Response Time Analysis of a Multi-Server Processor Sharing Model

Naoki Makimoto

University of Tsukuba, Bunkyo, Tokyo, Japan

Summary
In this paper, we investigate the response time of a multi-server processor sharing (PS) model. A multi-server PS model is a PS queue but the number of requests which can be processed simultaneously is limited by a threshold. An arriving request has to wait in the queue if the number of requests exceeds the threshold. Assuming Markovian arrival input of requests and general processing time distribution, we investigate how mean response time is affected by the threshold. Through numerical experiments for various instances, it turns out that the mean response time geometrically converges as the threshold increases. This property is used to derive a useful formula for capacity planning of multi-threaded server systems. We also develop a numerical procedure for computing the response time distribution. We found from numerical experiments that the processing time distribution has a strong impact on the response time distribution.

Key words:
processor sharing, response time, queueing analysis, multi-threaded server.

1. Introduction

When we develop a new system such as Web client-server system, capacity planning is a one of the most important issues to manage the system successfully. Lack of capacity could cause a serious congestion and instability of the system while too much capacity leads to large cost of investment.

In this paper, we investigate the response time of a multi-server processor sharing (PS) model. Analysis of such model is motivated by capacity planning of a multi-threaded server system. Figure 1 depicts a typical example of a system which adopts multi-threaded server architecture [5]. A request from Web clients (cellular, PC, mobile, etc.) is transmitted to the legacy server allocated at the back-most side. After execution of the request on the legacy server, it will be returned to the client. The thread is a primitive resource for all types of requests to be processed. Specifically, an arriving request directly goes into its own process if there is a vacant thread, otherwise it must wait in the queue until a thread becomes available.

When developing such type of systems, it is necessary to satisfy a prescribed quality of service (QoS). Typical examples of QoS are such that, the mean response time must be less than 4 seconds, and at least 90% of requests must be responded in 10 seconds. Among other design parameters, the number of threads is the most important parameter of the system since it determines the load to the legacy system. Therefore, accurate evaluation of response time is a key to choose appropriate number of threads.

A multi-server PS model is a PS queue but the number of requests which can be processed simultaneously is limited by a threshold. An arriving request has to wait in the queue if the number of requests exceeds the threshold. Thus, this system has a feature of both PS and ordinary queues which makes the analysis difficult. Assuming Markovian arrival input of requests and general processing time distribution, we first investigate how mean response time is affected by the threshold. Through numerical experiments for various instances, it turns out that the mean response time geometrically converges as the threshold increases. This property is then used to derive a simple and useful approximation formula for capacity planning of multi-threaded server systems. We also develop a numerical procedure for computing the distribution of the response time. Through numerical experiments, we found that among other things the processing time distribution has a strong impact on the response time distribution.

This paper is organized as follows. In the next section, we construct a multi-server PS model as a multi-server PS queue. The queueing model is then described as a continuous-time Markov process with a special structure. A numerical method for computing the stationary distribution is also discussed. Section 3 is concerned with the mean response time and the response time distribution is discussed in Section 4.

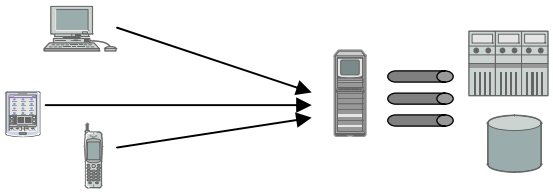

Fig. 1 An example of a multi-threaded server.

Manuscript received February 5, 2007
Manuscript revised February 25, 2007

2. Model Description

In this section, we construct a multi-server PS model as a multi-server PS queue. We assume that requests arrive to the system according to a Markovian Arrival Process (MAP) with a representation (D_0, D_1). Specifically, D_0 is a transition rate matrix of an underlying Markov process without arrivals while D_1 is a transition rate matrix accompanied with a new arrival of request. We denote by λ the mean input rate of requests. The distribution of each processing time is of phase-type (PH) with a representation (b, S) where b and S respectively denotes the initial vector and transition rate matrix of an underlying absorbing Markov process. We denote by μ the mean processing time of a request. For detailed description and basic properties of MAP and phase-type distributions, we refer to Latouche and Ramaswami [4].

The number of threads is given by m. In the context of queueing model, there are m spaces to process the requests which share the fixed total processing capacity 1. When there are k $(\leq m)$ requests in process, each of them shares equal processing capacity $1/k$. In other words, the remaining processing time of each request in process decreases at rate $1/k$. If an arriving request finds all processing spaces occupied, it must wait in the queue until one of the processing spaces becomes available. Note that the traffic intensity of the system is equal to $\rho = \lambda/\mu$. Throughout the paper, we assume $\rho < 1$ so that the system is stable.

