Drum queueing model

by S. R. ARORA

University of Minnesota and Univac Division, Sperry Rand Corporation
Minneapolis, Minnesota

and

G. P. JAIN

Univac Division, Sperry Rand Corporation
Minneapolis, Minnesota

INTRODUCTION

This paper deals with the analysis of queues at drums in a computer system. For this analysis the computer system may be viewed as being composed of two units; (1) central processing unit and (2) the auxiliary storage devices (drum subsystem). Requests for the drum subsystem originate from the central processing unit (CPU). In a multiprogramming environment, a number of jobs are concurrently active in the system. Each job may be either waiting or being serviced by one of these units. I/O requests generated by CPU may compete for the services provided by a single drum subsystem. Concurrent requests by the CPU cause queues in front of a drum subsystem. Queuing conflicts cause delays in servicing a request and reduce maximum throughput capability of the system. These conflicts may be reduced by applying one or more of the following techniques.

1. Increasing the number of channels to a subsystem.
2. Varying the number of devices to a subsystem.
3. Increasing the speed of a subsystem.
4. Better file organization.

etc.

The purpose of this study is to develop a model for obtaining expressions for average queue size and average waiting time for various request rates and subsystem configurations. This problem is of great interest to the system designers. The proposed model will aid in determining the number of channels and also the number of drums within a drum subsystem of an optimally balanced system.

MATH. MODEL

The model considers a drum subsystem with \( n \) drums, equipped with \( m \) transfer channels (Figure 1). Requests for the drum subsystem originate from the central processing unit. A request is always for a particular drum. The servicing of a request requires the availability of both the specific drum needed by the request and any one of the transfer channels. The following assumptions are made in this model.

ASSUMPTIONS

1. Requests follow a uniform distribution over the \( n \) drum units.
2. Both the drum and the channel are considered busy during the service time of the request.
3. Requests on any drum are serviced on a first-come-first-served basis.
4. Arrivals of requests follow a Poisson Process with a rate \( \lambda_K \), where index \( K \) represents the state of the system. The state of the system is defined by the number of outstanding requests present in the drum system.
5. Service time of any request follows a negative exponential distribution with an average service time of \( 1/\mu \).

In order to determine the average queue size and the average waiting time, we need to know the stationary distribution of the number of pending requests in the drum system. Let \( p_K \) denote the stationary probability of having \( K \) requests in the system. From standard results on queuing with Poisson arrivals at rate \( \lambda_K \) and
negative exponential service times at rate $\mu_K$, corresponding to the state $K$, the expression\* for $p_K$ is given by

$$p_K = \frac{\lambda_0}{\mu_1} \times \frac{\lambda_1}{\mu_2} \times \cdots \times \frac{\lambda_{K-1}}{\mu_K} \times \mu_K$$

(1)

Different arrival patterns can be interesting. Two arrival patterns are considered in this paper. The first arrival pattern considered is given by

$$\lambda_K = \begin{cases} 
\lambda & \text{for } K < K_{\text{max}}. \\
0 & \text{for } K \geq K_{\text{max}}.
\end{cases}$$

(2)

where $K_{\text{max}}$ is a specified constant. $K_{\text{max}}$ represents the number of concurrently active jobs in the system. When all the jobs are held over for I/O the CPU is idle and stops generating further requests. However, if the pending jobs for I/O are less than $K_{\text{max}}$, the CPU continues to generate further requests at a rate of $\lambda$.

In order to compute the values of $p_K$ from equation (1), we need to know $\lambda_K$ and $\mu_K$. Equation (2) specifies $\lambda_K$. We now need to develop an expression for the service rate $\mu_K$.

$\mu_K$ is dependent upon $K$, $m$ and $n$. The maximum number of requests that can be serviced is given by the number of requests for distinctive drums within this set. The number of channels further limit the maximum number of requests that can be serviced at any time. The minimum of the distinctive requests and number of channels will determine the number of requests that can be serviced at any time. Let $G(n, K)$ denote the number of distinct drums needed for $K$ requests. The range of $G(n, K)$ is from 1 to minimum $(n, K)$.

