Dataflow modelling in distributed diagnostic processing systems: a closed queuing network model with multiple servers

Vidhyacharan Bhaskar, K. Adjallah, Laurie Joiner

DATAFLOW MODELLING IN DISTRIBUTED DIAGNOSTIC-PROCESSING SYSTEMS: A CLOSED QUEUING NETWORK MODEL APPROACH WITH MULTIPLE SERVERS

Vidhyacharan Bhaskar[1] §, Kondo Hloindo Adjallah[2], Laurie L. Joiner[3]

[1]Departement Genie des Systemes d'Information
et de Telecommunication
Universite de Technologie de Troyes
12 Rue Marie Curie, 10010 Troyes Cedex, FRANCE
E-mail: Vidhyacharan.Bhaskar@utt.fr

[2]Institute of Computer Science and Engineering of Troyes
Universite de Technologie de Troyes
12 Rue Marie Curie, 10010 Troyes Cedex, FRANCE

[3]Department of Electrical and Computer Engineering
University of Alabama in Huntsville
Huntsville, AL 35899, USA

Abstract: In this paper, a closed queuing network model with multiple servers has been proposed to model dataflow in distributed diagnostic-processing systems. Multi-threading is useful in reducing the latency by switching among a set of threads in order to improve the processor utilization. Two sets of processors, synchronization and execution processors exist. Synchronization processors handle load/store operations and execution processors handle arithmetic/logic and control operations. A closed queuing network model is suitable for large number of job arrivals. The normalization constant is derived using a recursive algorithm for the given model. Performance measures such as average response times and average system throughput are derived and plotted against the total

number of processors in the closed queuing network model. Other important performance measures like processor utilizations, average queue lengths, average waiting times and relative utilizations are also derived.

AMS Subject Classification: 26A33
Key Words: Synchronization and Execution processors, Multi-programming, Queue lengths, Response times, Utilizations, Throughput.

1. Introduction

In an open-queuing network model, a job is scheduled to the main-memory and is able to compete for active resources such as synchronization and execution processors immediately on its arrival [11]. In practice, the number of main-memory partitions are limited. So, the existence of an additional queue is necessary. This queue is called a job-scheduler queue [11]. However, such a network is said to be multiple-resources holding. This is because a job cannot simultaneously hold main-memory and an active device. Such a network cannot be solved by product-form methods.

If the external arrival rate is low, then the probability that the job has to wait in the scheduler queue is also low. Thus, the open-queuing network model is a good solution when the arrival rate is low [1]. In other words, an open-queuing network model is a light-load approximation.

If the external arrival rate is high, the probability that there is at least one customer in the job-scheduler queue is very high. The departure of a job from the active-set immediately triggers the scheduling of an already waiting job into main-memory. Thus, a closed-queuing network model becomes imperative.

Input to the synchronization processor is in the form of threads and comprises a statistically determined sequence of RISC-style instructions [9]. Threads (sequence of tasks) are scheduled dynamically to be executed by the execution processors. The threads have a bounded execution time [9]. Our model also represents a distributed shared memory system (DSM) model in which all processors share a common address space [3]. So, the memory access time depends on the location of the accessed data.

Multi-threading can also achieve higher instruction rates on processors which contain multiple functional units (e.g. super-scalars) or

multiple-processing elements (e.g. chip multiprocessors) [7]. To achieve higher performance, it is therefore necessary to optimize the number of synchronization and execution units to find an appropriate multi-threaded model.

In [8], each thread is comprised of conventional control-flow instructions. These instructions do not retain functional properties and need Write-after-Write (WAW) and Write-After-Read (WAR) dependencies [4].

In [10], a few limitations of the pure dataflow model are presented. They are: 1) too-fine grained (instruction level) multi-threading and 2) difficulty in exploiting the memory hierarchies and registers [10]. However, in the model developed in [5], the instructions within a thread retain functional properties of dataflow model and thus eliminates the need for complex hardware. Our work models the dataflow instructions appearing in [5].

Section 2 shows the block diagram of the closed queuing network model with multiple servers, and describes in detail the calculation of the normalization constant and utilizations. Section 3 discusses the performance measures such as queue lengths, response times, waiting times, throughput and utilizations related model with multiple servers. Section 4 discusses the simulation results of the queuing model. Finally, Section 5 presents the conclusions.

2. System Model: Multiple Servers

Figure 1 represents a good approximation to an open-queuing network model having multiple servers for each queue under heavy-load conditions.