Now we construct a continuous-time Markov process which describes time dynamics of the above system. Let N_t denote the number of requests in the system at time t, let I_t be the phase of the MAP, and let J_t be the set of processing phases at time t. Note that J_t is a vector of the form $J_t = (J_{1,t}, \ldots, J_{\min(N_t,m),t})$ since the number of requests in process is $\min(N_t, m)$. We set $J_t = 0$ when the system is empty. Then, the triplet (N_t, I_t, J_t) constitutes a continuous time Markov chain. If we order all states in a lexicographic order, the transition rate matrix of this Markov process is given as follows :

$$Q = \begin{pmatrix} B_0 & A_0 & & & & & O \\ C_1 & B_1 & A_1 & & & & \\ & C_2 & \ddots & \ddots & & & \\ & & \ddots & & A_{m-1} & & \\ & & & C_m & B_m & A & \\ & & & & C & B & \ddots \\ & & & & & \ddots & \ddots \end{pmatrix}$$

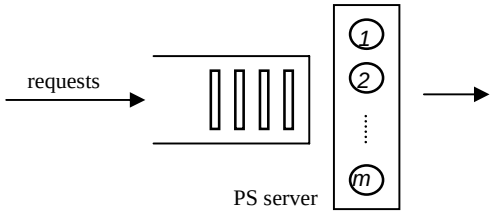

Fig. 2 A multi-server PS queue.

All submatrices of Q are give as

$$\begin{aligned} A_k &= D_1 \otimes I \otimes \cdots \otimes I \\ B_k &= D_0 \oplus (S/k) \oplus \cdots \oplus (S/k) \\ C_k &= O \oplus (S^0 b/k) \oplus \cdots \oplus (S^0 b/k) \\ A &= D_1 \otimes I \otimes \cdots \otimes I \\ B &= D_0 \oplus (S/m) \oplus \cdots \oplus (S/m) \\ C &= O \oplus (S^0 b/m) \oplus \cdots \oplus (S^0 b/m) \end{aligned}$$

where $\otimes$ and $\oplus$ denote Kronecker's product and sum, O is a zero matrix, $S^0 = -S1$ where 1 is a column vector with all components equal to 1. Here and in what follows, dimensions of all matrices and vectors should be understood properly so that all expressions are well-defined.

Let $\pi = (\pi_0, \pi_1, \pi_2, \ldots)$ be the stationary distribution of this Markov process where π_k is a vector of stationary probabilities of the states with n requests in the system. Since the transition rate matrix Q is of quasi-birth-death type, π enjoys so-called matrix-geometric form [4]. That is, we know that

$$\pi_n = \pi_m R^{n-m}, \qquad n \geq m \tag{1}$$

where R is the minimal non-negative solution to the following matrix quadratic equation

$$A + RB + R^2 C = O$$

This special structure enables us to efficiently compute π. Specifically, we first compute R by an iteration method. Starting with $R_0 = O$, we compute $R_1, R_2, \ldots$ by

$$R_n = (A + R_{n-1}^2 C)B^{-1}, \qquad n = 1, 2, \ldots$$

until R_{n-1} and R_n are close enough. This is a simple and numerically stable algorithm since it has been proved that R_n monotonically converges from below to R [4]. After computing R, we then solve the finite system of linear equations $\pi^{(m)} Q^{(m)} = 0$ to compute $\pi^{(m)} = (\pi_1, \ldots \pi_m)$ where

$$Q = \begin{pmatrix} B_0 & A_0 & & & O \\ C_1 & B_1 & A_1 & & \\ & C_2 & \ddots & \ddots & \\ & & \ddots & & A_{m-1} \\ O & & & C_m & B_m + RC \end{pmatrix}$$

The remaining part of π is given by Eq. 1.

Once the stationary distribution π is at hand, then we can compute performance measures of the system such as mean response time, mean number of requests in the system easily.

3. Mean Response Time

To see the relationship between the number of threads and mean response time, we conducted numerical experiments for various combinations of MAP and processing time distribution.