The problem of finding the distribution\** of $G(n, K)$ is very similar to the well known problem of distributing $K$ balls randomly in $n$ boxes and finding the distribution for the number of boxes which are occupied by one or more balls. Distribution of $G(n, K)$ is given by

$$P[G(n, K) = g] = \binom{n}{g} \sum_{j=0}^{g} \binom{g}{j} \left(1 - \frac{n-g+j}{n}\right)^K$$

(3)

As pointed out earlier, the number of requests that can be serviced at any given time is also restricted by $m$, the number of channels. For $G(n, K)$ equal to $g$, the minimum of $m$ and $g$ will give the number of requests


that can be serviced. Define $S(m, n/K)$ as the minimum value of $m$ and $g$.

$$S(m, n/K) = \min(m, g)$$

$S(m, n/K)$ represents the number of requests that can be serviced simultaneously, when $K$ requests are in the system. Since $G(n, K)$ is a random variable, $S(m, n/K)$ is also a random variable. For abbreviation, we will denote $S(m, n/K)$ by $S_K$.

Probability of a request being completed from the drum system during time $t$ and $t + \Delta t$, when there are $K$ units in the system is given by $S_K p(t)$. Probability of completing more than one service during time $t$ and $t + \Delta t$ is a function involving terms of second and higher order of $\Delta t$, which is represented in the conventional notation of $O(\Delta t)$.

There can be $K$ units in the system at time $t + \Delta t$ in any of the following ways.

(i) there were $K$ units in the system at time $t$ and no arrival or departure took place during time $t$ and $t + \Delta t$.

(ii) there were $K - 1$ units at time $t$ and one arrival and no departure took place during time $t$ and $t + \Delta t$.

(iii) there were $K + 1$ units at time $t$ and no arrival and one departure took place during time $t$ and $t + \Delta t$.

All other cases involve more than one transition, which involves second and higher order terms of $\Delta t$.

$$p_K(t + \Delta t) = p_K(t) \times \text{Prob [no arrival, no departure during (t, t + \Delta t)]}$$

$$+ p_{K-1}(t) \times \text{Prob [one arrival, no departure during (t, t + \Delta t)]}$$

$$+ p_{K+1}(t) \times \text{Prob [no arrival, one departure during (t, t + \Delta t)]}$$

$$+ 0(\Delta t)$$

The probability of one arrival during $(t, t + \Delta t)$ is $\lambda \Delta t$ and the probability of no arrival during $(t, t + \Delta t)$ is $(1 - \lambda \Delta t)$. The probability of one departure during $(t, t + \Delta t)$ is a random variable and is given by $S_K \mu \Delta t$. The probability of no departure during $(t, t + \Delta t)$ is $(1 - S_K \mu \Delta t)$.

Kolmogorov equations linking the state probabilities for times $t$ and $t + \Delta t$ are given by

$$p_K(t + \Delta t) = \sum_{S_K} p_K(t) (1 - \lambda \Delta t) (1 - S_K \mu \Delta t) \phi(S_K)$$

$$+ p_{K-1}(t) \lambda \mu \Delta t$$

$$+ \sum_{S_K} p_{K+1}(t) S_K \mu \Delta t \phi(S_K) + 0(\Delta t)$$

for $K \geq 1$ (4)

and

$$p_0(t + \Delta t) = p_0(t) (1 - \lambda \Delta t) + p_1(t) S \mu \Delta t + 0(\Delta t)$$

for $K = 0$

where $\phi(S_K)$ denotes the density function of $S_K$ and $S_{K+1}$.

