random numbers, adds them together and returns the summation. By varying N_r, we can simulate different loads on the web server.

We were interested in the following performance metrics: average response time, throughput, and blocking probability. The throughput was estimated by taking the ratio between the total number of successful replies and the time span of measurement. The response time is the time difference between when a request is sent and when a successful reply is fully received. The average response time was calculated as the sample mean of the response times after removing transients. An HTTP request sent by a client computer will be blocked either when the maximum number of connections, denoted as $N_{\text{conn,max}}$, in the server has been reached or the TCP connection is timed out at the client computer. A TCP connection will be timed out by a client computer when it takes too long time for the server to return an ACK of the TCP-SYN. The blocking probability was then estimated as the ratio between the number of blocking events and the number of connection attempts in a measurement period.

TABLE I
THE CONFIGURATION OF FOUR EXPERIMENTS

	$N_r = 1000$	$N_r = 2000$
$N_{\text{conn,max}} = 75$	A1	B1
$N_{\text{conn,max}} = 150$	A2	B2

TABLE II
ESTIMATED PARAMETERS OF THE MODEL

	A1	A2	B1	B2
$\hat{\bar{x}}$	0.00708	0.00708	0.00866	0.00834
$\hat{K}$	208	286	215	298

We carried out the experiments in four cases by varying N_r and $N_{\text{conn,max}}$. Table I shows the configurations of four experiments: A1, A2, B1 and B2. In each case, the performance metrics were collected while the arrival rate (in number of requests/second) was changed from 20 to 300 with step size 20.

VI. Results and Discussion

The method developed in section IV were used to estimate the parameters from the measurements. The results are presented in Table II.

Using the estimated parameters, we can predict the web server performance and compare it with the measurements. Fig. 2 and 3 show the average response time, the throughput and the blocking probability curves. To facilitate the discussion, we divide four experiments into two groups. The first group called α contains experiments A1 and A2 and the second group β contains B1 and B2.

We notice the following relations in Table II

$$\hat{\bar{x}}_{A1} = \hat{\bar{x}}_{A2} < \hat{\bar{x}}_{B1} \approx \hat{\bar{x}}_{B2}.$$

Recall that the same CGI script is used for experiments in the same group. The script for group β is more computational intensive than the one for group α. The script for the group α adds 1000 numbers but the script for the other adds 2000 numbers. However $\bar{x}_{B1}$(or $\bar{x}_{B2}$) is not twice as large as $\bar{x}_{A1}$(or $\bar{x}_{A2}$). This can be understood as that the time spent on the summations is only a fraction of the sojourn time. Other parts of $\bar{x}$ include the connection setup time, the file transferring time, etc., which can be considered as constants in all experiments.