The most interesting features observed from these experiments is that the mean response time geometrically converges to that of ordinary processor sharing system as the number of threads *m* increases. To be more precise, let *RT*(*MAP/PH/m*) denote the mean response time of *MAP/PH/m* model and let

$$\Delta(MAP/PH/m) = RT(MAP/PH/m+1) - RT(MAP/PH/m)$$

Then, we observe in most examples that

$$\Delta(MAP/PH/m) \approx c\kappa^m \qquad (2)$$

holds as *m* increases for some constants *c* and κ. Figure 3 depicts an example in $M/H_2/m$, i.e., a model with a Poisson arrival of requests and hyper-exponential processing time distribution. The left panel shows that the mean response time $RT(M/H_2/m)$ (*y*-axis) geometrically converges as the number of threads (*x*-axis) increases. We can observe the geometric convergence more explicitly in the right panel where *y*-axis shows $\log|\Delta(M/H_2/m)|$. Since the graph looks like almost straight line, this implies that the relation Eq. 2 holds fairly accurately. Note here that the slope of the straight line corresponds to $\log\kappa$. Similar relation can be observed for most combinations of MAP and processing time distribution. Moreover, Eq. 2 holds with reasonable accuracy even for small *m* as seen in Figure 3.

Another important feature we observed from numerical experiments is that, when requests arrive according to a Poisson process, $\kappa \approx \rho$ provides a good approximation for the decay rate. Table 1 shows κ and ρ for various processing time distribution and traffic intensity ρ.

$$PH \quad : \quad b = (0.5, 0.5), \qquad S = \begin{pmatrix} -2 & 1 \\ 2 & -4 \end{pmatrix}$$

$$H_2 \quad : \quad b = (0.2, 0.8), \qquad S = \begin{pmatrix} -1 & 0 \\ 0 & -4 \end{pmatrix}$$

κ is estimated by a regression of $\log|\Delta(M/H_2/m)|$ with respect to *m*. As can be seen in the table, κ and ρ are close independently of the processing time distribution and ρ. Since ρ is easily computed from model parameters, this feature can be used to develop a simple approximation formula which is useful for capacity planning as we will see next.

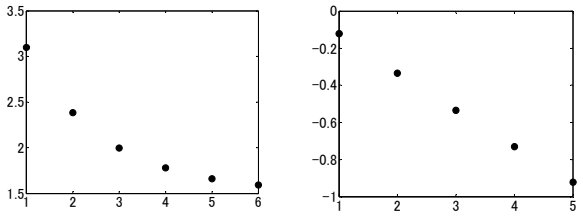

Fig. 3 $\Delta(M/H_2/m)$ and $\log|\Delta(M/H_2/m)|$.

Table 1: κ and ρ for *M/PH/m* models

Model	M/E_3		M/PH		M/H_2	
ρ	0.6	0.8	0.6	0.8	0.6	0.8
κ	0.60	0.80	0.61	0.81	0.58	0.79

First, we consider the case of Poisson arrivals. For Poisson arrivals, we have the Pollaczeck-Khinchin formula [8] for *m=1*

$$RT(M/PH/1) = \frac{1}{(1-\rho)\mu} + \frac{(C_s^2 - 1)\rho}{2(1-\rho)\mu} \tag{3}$$

where C_s stands for coefficient of variation (standard deviation divided by its mean) of the processing time distribution. On the other hand, Sakata et al. [7] proved the following formula for $m = \infty$

$$RT(M/PH/\infty) = \frac{1}{(1-\rho)\mu} \tag{4}$$

Interporating these two exact formulas by using the geometrically decaying property with the rate ρ

$$\Delta(M/PH/m) = c\rho^m, \qquad m = 2,3,\ldots$$

we can identify the coefficient *c* as

$$c = \frac{1 - C_s^2}{2\mu}$$

Combining these results, we get the following approximation for the mean response time:

$$RT(M/PH/m) \approx \frac{1}{(1-\rho)\mu} + \frac{(C_s^2 - 1)\rho^m}{2(1-\rho)\mu} \tag{5}$$

As an application of Eq. 5 to capacity planning for multi-threaded server, suppose that we need to choose the number of threads m so as to satisfy $RT(M/PH/m) \le RT_{\max}$ for a given upper limit $RT_{\max}$ of the mean response time. If the arrival of requests to the target system can be viewed as a Poisson process, we can use Eq. 5 to choose appropriate number of threads *m* as

$$m = \frac{2(1-\rho)\mu}{(C_s^2 - 1)\log\rho}\left\{RT_{\max} - \frac{1}{(1-\rho)\mu}\right\}$$

For Markovian arrival processes, we need to invoke numerical computation because no explicit formulas such as Eq. 3 and Eq. 4 are available. In the first step, we compute *RT*(*MAP/PH/m*) for $m = 1,\ldots k$ where *k* is a pre-determined threshold. Assuming the geometric decay Eq. 2, we then estimate *c* and κ from the numerical results in the first step by regression. Although we need to choose *k* a priori, the results are not different significantly from our experience since the geometric decay is rather accurate even for small *k* as we already explained before. Once *c* and κ are obtained, we can use an approximation

$$RT(MAP/PH/m) \approx RT(MAP/PH/1) + \sum_{k=1}^{m-1} c\kappa^k$$

$$= RT(MAP/PH/1) + \frac{c(\kappa - \kappa^m)}{1-\kappa}$$

to choose appropriate number of threads so as to satisfy $RT(MAP/PH/m) \le RT_{\max}$.