The stationary probabilities are obtained from equation (4) by transposing $p_K(t)$ to the left hand side, dividing throughout by $\Delta t$, taking the limit $\Delta t \to 0$ and equating the first derivative of $p_K(t)$ to zero. Equation (4) reduces to

$$\{ \lambda + Z(m, n, K) \mu \} p_K = \lambda p_{K-1} + Z(m, n, K+1) \mu p_{K+1}$$

for $K \geq 1$

and

$$\lambda p_0 = Z(m, n, 1) \mu p_1$$

for $K = 0$ (5)

where $Z(m, n, K)$ denotes the expected value of $S(m, n/K)$

$$Z(m, n, K) = \mathbb{E} S(m, n/K)$$

$Z(m, n, K)$ represents the average number of requests that can be serviced when there are $K$ requests in the system. Solving equation (5) recursively we obtain

$$p_K(m, n) = \frac{\lambda}{\mu} \frac{p_0(m, n)}{\prod_{j=1}^{K} Z(m, n, j)}$$

(7)

$$= \frac{p^K p_0(m, n)}{y_K}$$

(8)

where

$$\frac{\lambda}{\mu} = \rho$$

and

$$\prod_{j=1}^{K} Z(m, n, j) = y_K$$

$p_K(m, n)$ denotes the stationary probability of having $K$ requests in the system.

The expected service rate when there are $K$ requests in the system is given by $Z(m, n, K) \mu$, which is identical to the service rate $\mu_K$ in equation (1).

Since

$$p_0(m, n) + p_1(m, n) + \ldots + p_{K_{\text{max}}}(m, n) = 1$$

(9)

We obtain by substituting (8) in (9)

$$p_0(m, n) \left[ 1 + \frac{\rho}{y_1} + \ldots + \frac{\rho K_{\text{max}}}{y_{K_{\text{max}}}} \right] = 1$$

(10)
or

\[ p_b(m, n) = \frac{1}{1 + \frac{\rho}{y_1} + \cdots + \frac{\rho K_{\text{max}}}{y_{K_{\text{max}}}}} \]  
(11)

Substituting (11) in (8) we obtain

\[ p_K(m, n) = \frac{\rho^K}{1 + \frac{\rho}{y_1} + \cdots + \frac{\rho K_{\text{max}}}{y_{K_{\text{max}}}}} \]  
(12)

This gives the distribution of the number of requests in the system. The drum system becomes a bottleneck when the number of requests in the system reach \( K_{\text{max}} \), as this is the time, according to our assumptions, when the CPU stops sending any further requests until the time when the state of the system falls below \( K_{\text{max}} \). The quantity \( p_{K_{\text{max}}}(m, n) \) will, consequently, give the probability with which the drum system becomes a bottleneck.

**AVERAGE QUEUE SIZE**

Expression for the average queue size of waiting requests is the difference between the expected number of requests in the system and the number of requests receiving service.

\[ \text{Average queue size} = \sum_{k=0}^{K_{\text{max}}} K p_K(m, n) - \sum_{k=0}^{K_{\text{max}}} Z(m, n, K) p_K(m, n) \]  
(13)

**AVERAGE WAITING TIME**

In order to find the average waiting time we have to consider the average time required to clear the number of requests on each drum individually.

Consider the \( K \)th request in the system. Let this request be for a drum which has \((r-1)\) requests already ahead of this request. In order to find the average waiting time of a new request we have to find the time it takes to clear \((r-1)\) requests which are ahead of this request.

With the new request the total number of requests in the system are \( K \) and the number of requests for one particular drum are \( r \). The probability of having \( r \) requests for a particular drum given that there are \( K \) requests in the system follow a binomial distribution.

Let this probability be denoted by \( q(r/K) \), where

\[ q(r/K) = \binom{K}{r} \left( \frac{1}{n} \right)^r \left( 1 - \frac{1}{n} \right)^{K-r} \]  
(14)

From equation (6) we know \( z(m, n/K) \) represents the average number of requests that can be serviced when there are \( K \) requests for \( n \) drums in the system. Therefore \( \left[ z(m, n/K) \right]/n \) gives the average number of requests that can be serviced from an individual drum in \( 1/\mu \) time.

The time to service \( r \) requests will be \( n/[z(m, n/K)] \) \((r/\mu)\).