Each job circulating in the closed network is said to be an active job and must be allocated a partition of main memory. The number of active jobs, k'_1 at server 1 among the EPs, is the sum of the number of jobs (tasks) currently being served (either 0 or 1) by server EP_1 and the number of jobs, k_{EP} waiting in the queue, Q_{EP}. The number of active jobs, k'_j, at node j $(2 \leq j \leq n)$ among the EPs, is the number of jobs currently being served by server EP_j $(2 \leq j \leq n)$, which is either 0 or 1. For a closed queuing model with m servers for SPs and n servers for EPs, N is the total number of tasks at the m SPs and n EPs, or $N = k_1 + k_2 + \ldots + k_m + k'_1 + k'_2 + \ldots + k'_n$, where N is the total number

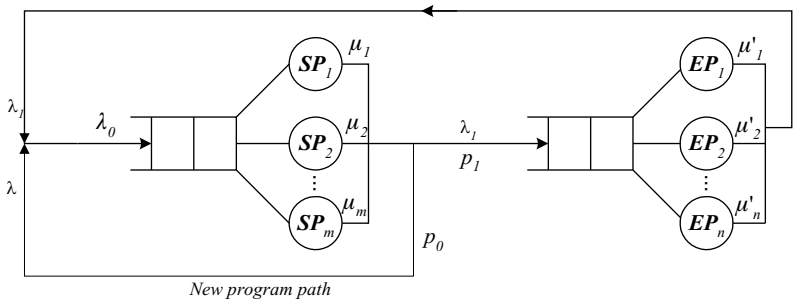

Figure 1: Multiple servers - closed queuing model with feedback

of jobs in the system. The total number of active jobs, N, is called the degree of multi-programming [11].

The number of jobs in the system is restricted to N due to the constraints of finite main memory [6]. Ready jobs will wait at their terminals until some job leaves from the active set, at which time one of the ready jobs enters the active set and is allowed to compete for the system resources [6].

Let the external arrival rate be λ, arrival rate to all SPs be λ_0, and the arrival rate to the EPs be λ_1. So,

$$\lambda_1 = \lambda_0 p_1. \tag{1}$$

The total arrival rate to all SPs is equal to sum of the external arrival rate and the arrival rate to all EPs. Hence,

$$\lambda_0 = \lambda + \lambda_1. \tag{2}$$

Substituting (1) in (2) and rearranging,

$$\lambda_0 = \frac{\lambda}{1 - p_1}. \tag{3}$$

Also,

$$p_0 + p_1 = 1. \tag{4}$$

Substituting (4) in (3), we have

$$\lambda_0 = \frac{\lambda}{p_0}. \tag{5}$$

Equation 5 is valid only at steady state. The arrival rate to the EPs is

$$\lambda_1 = \lambda_0 p_1 = \frac{\lambda}{p_0} p_1. \tag{6}$$

Let μ_i be the service rate (rate at which tasks are being served) of server i among the SPs, and let μ'_j be the service rate of server j among the EPs. The utilization of each SP_i $(1 \leq i \leq m)$ is

$$\gamma_i = \frac{\lambda_0}{\mu_i} = \frac{\lambda}{p_0 \mu_i}, \tag{7}$$

and the utilization of each EP_j $(1 \leq j \leq n)$ is

$$\gamma'_j = \frac{\lambda_1}{\mu'_j} = \frac{\lambda p_1}{p_0 \mu'_j}. \tag{8}$$

Consider the closed network of queues shown in Figure 1. The state of the network is given by an $(m+n)$-tuple vector,

$$s = \left(k_1, k_2, \ldots, k_m, k'_1, k'_2, \ldots, k'_n\right).$$

Assuming that the service times of all servers are exponentially distributed, the stochastic process modelling the behavior of the network is a finite-state homogeneous continuous-time Markov chain (CTMC), which can be shown to be irreducible and recurrent non-null [11] (assuming that $0 < p_0 < 1$ and $0 < p_1 < 1$).

The transient probability matrix, $\mathbf{T}$, of the Discrete-time Markov chain (DTMC) is given by

$$\mathbf{T} = \begin{bmatrix} p_0 & p_1 \\ 1 & 0 \end{bmatrix}. \tag{9}$$

Notice that all rows sum to 1, i.e., $p_0 + p_1 = 1$. The DTMC is finite, and if we assume that $0 < p_0 < 1$ and $0 < p_1 < 1$, then the DTMC can be shown to be irreducible and periodic. Then, the unique relative visit count vector $\underline{\mathbf{v}} = (v_0, v_1)$ can be obtained by solving the system of equations,

$$\underline{\mathbf{v}} = \underline{\mathbf{v}}\mathbf{T}. \tag{10}$$

If we observe the system for a real-time interval of duration τ, then the terms $v_0\tau$ and $v_1\tau$ represent the average number of visits to SPs and

EPs respectively [11]. In this sense, v_0 can be thought of as a relative visit count to the SPs, and v_1 as a relative visit count to the EPs. Thus, v_0 (and v_1) represent the relative throughputs of the respective nodes. For the network of queues shown in Figure 1, (10) becomes

$$[v_0\, v_1] = [v_0\, v_1]\, \mathbf{T}. \tag{11}$$

The system of linear equations represented by (11) is

$$\begin{aligned} v_0 &= v_0 p_0 + v_1 \Rightarrow v_1 = v_0(1-p_0) \\ v_1 &= v_0 p_1. \end{aligned} \tag{12}$$