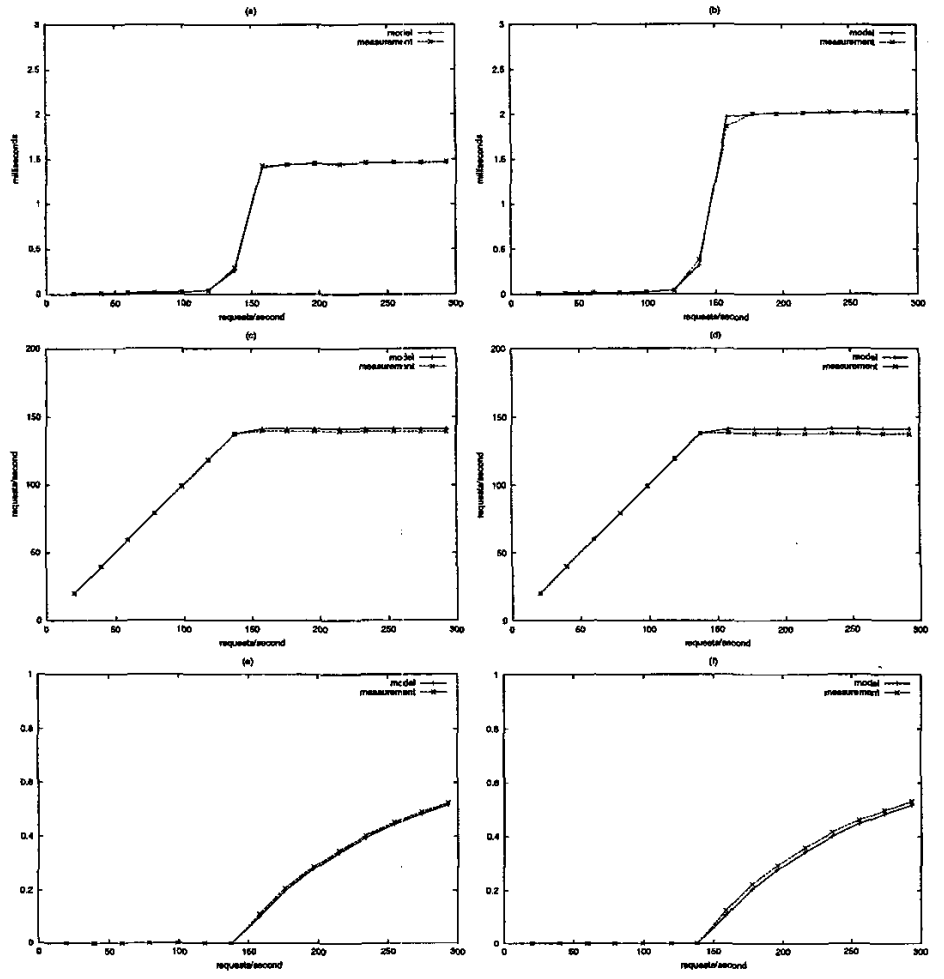

Fig. 2. (a) Average response time of A1. (b) Average response time of A2. (c) Throughput of A1. (d) Throughput of A2. (e) Blocking probability of A1. (f) Blocking probability of A2.

We find that the estimated K in all experiments is much greater than $N_{\text{conn,max}}$ which is a parameter in the Apache configuration. One may expect that $K \approx N_{\text{conn,max}}$. However, recognize that in our model K is the limit of the total number of jobs in the system. The jobs can be in the HTTP processing phase as well as in the TCP connection setup phase in which the Apache has no control. On the other hand, $N_{\text{conn,max}}$ is the maximum number of jobs handled by the Apache which runs on top of the TCP layer. Therefore K should be greater than $N_{\text{conn,max}}$.

One can reasonably predict that within the same experiment group, α or β, the difference of $\hat{K}$ should be approximately equal to the difference of $N_{\text{conn,max}}$ which is 75. In our experiments, $\hat{K}_{\text{A2}} - \hat{K}_{\text{A1}} = 78$, $\hat{K}_{\text{B2}} - \hat{K}_{\text{B1}} = 83$. There is a reason why the differences are close and greater than 75. When $N_{\text{conn,max}}$ is increased, the average load of CPU will increase. As a result, the TCP listening queue will be visited less frequently by the operating system. This implies that the TCP listening queue size will increase. So the increase of K will be greater than the increase of $N_{\text{conn,max}}$. This explanation is also supported by the fact that the increase of K in the experiment group β is larger than in the group α. As we mentioned earlier, the CGI script of the group β is more CPU demanding than that of the group α.

Now we turn our attention from the estimated parameters to the predicted performance metrics. The measured and the predicted average response time in all four experiments fit well. This should be of a little surprise because the measured average response times at various arrival rates are used to estimate the parameters of the model.