4. Response Time Distribution

So far, we have considered the mean response time of the system. In this section, we will investigate the response time distribution. Comparing with ordinary queueing models, it is difficult to obtain the waiting time distribution of a processor sharing queue because total processing time (duration between the beginning and the end of the process) is affected by the requests which arrive after the process has started.

We first introduce some notations. Let $y_{n,i,j}(x)$ be the tail distribution of the total processing time of a request conditioned on that the state of the system at the beginning of the service is (n,i,j). Also, let $w_{n,k,a,b,c,d}(x)$ be the joint probability of the following events: (1) waiting time of a request (we call this request A) exceeds *x*, (2) *k* requests are waiting and arrival and service phases are *c* and *d* when A's process starts; conditioned on that the state of the system was (n,a,b) when A arrived at the system. Further, we denote by $\xi_{n,i,j}$ the stationary probability at arrival epochs of requests. Then, tail distribution of the response time can be expressed as

$$P(RT > x) = \sum_{n<m,i,j} \xi_{n,i,j} y_{n,i,j}(x) - \sum_{n\ge m,a,b} \xi_{n,i,j} \sum_{k,c,d} w_{n,k,a,b,c,d}(du) y_{m+k,c,d}(x-u) \tag{6}$$

where *RT* stands for a generic random variable of a response time. Note that the first term represents the response time when a request does not have to wait while the second term represents the convolution of waiting time and total service time. In what follows, we use vector

representation such as $y_n(x) = (y_{n,\cdot,\cdot}(x))$ and $y(x) = (y_0(x), y_1(x), \ldots,)$.

Among three components ξ, $y(x)$ and $w(x)$ in Eq. 6, ξ can be easily obtained from $\xi_n = \lambda^{-1}\pi_n D_1$ once we have computed π [4]. The key issue to compute the response time distribution is therefore how to compute $y(x)$ and $w(x)$. Since a similar approach can be applied to both components, we will explain how to compute $y(x)$.

From Kolmogorov's backward equation, we see that $y(x)$ satisfies

$$y'(x) = Q_1 y(x) \tag{7}$$

where

$$Q_1 = \begin{pmatrix} B_1 & A_1 & & O \\ C_2 & B_2 & A_2 & \\ & \ddots & \ddots & \ddots \\ O & & \ddots & \ddots \end{pmatrix}$$

Eq. 7 is an extended version of the equation for M/M/1 PS model [1]. Since the solution of the differential equations Eq. 7 is given as

$$y(x) = \exp(Q_1 x)1$$

we can construct an algorithm for computing $y(x)$. Specifically, let $P_1 = I + Q_1/\nu_1$ where ν_1 is the maximum absolute value among all diagonal elements of Q_1. Then, $y(x)$ can be computed by

$$y(x) = e^{-\nu_1 x}\sum_{n=0}^{\infty}\frac{(\nu_1 x)^n}{n!}P_1^n 1 \tag{8}$$

Since the right-hand side of Eq. 8 contains only addition and product of nonnegative numbers, this method is numerically stable. When implementing the algorithm, we need to truncate Q_1 at some level and truncate infinite sum in Eq. 8. the truncation size should be chosen so that the truncation error becomes negligible.

A similar approach is applied to compute $w(x)$. That is, $w(x)$ is also expressed as $w(x) = \exp(Q_2 x)w(0)$ for some block tri-diagonal matrix Q_2. Thus, we obtain

$$w(x) = e^{-\nu_2 x}\sum_{n=0}^{\infty}\frac{(\nu_2 x)^n}{n!}P_2^n w(0) \tag{9}$$

where $P_2 = I + Q_2/\nu_2$ and ν_2 is the maximum absolute value among all diagonal elements of Q_2. Since $w(0)$ is easily computed from π, $w(x)$ can be computed in a similar way to $y(x)$.

We summarize the numerical algorithm for computing the response time distribution.

1. Compute the time stationary distribution $\pi = (\pi_0, \pi_1, \pi_2, \ldots)$ as described in Section 2.
2. Compute the stationary distribution at arrival epochs by $\xi_n = \lambda^{-1}\pi_n D_1$.
3. Compute the tail distribution $y(x)$ of the total service time from Eq. 8.
4. Compute the tail distribution $w(x)$ of the waiting time from Eq. 9.
5. Compute the response time tail distribution from Eq. 6.