It is clear that both the equations shown above are identical because $p_0 + p_1 = 1$. v_0 can be chosen as any real value that will aid us in our computations. The usual choices for v_0 are $\frac{1}{p_0}$, μ_1, and 1. If we choose $v_0 = \frac{1}{p_0}$, then from (12), we have

$$v_1 = \left(\frac{1-p_0}{p_0}\right). \tag{13}$$

The relative utilization of node i $(1 \le i \le m)$ is

$$\rho_i = \frac{v_0}{\mu_i} = \frac{1}{p_0 \mu_i}. \tag{14}$$

The relative utilization of node j $(1 \le j \le n)$ is

$$\rho'_j = \frac{v_1}{\mu'_j} = \frac{p_1}{p_0 \mu'_j}. \tag{15}$$

Substituting (14) and (15) in the expression for steady-state probability, see [2],

$$\begin{aligned} p\left(k_1, \ldots, k_m, k'_1, \ldots, k'_n\right) &= \frac{1}{C(N)} \prod_{i=1}^{m} \rho_i^{k_i} \prod_{j=1}^{n} \left(\rho'_j\right)^{k'_j} \\ &= \frac{1}{C(N)} \prod_{i=1}^{m} \left(\frac{1}{p_0 \mu_i}\right)^{k_i} \prod_{j=1}^{n} \left(\frac{p_1}{p_0 \mu'_j}\right)^{k'_j} \end{aligned} \tag{16}$$

where $C(N)$ is the normalization constant chosen so that the sum of the steady-state probabilities is one. The normalization constant can be expressed as [11]

$$C(N) = \sum_{s \in I} \prod_{i=1}^{m} \rho_i^{k_i} \prod_{j=1}^{n} \left(\rho'_j\right)^{k'_j}, \tag{17}$$

where the state space

$$I = \Bigg\{(k_1, k_2, \ldots, k_m, k'_1, k'_2, \ldots, k'_n) | k_i \geq 0, \\ k'_j \geq 0 \; \forall \; i, j \text{ and } \sum_{i=1}^{m} k_i + \sum_{j=1}^{n} k'_j = N \Bigg\}, \quad (18)$$

and $s = (k_1, k_2, \ldots, k_i, k'_1, k'_2, \ldots, k'_n)$ is a particular state of the network.

2.1. Normalization Constant: Recursive Algorithm

The computation of the normalization constant from (17) is very expensive and numerically unstable because the number of states grow exponentially with the number of customers and the number of service centers [11]. It is therefore necessary to derive computationally stable and efficient algorithms to obtain the normalization constant, $C(N)$.
Let us consider the following polynomial in z.

$$\begin{aligned} G(z) &= \prod_{i=1}^{m} \frac{1}{1-\rho_i z} \prod_{j=1}^{n} \frac{1}{1-\rho'_j z} \\ &= \left(1 + \rho_1 z + (\rho_1 z)^2 + \ldots\right) \times \ldots \times \left(1 + \rho_m z + (\rho_m z)^2 + \ldots\right) \\ &\times \left(1 + \rho'_1 z + \left(\rho'_1 z\right)^2 + \ldots\right) \times \ldots \times \left(1 + \rho'_n z + \left(\rho'_n z\right)^2 + \ldots\right). \quad (19) \end{aligned}$$

Now, the coefficient of $G(z)$ is equal to the normalization constant, $C(N)$, since the coefficient is the sum of all terms of the form

$$\rho_1^{k_1} \rho_2^{k_2} \ldots \rho_m^{k_m} \left(\rho'_1\right)^{k'_1} \left(\rho'_2\right)^{k'_2} \ldots \left(\rho'_n\right)^{k'_n},$$

with $\sum_{i=1}^{m} k_i + \sum_{j=1}^{n} k'_j = N$. In other words, $G(z)$ is the generating function of the sequence $C(1), C(2), \ldots$
We can write

$$G(z) = \sum_{N=0}^{\infty} C(N) z^N = C(0) + C(1)z + C(2)z^2 + \ldots \quad (20)$$

where $C(0)$ is defined to be equal to unity [11]. Since $C(N)$ is not a probability, $G(z)$ is not necessarily equal to unity. The polynomial

$G(z) \geq 0$. In order to derive a recursive relation for computing $C(N)$, we define

$$\begin{aligned} G_{i,j}(z) &= \prod_{k=1}^{i} \frac{1}{(1-\rho_k z)} \prod_{l=1}^{j} \frac{1}{\left(1-\rho'_l z\right)} = \frac{1}{(1-\rho_1 z)\ldots(1-\rho_i z)} \\ &\quad \times \frac{1}{(1-\rho'_1 z)\ldots\left(1-\rho'_j z\right)} \end{aligned} \tag{21}$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$. Here, $G_{m,n}(z) = G(z)$. Also, let us define $C_{i,j}(l)$ as

$$G_{i,j}(z) = \sum_{l=0}^{\infty} C_{i,j}(l) z^l \,, \tag{22}$$

where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$ so that $C_{m,n}(l) = C(l)$. Let