The predicted blocking probability is slightly less than

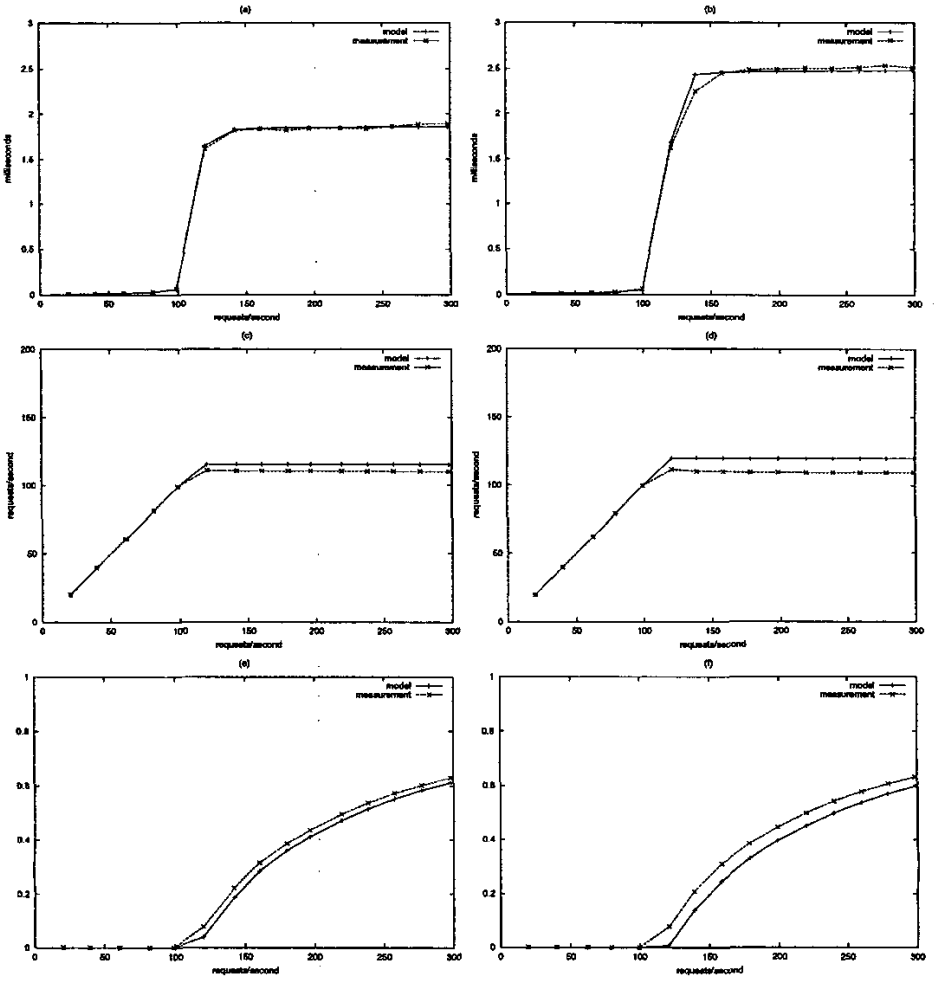

Fig. 3. (a) Average response time of B1. (b) Average response time of B2. (c) Throughput of B1. (d) Throughput of B2. (e) Blocking probability of B1. (f) Blocking probability of B2.

the measurements in all four experiments. According to (3), the error in the prediction of P_b will also affect the prediction of the throughput. Such divergence is expected since we only use the measured average response time in our parameter estimation.

VII. Conclusions

We have presented an M/G/1/K*PS queueing model of a web server. We obtained closed form expressions for web server performance metrics such as average response time, throughput and blocking probability. Model parameters were estimated from the measured average response time. We validated the model through four sets of experiments. The performance metrics predicted by the model fitted well to the experimental outcome.

Future work will include more validation under different types of loads such as network intensive and hard-disk intensive cases. It would also be interesting to see how well the model fits web servers that use an event-driven approach instead of multi-threading.

Acknowledgments

We would like to thank Thiemo Voigt for sharing his code with us and Niklas Widell for useful and interesting discussions. The work has been supported by the Swedish Research Council under contract No. 621-2001-3053.

References

[1] J. Hu, S. Mungee, and D. Schmidt, "Principles for developing and measuring high-performance web servers over ATM," in *Proceedings of INFOCOM '98, March/April 1998*, 1998.

[2] D. A. Menascé and V. A. F. Almeida, *Capacity Planning for Web Services*. Prentice Hall, 2002.

[3] N. Widell, "Performance of distributed information systems," Department of Communication Systems, Lund Institute of Technology, Tech. Rep. 144, 2002, lic. Thesis.

[4] J. Cao and C. Nyberg, "On overload control through queue length for web servers," in *16th Nordic Teletraffic Seminar*, 2002, espoo, Finland.
[5] R. D. V. D. Mei, R. Hariharan, and P. K. Reeser, "Web server performance modeling," *Telecommunication Systems*, vol. 16, no. 3,4, pp. 361–378, 2001.
[6] L. Wells, S. Christensen, L. M. Kristensen, and K. H. Mortensen, "Simulation based performance analysis of web servers," in *Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM 2001)*. IEEE Computer Society, 2001, pp. 59–68.
[7] J. Dilley, R. Friedrich, T. Jin, and J. Rolia, "Web server performance measurement and modeling techniques," *Performance Evaluation*, vol. 33, pp. 5–26, 1998.
[8] L. Cherkasova and P. Phaal, "Session-based admission control: A mechanism for peak load management of commercial web sites," *IEEE Transactions on computers*, vol. 51, no. 6, pp. 669–685, June 2002.
[9] J. Beckers, I.Hendrawan, R.E.Kooij, and R. van der Mei, "Generalized processor sharing performance model for internet access lines," in *9th IFIP Conference on Performance Modelling and Evaluation of ATM and IP Networks*, 2001, budapest.
[10] W. Stallings, *Data & Computer Communications*. Prentice Hall, 2000, sixth Edition.
[11] "Apache web server," http://www.apache.org.
[12] T. Voigt, "Overload behaviour and protection of event-driven web servers," in *In proceedings of the International Workshop on Web Engineering*, May 2002, pisa, Italy.
[13] P. J. B. King, *Computer and Communication Systems Performance Modelling*. Prentice Hall, 1990.
[14] L. Kleinrock, *Queueing Systems, Volume 1: Theory*. John Wiley & Sons, 1975.
[15] S. Lam, "Queueing networks with population size constraints," *IBM Journal of Research and Development*, vol. 21, no. 4, pp. 370–378, July 1977.
[16] G. Banga and P. Druschel, "Measuring the capacity of a web server," in *USENIX Symposium on Internet Technologies and Systems*, December 1997, pp. 61–71.
[17] R. Jain, *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.