The average waiting time (AWT) of the \( K \)th request will be given by (the service time of \( r \) requests—service time of \( r \)th request for this drum).

\[ \text{AWT} = \sum_{K=0}^{K_{\text{max}}} \left[ \sum_{r=0}^{n} \frac{r}{\mu} \cdot q(r/K) p_K(m, n) \right] \]  
(15)

Another arrival pattern of interest is where the request rate \( \lambda_K \) gradually goes down as the queue gets larger.

\[ \left\{ \begin{array}{ll}
\lambda & \text{for } K = 0 \\
\lambda_K/\mu & \text{for } 0 < K < K_{\text{max}} \\
0 & \text{for } K \geq K_{\text{max}}
\end{array} \right. \]

The probabilities \( p_K(n, m) \) for this is given by

\[ p_K(n, m) = \left( \frac{\lambda}{\mu} \right)^K \prod_{j=1}^{K} \frac{Z(m, n, j)^{\rho}}{y_{K_{\text{max}}}} \]  
(8a)

By substituting (8a) in (9) we obtain

\[ p_b(m, n) \left[ 1 + \frac{\rho}{y_1} + \cdots + \frac{\rho K_{\text{max}}}{y_{K_{\text{max}}}} \right] = 1 \]

\[ p_b(m, n) = \frac{1}{1 + \frac{\rho}{y_1} + \cdots + \frac{\rho K_{\text{max}}}{y_{K_{\text{max}}}}} \]  
(11a)

Substituting (11a) in (8a) we obtain

\[ p_K(m, n) = \frac{\rho K}{y_1} \]  
(12a)
Formulas for Average Queue Size and Average Waiting Time are given by equations (13) and (15) respectively.

### ANALYSIS

Theoretically, if the requests are generated faster than these can be serviced the queue may approach infinity but in a computer system the number of requests that can be made to the drum subsystem are restricted. The number of requests are limited by the number of active jobs and the amount of parallel I/O activity within each job. For this hypothetical example it is assumed that \( \mu = .011 \) and the maximum number of requests \((K)\) in the system are 20, i.e., for \( K = 20 \) the system stops generating any more requests. It is further assumed that the system is generating requests at the rate of \( \lambda \).

Table I presents the mean queue distribution and waiting time distribution for various combinations of subsystem request rates, and subsystem configurations when the CPU request rates remain independent of queue length.

Some of these results are presented in Figures 2, 3 and 4. One can observe from Figure 1 that in a single channel system the queue size and the waiting time are not affected by change in the number of drums on a subsystem. But as \( \rho \) approaches 1 the queue size and the waiting time get extremely large.