$$G_{1,0}(z) = \frac{1}{1-\rho_1 z}. \tag{23}$$

The generalized expression is

$$G_{i,j}(z) = G_{i-1,j}(z) \frac{1}{1-\rho_i z}. \tag{24}$$

Now, (24) can be rewritten as

$$\begin{aligned} G_{i,j}(z)\,(1-\rho_i z) &= G_{i-1,j}(z)\,, \\ \text{i.e.,} \qquad G_{i,j}(z) &= \rho_i z G_{i,j}(z) + G_{i-1,j}(z)\,, \\ \text{i.e.,} \quad \sum_{l=0}^{\infty} C_{i,j}(l) z^l &= \sum_{l=0}^{\infty} \rho_i z C_{i,j}(l) z^l + \sum_{l=0}^{\infty} C_{i-1,j}(l) z^l. \end{aligned} \tag{25}$$

Equating the coefficients of z^l on both sides, we have a recursive formula,

$$C_{i,j}(l) = \rho_i C_{i,j}(l-1) + C_{i-1,j}(l), \tag{26}$$

where $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, N$. The initialization is obtained from (22) and (23) as

$$C_{1,0}(l) = \rho_1^l \;\forall\, l = 0, 1, \ldots, N. \tag{27}$$

Also from (21), we have the coefficient of z^0 in $G_{i,j}(z)$ as unity. Hence,

$$C_{i,j}(0) = 1 \;\; \forall\, i = 1, 2, \ldots, m \;\text{ and } \forall\, j = 1, 2, \ldots, n. \tag{28}$$

2.2. Utilization of i-th Device

Consider a slight modification to the generating function, $G(z)$, denoted as $H_i(z)$:

$$\begin{aligned} H_i(z) &= \left(\prod_{j=1(j\neq i)}^{m}\frac{1}{1-\rho_j z}\right)\left(\prod_{k=1}^{n}\frac{1}{1-\rho'_k z}\right)\left(\frac{1}{1-\rho_i z}-1\right) \\ &= \left(1+\rho_1 z+(\rho_1 z)^2+\ldots\right)\times\ldots\times\left(1+\rho_{i-1}z+(\rho_{i-1}z)^2+\ldots\right) \\ &\quad\times\left(\rho_i z+(\rho_i z)^2+\ldots\right)\left(1+\rho_{i+1}z+(\rho_{i+1}z)^2+\ldots\right)\times\ldots \\ &\quad\times\left(1+\rho_m z+(\rho_m z)^2+\ldots\right)\left(1+\rho'_1 z+\left(\rho'_1 z\right)^2+\ldots\right)\times\ldots \\ &\quad\times\left(1+\rho'_n z+\left(\rho'_n z\right)^2+\ldots\right). \end{aligned} \tag{29}$$

The difference between $H_i(z)$ and $G(z)$ is that the first term is omitted in the factor corresponding to the i^{th} device. So, the coefficient of z^N in $H(z)$ will be the sum of all terms $\rho_1^{k_1}\rho_2^{k_2}\ldots\rho_m^{k_m}\left(\rho'_1\right)^{k'_1}\left(\rho'_2\right)^{k'_2}\ldots\left(\rho'_n\right)^{k'_n}$, where $k_i \geq 1\ \forall\, i=1,2,\ldots,m$ and $k'_j \geq 1\ \forall\, j=1,2,\ldots,n$. Here, ρ_i is the relative utilization of device i $(1\leq i\leq m)$, and ρ'_j is the relative utilization of device j $(1\leq j\leq n)$.

From (16), we see that the coefficient of z^N in $G(z)$ yields the marginal probability $P\left(N_i\geq 1\right)$, which is exactly the utilization $U_i(N)$. So,

$$H_i(z)=\frac{G(z)\left[\dfrac{1}{1-\rho_i z}-1\right]}{\dfrac{1}{1-\rho_i z}}=\rho_i z G(z). \tag{30}$$

From (30), the coefficient of z^N in $H_i(z)$ is simply ρ_i times the coefficient of z^{N-1} in $G(z)$. Therefore, we have

$$U_i(N)=\rho_i\frac{C(N-1)}{C(N)}=\left(\frac{1}{p_0\mu_i}\right)\frac{C(N-1)}{C(N)} \tag{31}$$

as the utilization of the i^{th} node among the SPs.
Similarly, the utilization of the j^{th} node among the EPs is

$$U'_j(N)=\rho'_j\frac{C(N-1)}{C(N)}=\left(\frac{p_1}{p_0\mu'_j}\right)\frac{C(N-1)}{C(N)}. \tag{32}$$

The ratio of the actual utilizations of SPs and EPs when there are N jobs in the system is given by $U_i(N)/U'_j(N)$. From (31) and (32), the ratio of actual utilizations is

$$\frac{U_i(N)}{U'_j(N)} = \frac{\rho_i}{\rho'_j} = \left(\frac{\mu'_j}{\mu_i}\right)\left(\frac{1}{p_1}\right). \tag{33}$$

Equation 33 explains the reason for calling ρ_i and ρ'_j "relative utilizations".