In the rest of this section, we show some numerical results of the response time distribution. As a base case, we consider $M/H_2/m$ model with $\rho = 0.6$ and hyper-exponential processing time distribution

$$b = (0.1, 0.9), \qquad S = \begin{pmatrix} -1 & 0 \\ 0 & -9 \end{pmatrix}$$

Figure 4 compares the response time tail distributions for different number of threads $m = 1, 3, 5$. y-axis displays the tail probability Eq. (6) (in log scale) that the response time exceeds x. In a low-medium range of x, $M/H_2/5$ has the smallest tail probability while the order is reversed for large x. This phenomena can be explained by the trade-off between waiting time and total processing time. Intuitively, waiting time decreases as m increases since there are more spaces to be processed. On the other hand, total processing time increases as m increases since fixed processing capacity is shared by many requests. Thus, $M/H_2/5$ shows better performance in the range where waiting time dominates over total processing time and $M/H_2/1$ shows better performance in the range where total service time dominates.

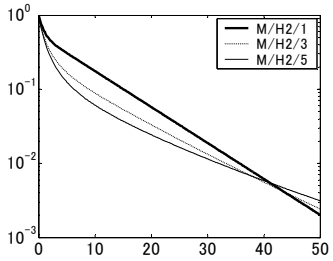

Fig. 4 Effect of the number of threads.

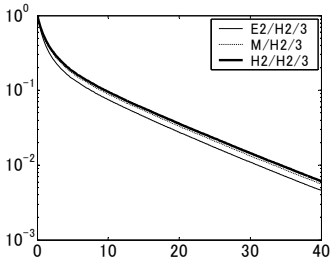

Fig. 5 Effect of input process.

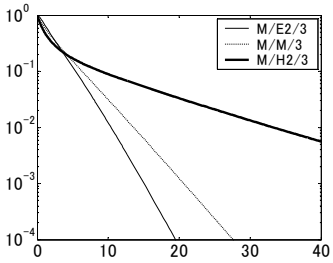

Fig. 6 Effect of processing time distribution.

Figures 5 shows the response time tail distributions of $E_2/H_2/3$, $M/H_2/3$ and $H_2/H_2/3$. Three models have the same processing time distribution but the input processes are different. As will be expected from ordinary queueing analysis, the Erlang renewal input exhibits the shortest response time and the hyper-exponential input has the longest response time. However, the graph shows that the difference of the response time is quite small. Since the number of threads is $m=3$, it seems that the fluctuation of inputs does not give a strong impact on the response time.

Finally, Figure 6 shows the response time tail distribution of $M/E_2/3$, $M/M/3$ and $M/H_2/3$. Three models have the same Poisson input but the processing time distributions are different. In contrast to Figure 5, the response time is largely affected by the processing time distribution since an asymptotic of the response time tail distribution is in principle determined by the service time distribution.

References

[1] S. Asumussen, Applied Probability and Queues, John Wiley and Sons, 1987.
[2] J.L. van den Berg, Sojourn Times in Feedback and Processor Sharing Queues, CWI, 1993.
[3] E. Gelenbe and I. Mitrani, Analysis and Synthesis of Computer Systems, Academic Press, 1980.
[4] G. Latouche and V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Models, SIAM, 1999.
[5] N. Makimoto and H. Sakata, A Hybrid Approach for Performance Evaluation of Web-Legacy Client/Server Systems, Grammar of Technology Development, Springer, 2007 (to appear).
[6] D.A. Menasce and V.A.F. Almeida, Capacity Planning for Web Performance, Prentice Hall PTR, 1998.
[7] M. Sakata, S. Noguchi and J. Oizumi, ``Analysis of a Processor Shared Queueing Model for Time Sharing Systems," Proceedings of 2nd Hawaii International Conference on System Sciences, pp.625-628, 1969.
[8] R. Wolf, Stochastic Modeling and the Theory of Queues, Prentice-Hall, 1989.

Naoki Makimoto received the B.S., M.S. and D.S. degree in Information Sciences from Tokyo Institute of Technology in 1987, 1989 and 1992, respectively. After working as a research assistant and a lecturer in the Department of Information Sciences, Tokyo Institute of Technology, he joined the Department of Systems Management, University of Tsukuba where he is currently an associate Professor. His research interest includes stochastic models and their applications to performance evaluation of information systems. He is a member of ORSJ and SIAM Japan.