---

**TABLE I**

<table>
<thead>
<tr>
<th>No. of drums ( n )</th>
<th>( \lambda/\mu = \rho )</th>
<th>m = 1</th>
<th>m = 2</th>
<th>m = 4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AQS AQTS AQTS</td>
<td>AWT</td>
<td>AWT</td>
<td>AWT</td>
</tr>
<tr>
<td>2</td>
<td>( .3 )</td>
<td>.13</td>
<td>11.7</td>
<td>.03</td>
</tr>
<tr>
<td>4</td>
<td>( .13 )</td>
<td>11.7</td>
<td>.02</td>
<td>.9</td>
</tr>
<tr>
<td>6</td>
<td>( .13 )</td>
<td>11.7</td>
<td>.01</td>
<td>.7</td>
</tr>
<tr>
<td>8</td>
<td>( .13 )</td>
<td>11.7</td>
<td>.01</td>
<td>.5</td>
</tr>
<tr>
<td>10</td>
<td>( .13 )</td>
<td>11.7</td>
<td>.01</td>
<td>.3</td>
</tr>
<tr>
<td>2</td>
<td>( .5 )</td>
<td>.50</td>
<td>45.4</td>
<td>.11</td>
</tr>
<tr>
<td>4</td>
<td>( .50 )</td>
<td>45.4</td>
<td>.06</td>
<td>3.0</td>
</tr>
<tr>
<td>6</td>
<td>( .50 )</td>
<td>45.4</td>
<td>.05</td>
<td>2.3</td>
</tr>
<tr>
<td>8</td>
<td>( .50 )</td>
<td>45.4</td>
<td>.04</td>
<td>2.1</td>
</tr>
<tr>
<td>10</td>
<td>( .50 )</td>
<td>45.4</td>
<td>.04</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>( .7 )</td>
<td>1.62</td>
<td>147.4</td>
<td>.24</td>
</tr>
<tr>
<td>4</td>
<td>( 1.62 )</td>
<td>147.4</td>
<td>.15</td>
<td>7.0</td>
</tr>
<tr>
<td>6</td>
<td>( 1.62 )</td>
<td>147.4</td>
<td>.13</td>
<td>5.9</td>
</tr>
<tr>
<td>8</td>
<td>( 1.62 )</td>
<td>147.4</td>
<td>.12</td>
<td>5.5</td>
</tr>
<tr>
<td>10</td>
<td>( 1.62 )</td>
<td>147.4</td>
<td>.12</td>
<td>5.2</td>
</tr>
<tr>
<td>2</td>
<td>( .9 )</td>
<td>5.53</td>
<td>502.9</td>
<td>.45</td>
</tr>
<tr>
<td>4</td>
<td>( 5.53 )</td>
<td>502.9</td>
<td>.31</td>
<td>14.4</td>
</tr>
<tr>
<td>6</td>
<td>( 5.53 )</td>
<td>502.9</td>
<td>.28</td>
<td>12.7</td>
</tr>
<tr>
<td>8</td>
<td>( 5.53 )</td>
<td>502.9</td>
<td>.26</td>
<td>12.0</td>
</tr>
<tr>
<td>10</td>
<td>( 5.53 )</td>
<td>502.9</td>
<td>.25</td>
<td>11.7</td>
</tr>
<tr>
<td>2</td>
<td>( 1.1 )</td>
<td>12.3</td>
<td>1117.9</td>
<td>.81</td>
</tr>
<tr>
<td>4</td>
<td>( 12.3 )</td>
<td>1117.9</td>
<td>.59</td>
<td>27.2</td>
</tr>
<tr>
<td>6</td>
<td>( 12.3 )</td>
<td>1117.9</td>
<td>.54</td>
<td>24.9</td>
</tr>
<tr>
<td>8</td>
<td>( 12.3 )</td>
<td>1117.9</td>
<td>.52</td>
<td>23.9</td>
</tr>
<tr>
<td>10</td>
<td>( 12.3 )</td>
<td>1117.9</td>
<td>.51</td>
<td>23.2</td>
</tr>
<tr>
<td>2</td>
<td>( 1.3 )</td>
<td>15.75</td>
<td>1433.0</td>
<td>1.4</td>
</tr>
<tr>
<td>4</td>
<td>( 15.75 )</td>
<td>1433.0</td>
<td>1.1</td>
<td>50.3</td>
</tr>
<tr>
<td>6</td>
<td>( 15.75 )</td>
<td>1433.0</td>
<td>1.0</td>
<td>47.2</td>
</tr>
<tr>
<td>8</td>
<td>( 15.75 )</td>
<td>1433.0</td>
<td>1.0</td>
<td>46.0</td>
</tr>
<tr>
<td>10</td>
<td>( 15.75 )</td>
<td>1433.0</td>
<td>1.0</td>
<td>45.0</td>
</tr>
</tbody>
</table>
Addition of drums to a dual channel system reduces both the queue size and the waiting time but for more than four drums to a system the queue size and waiting time do not change significantly. (See Figures 2 and 3.)

Another observation which can be made from Table I is that the addition of an extra channel to a single channel system doubles the throughput rate. Also, the queue size is slightly decreased.

SUMMARY

Many systems and processes in use today are quite complex. Queuing problems occur from time to time when there are sufficient requests or severe irregularity in the system. Multiprogramming environment may place a heavy load on the auxiliary storage devices and thus cause queues. Therefore it is important to design an auxiliary storage system for the desired response time.

Standard queuing formulas cannot be used to find the queue size and waiting times for a drum system because in a drum system the queue size is restricted and the request rate goes down as the queue gets larger. With these two unique features in mind an analytical queuing model is developed to provide estimates of queue size and waiting times. The output of this model enables the system designer to identify drum system limiting factors and allow the determination of system sensitivities.