3. Performance Measures

3.1. Queue Lengths

The probability that there are k or more jobs at node i is given by [11]

$$P\left(N_i \geq k\right) = \rho_i^k \frac{C(N-k)}{C(N)}. \tag{34}$$

The average queue length at node i in the SPs when there are N jobs in the system is given by

$$E\left[L_i(N)\right] = \sum_{l=1}^{N} \rho_i^l \frac{C\left(N-l\right)}{C(N)}. \tag{35}$$

Similarly, the average queue length at node j in the EPs when there are N jobs in the system is given by

$$E\left[L'_j(N)\right] = \sum_{l=1}^{N} \left(\rho'_j\right)^l \frac{C\left(N-l\right)}{C(N)}. \tag{36}$$

The total number of jobs in the system is equal to the sum of the expected number of jobs in SPs and EPs. i.e.,

$$\begin{aligned} L &= \sum_{i=1}^{m} E\left[L_i(N)\right] + \sum_{j=1}^{n} E\left[L'_j(N)\right] \\ &= \sum_{i=1}^{m}\sum_{l=1}^{N} \rho_i^l \frac{C(N-l)}{C(N)} + \sum_{j=1}^{n}\sum_{l=1}^{N} \left(\rho'_j\right)^l \frac{C(N-l)}{C(N)} \end{aligned}$$

$$= \frac{1}{C(N)}\left\{\sum_{i=1}^{m}\sum_{l=1}^{N}\left(\frac{1}{p_0\mu_i}\right)^l C(N-l) + \sum_{j=1}^{n}\sum_{l=1}^{N}\left(\frac{p_1}{p_0\mu'_j}\right)^l C(N-l)\right\}. \tag{37}$$

3.2. Normalization Constant

Cross-multiplying in (37), and replacing L by N, the total number of jobs in SPs and EPs, an alternative recursive formula for the computation of $C(N)$ is

$$C(N) = \frac{1}{N}\sum_{l=1}^{N} C(N-l)\left[\sum_{i=1}^{m}\rho_i^l + \sum_{j=1}^{n}\left(\rho'_j\right)^l\right], \tag{38}$$

with the initial condition $C(0) = 1$. Equation 38 requires more arithmetic operations than (26) to compute the normalization constant. But, it is more efficient to pre-compute the factors $\sum_{i=1}^{m}\rho_i^j + \sum_{l=1}^{n}\left(\rho'_l\right)^j$ for each i and j, and uses the non-recursive formula.

3.3. Response Times

The time spent by a job (also called a request) in a queue and its server in the SPs is called the average response time in SPs. The average response time of a task in node i in SPs ($1 \le i \le m$) is given by

$$\begin{aligned} R_{SP_i} &= \frac{E\left[L_i(N)\right]}{\lambda_0} = \frac{1}{\lambda_0}\sum_{l=1}^{N}\rho_i^l\frac{C(N-l)}{C(N)} \\ &= \frac{p_0}{\lambda}\sum_{l=1}^{N}\left(\frac{1}{p_0\mu_i}\right)^l\frac{C(N-l)}{C(N)}, \end{aligned} \tag{39}$$

since $\lambda_0 = \dfrac{\lambda}{p_0}$ and $\rho_i = \dfrac{1}{p_0\mu_i}$.

The average response time of a task in node j in EPs ($1 \le j \le n$) is given by

$$R_{EP_j} = \frac{E\left[L'_j(N)\right]}{\lambda_1} = \frac{1}{\lambda_0 p_1}\sum_{l=1}^{N}\left(\rho'_j\right)^l\frac{C(N-l)}{C(N)}$$

$$= \frac{p_0}{\lambda p_1} \sum_{l=1}^{N} \left(\frac{p_1}{p_0 \mu'_j} \right)^l \frac{C(N-l)}{C(N)}, \tag{40}$$

since $\lambda_1 = \lambda_0 p_1 = \dfrac{\lambda}{p_0} p_1$ and $\rho'_j = \dfrac{p_1}{p_0 \mu'_j}$.

The time spent by N jobs in all the queues and servers is called the total response time. The total response time of the system (SPs and EPs) is given by

$$\begin{aligned} R &= \sum_{i=1}^{m} R_{SP_i} + \sum_{j=1}^{n} R_{EP_j} \\ &= \frac{p_0}{\lambda} \left\{ \sum_{i=1}^{m} \sum_{l=1}^{N} \rho_i^l \frac{C(N-l)}{C(N)} + \sum_{j=1}^{n} \sum_{l=1}^{N} \left(\rho'_j\right)^l \frac{C(N-l)}{C(N)} \right\} \\ &= \frac{p_0}{\lambda} \left\{ \sum_{i=1}^{m} \sum_{l=1}^{N} \left(\frac{1}{p_0 \mu_i} \right)^l \frac{C(N-l)}{C(N)} + \sum_{j=1}^{n} \sum_{l=1}^{N} \left(\frac{p_1}{p_0 \mu'_j} \right)^l \frac{C(N-l)}{C(N)} \right\}. \end{aligned} \tag{41}$$

3.4. Waiting Times

The total waiting time at Q_{SP} is given by

$$\begin{aligned} E[W] &= \sum_{i=1}^{m} \left(R_{SP_i} - \frac{1}{\mu_i} \right) \\ &= \sum_{i=1}^{m} \left(\frac{p_0}{\lambda} \sum_{l=1}^{N} \left(\frac{1}{p_0 \mu_i} \right)^l \frac{C(N-l)}{C(N)} - \frac{1}{\mu_i} \right), \end{aligned} \tag{42}$$

where $\dfrac{1}{\mu_i}$ is the average service time SP_i $(1 \le i \le m)$. The total waiting time at Q_{EP} is given by

$$\begin{aligned} E\left[W'\right] &= \sum_{j=1}^{n} \left(R_{EP_j} - \frac{1}{\mu'_j} \right) \\ &= \sum_{j=1}^{n} \left(\frac{p_0}{\lambda p_1} \sum_{l=1}^{N} \left(\frac{p_1}{p_0 \mu'_j} \right)^l \frac{C(N-l)}{C(N)} - \frac{1}{\mu'_j} \right), \end{aligned} \tag{43}$$

where $\frac{1}{\mu'_j}$ is the average service time of EP_j $(1 \leq j \leq n)$. The total waiting time at both the queues (Q_{SP} and Q_{EP}), $E[W_s]$, is given by

$$\begin{aligned} E[W_s] &= \sum_{i=1}^{m} \left(\frac{p_0}{\lambda} \sum_{l=1}^{N} \left(\frac{1}{p_0 \mu_i} \right)^l \frac{C(N-l)}{C(N)} - \frac{1}{\mu_i} \right) \\ &+ \sum_{j=1}^{n} \left(\frac{p_0}{\lambda p_1} \sum_{l=1}^{N} \left(\frac{p_1}{p_0 \mu'_j} \right)^l \frac{C(N-l)}{C(N)} - \frac{1}{\mu'_j} \right). \end{aligned} \tag{44}$$

3.5. Utilizations

The steady-state probability of having all jobs served by the EPs is given by

$$\begin{aligned} p\left(0, 0, \ldots, 0, l'_1, l'_2, \ldots, l'_n\right) &= \frac{1}{C(N)} \prod_{j=1}^{n} \left(\rho'_j\right)^{l'_j} \\ &= \frac{1}{C(N)} \prod_{j=1}^{n} \left(\frac{p_1}{p_0 \mu'_j} \right)^{l'_j}, \end{aligned} \tag{45}$$

where $\sum_{j=1}^{n} l'_j = N$. The steady-state probability of having at least one job served by the SPs is given by

$$U_0 = 1 - p\left(0, \ldots, 0, l'_1, l'_2, \ldots, l'_n\right) = 1 - \frac{1}{C(N)} \prod_{j=1}^{n} \left(\frac{p_1}{p_0 \mu'_j} \right)^{l'_j}. \tag{46}$$

If $U_0 > 1 - U_0$, or equivalently, $U_0 > \frac{1}{2}$, we have more SP utilization. This indicates that the execution of the program is dominated by SPs. If $U_0 < \frac{1}{2}$, we have more EP utilization. In this case, the execution of the program is dominated by the EPs. When $U_0 = \frac{1}{2}$, the program execution is said to be balanced.

3.6. System Throughput

The average throughput of the i^{th} node in the SPs $(1 \leq i \leq m)$ is given by

$$E\left[T_i(N)\right] = \mu_i p_0 U_i(N) = \mu_i p_0 \rho_i \frac{C(N-1)}{C(N)}. \tag{47}$$

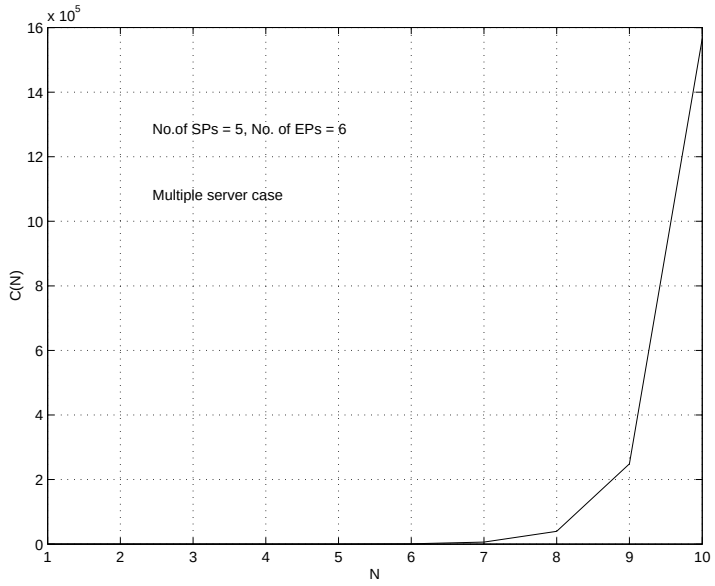

Figure 2: Normalization constant of a closed queuing network versus the total number of jobs

The average throughput of the j^{th} node in the EPs ($1 \leq j \leq n$) is given by

$$E\left[T'_j(N)\right] = \mu'_j p_0 U'_j(N) = \mu'_j p_0 \rho'_j \frac{C(N-1)}{C(N)}. \tag{48}$$

Now, the system throughput is provided only by the contribution of the SPs [11]. The system throughput is

$$\begin{aligned} E\left[T(N)\right] &= \sum_{i=1}^{m} E\left[T_i(N)\right] = \sum_{i=1}^{m} \mu_i p_0 \rho_i \frac{C(N-1)}{C(N)} \\ &= \sum_{i=1}^{m} \frac{C(N-1)}{C(N)}. \end{aligned} \tag{49}$$

4. Simulation Results

A simulation is performed for the closed queuing network model with multiple servers. The number of synchronization processors is 5 and

(m,n)	L_1	L_2	$L = L_1 + L_2$
(1,1)	6.606	2.399	9.006
(1,2)	5.2691	3.0829	8.352
(2,2)	5.8639	2.8571	8.72112
(3,2)	6.42106	2.7643	9.1854
(3,3)	6.24004	3.2577	9.49778
(3,4)	6.12623	3.62211	9.7483
(4,4)	6.6435	3.578	10.2216
(5,4)	7.0656	3.54803	10.6136
(5,5)	7.00727	3.84205	10.8493
(5,6)	6.96239	4.08607	11.04846

Table 1: Total number of jobs in SPs, EPs, and system

(m,n)	$U_i(N)$, $i = 1, 2, \ldots, m$	$U'_j(N)$, $j = 1, 2, \ldots, n$
(1,1)	(0.625)	(0.375)
(1,2)	(0.61349)	(0.22085, 0.1656)
(2,2)	(0.4167, 0.2083)	(0.25, 0.125)
(3,2)	(0.3246, 0.1948, 0.12987)	(0.2337, 0.1168)
(3,3)	(0.2751, 0.15287, 0.1179)	(0.24765, 0.1238, 0.08255)
(3,4)	(0.2137, 0.14246, 0.10958)	(0.25643, 0.1282, 0.08547, 0.0641)
(4,4)	(0.1454, 0.1247, 0.0969, 0.0872)	(0.26186, 0.1309, 0.0872, 0.0654)
(5,4)	(0.3851, 0.19255, 0.077, 0.035, 0.077)	(0.1155, 0.04621, 0.0385, 0.033)
(5,5)	(0.3912, 0.1956, 0.04891, 0.04891, 0.04891)	(0.11738, 0.04695, 0.03912, 0.0335, 0.02934)
(5,6)	(0.092267, 0.107229, 0.0762, 0.0748, 0.06612)	(0.23804, 0.11902, 0.07934, 0.05951, 0.0476, 0.0396)

Table 2: Actual Utilizations of SPs and EPs

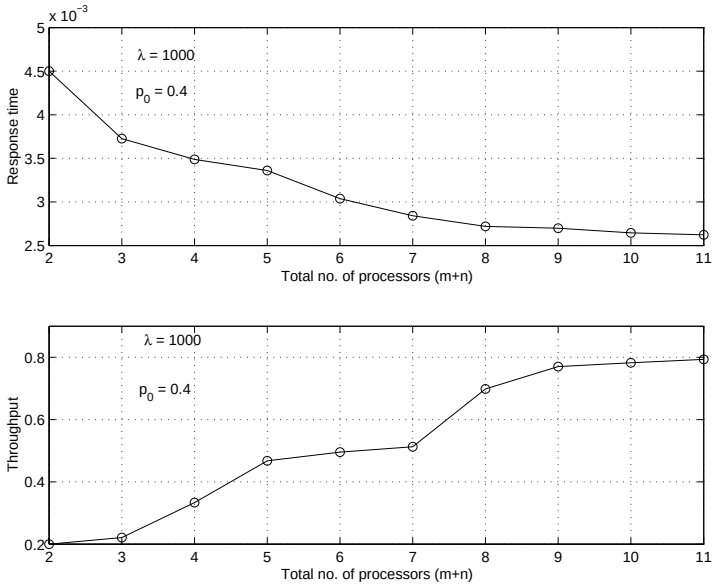

Figure 3: Response time and throughput versus the total number of processors (SPs and EPs)

the number of execution processors is chosen to be 6. The service rates μ_i and μ'_j, and the probability of jobs getting serviced at Q_{EP}, p_1, are appropriately chosen to have a constant probability of entering the new program path ($p_0 = 0.4$).

The estimated number of jobs in SPs (L_1), jobs in EPs (L_2) and jobs in the whole system ($L = L_1 + L_2$), are shown in Table 1. It is found that the estimated total number of jobs in the system are close to N (N is chosen as 10 in the simulation). The actual utilizations, $U_i(N)$ and $U'_j(N)$, and the relative utilizations, ρ_i and ρ'_j are tabulated in Tables 2 and 3 respectively.

The normalization constant is computed from (38) by pre-computing the utilizations at each of the nodes in SPs and EPs. The normalization constant is plotted against N, the number of jobs in the system in Figure 2. Knowing the normalization constant, the total system response time is computed from (41). The response time is plotted in Figure 3. For appropriate choices of the service rates of SPs and EPs, the response time is found to decrease as the total number of processors, $(m + n)$ increases.

(m,n)	$\rho_i,\ i = 1, 2, \ldots, m$	$\rho'_j,\ j = 1, 2, \ldots, n$
(1,1)	(3.125)	(1.875)
(1,2)	(2.778)	(1, 0.75)
(2,2)	(2.5, 1.25)	(1.5, 0.75)
(3,2)	(2.0833, 1.25, 0.833)	(1.5, 0.75)
(3,3)	(1.667, 0.9259, 0.7142)	(1.5, 0.75, 0.5)
(3,4)	(1.25, 0.833, 0.64102)	(1.5, 0.75, 0.5, 0.375)
(4,4)	(0.833, 0.7142, 0.555, 0.5)	(1.5, 0.75, 0.5, 0.375)
(5,4)	(2.5, 1.25, 0.5, 0.2272, 0.5)	(0.75, 0.3, 0.25, 0.21428)
(5,5)	(2.5, 1.25, 0.3125, 0.3125, 0.3125)	(0.75, 0.3, 0.25, 0.2143, 0.1875)
(5,6)	(0.5813, 0.6756, 0.4807, 0.4716, 0.4167)	(1.5, 0.75, 0.5, 0.375, 0.3, 0.25)

Table 3: Relative Utilizations of SPs and EPs

The system throughput is obtained from (49) and is plotted in Figure 3 against the total number of processors. The system throughput increases as the total number of processors increases. The arrival rate ($\lambda = 1000$) and $p_0 (= 0.4)$ are kept constant throughout the simulation. An increase in the number of functional units implies that there is a greater chance of finding a program ready to run (on the SPs) whenever the currently executing program incurs a page fault. Thus, increasing the number of processors will tend to increase the system throughput. This also decreases the system response time.

5. Conclusions

In this paper, we introduced a closed network of queues to model dataflow in a multi-processor system. The instruction streams are executed simultaneously (multi-threading) to minimize the loss of CPU cycles. A recursive algorithm is used to compute the normalization constant as a function of the degree of multiprogramming (number of active jobs) in the queuing model. The system performance measures are derived knowing the normalization constant. The normalization constant is plotted against the degree of multiprogramming. The number of jobs in SPs, EPs, and in the system are tabulated for different (m, n). The response time decreases as the number of processors increases, and the throughput increases as the number of processors increases. The system throughput

approaches an optimum value when the number of (synchronization + execution) processors is greater than or equal to nine. The optimum value for throughput is obtained by achieving a good balance of utilization between the pipelines (SPs and EPs). Thus, we have found the optimum number of functional units in a multi-threaded model to achieve higher instruction rates. Our model could be used for chip multiprocessors and super-scalar processors.

References

[1] V. Bhaskar, L. Joiner, Modelling scheduled dataflow architecture: An open queuing network model approach, *International Journal of Pure and Applied Mathematics*, **18**, No. 3 (2005).

[2] W. Gordon, G. Newell, *Closed Queuing Systems with Exponential Servers*, Oper. Res., **15** (1947).

[3] W. Grunewald, T. Ungerer, A multithreaded processor design for Distributed Shared Memory (DSM) system, In: *Proc. of the Intl Conference on Advances in parallel and distributed computing* (1997).

[4] K.M. Kavi, J. Arul, R. Giorgi, Execution and cache performance of the scheduled dataflow architecture, *Journal of Universal Computer Science* (October 2000).

[5] K.M. Kavi, R. Girogi, J. Arul, Scheduled dataflow: Execution paradigm, architecture, and performance evaluation, *IEEE Transactions on Computers*, **50**, No. 8 (August 2001), 834-846.

[6] L. Kleinrock, *Queuing systems, Volume II: Computer Applications*, John Wiley and Sons Inc. (1976).

[7] M. Lam, R. Wilson, Limits of control flow on parallelism, In: *Proc. of the 19th Intl Symposium on Computer Architecture (ISCA-19)* (May 1992), 46-57.

[8] S. Sakai, Architectural and software mechanisms for optimizing parallel computations, In: *Proc. of 1993 Intl Conference on Supercomputing* (July 1993).

[9] B. Shankar, L. Rho, W. Bohm, W. Najjar, Control of parallelism in multithreaded code, In: *Proc. of the Intl Conference on Parallel Architectures and Compiler Techniques (PACT-95)* (June 1995).

[10] M. Takesue, A unified resource management and execution control mechanism for dataflow machines, In: *Proc. 14th Int'l Symp. on Computer Architecture (ICSA-14)* (June 1987), 90-97.

[11] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, New Jersey, Prentice-Hall (1982).