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

Vidhyacharan Bhaskar, Kondo Hloindo Adjallah, Laurie L. Joiner

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

VIDHYACHARAN BHASKAR, KONDO H. ADJALLAH, AND LAURIE L. JOINER

Abstract. In this paper, a closed queuing network model with single 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.

1. INTRODUCTION

In open-queuing network models, 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 [1]. 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 [1]. However, such a network is said to be multiple-resources holding, since 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 [2]. In other words, an open-queuing network model is a light-load approximation.

2000 *Mathematics Subject Classification.* 68M20.

Key words and phrases. Synchronization and Execution processors, Multi-programming, Queue lengths, Response times, Utilizations, Normalization constant, Throughput.

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 [3]. Threads (sequence of tasks) are scheduled dynamically to be executed by the execution processors. The threads have a bounded execution time [3]. Our model also represents a distributed shared memory (DSM) system model in which all processors share a common address space [4]. 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) [5]. 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 [6], each thread is composed of conventional control-flow instructions. These instructions do not retain functional properties and need Write-after-Write (WAW) and Write-After-Read (WAR) dependencies [7].

In [8], 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 [8]. However, in the model developed in [9], the instructions within a thread retain functional properties of dataflow model and thus eliminate the need for complex hardware. Our work models the dataflow instructions appearing in [9].

Section II shows the block diagram of the closed queuing network of queues having single servers, and describes in detail the calculation of the normalization constant and utilizations. Section III discusses the performance measures such as queue lengths, response times, waiting times, throughput and utilizations for the closed queuing network model with single servers. Section IV discusses the simulation results of the queuing model. Finally, Section V presents the conclusions.

2. System Model: Single server

Figure 1 represents a good approximation to an open-queuing network model having a single-server 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_i, at node i among the synchronization processors, SPs (denoted SP_i), is the number of jobs (tasks) at node i (including any in service). The number of active jobs, k'_j, at node j among the execution processors, EPs (denoted EP_j), is the number of jobs (tasks) at node j (including any in service). For a closed queuing model with m servers for SPs and n servers for EPs, N is the total number of tasks

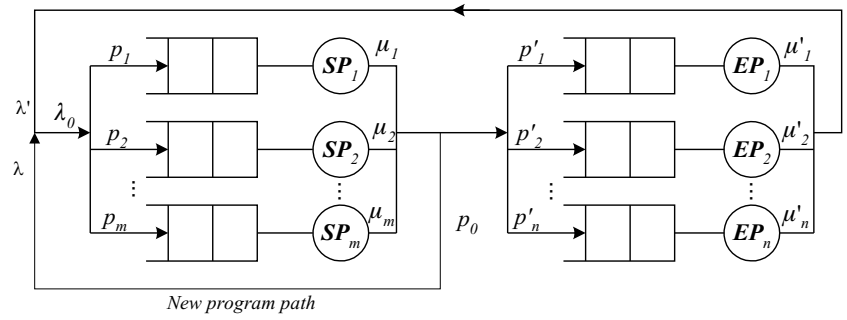

FIGURE 1. Single server - Closed queuing model with feedback

at the m SPs and the n EPs, or $N = k_1 + k_2 + \ldots + k_m + k'_1 + k'_2 + \ldots + k'_n$, where N is the total number of jobs in the system. The total number of active jobs, N, is called the degree of multi-programming [1].

The total number of tasks in the system is restricted to N due to the constraints of finite main memory [10]. 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 [10].

Let the external arrival rate be λ, arrival rate to all SPs be λ_0, and the arrival rate to each EP be λ'_j $(1 \leq j \leq n)$. The total arrival rate to all SPs is equal to sum of the external arrival rate and the arrival rate to all EPs. Hence,

$$\begin{aligned} \lambda_0 &= \lambda + \lambda' \\ &= \lambda + \sum_{j=1}^{n} \lambda'_j. \end{aligned} \tag{1}$$

The arrival rate to each of the EPs is

$$\lambda'_j = \lambda_0 p'_j \quad \forall\, 1 \leq j \leq n, \tag{2}$$

where p'_j is the probability that a job is sent to server EP_j. Substituting (2) in (1) and rearranging,

$$\lambda_0 = \frac{\lambda}{1 - \sum_{j=1}^{n} p'_j}. \tag{3}$$

Also,

$$p_0 + \sum_{j=1}^{n} p'_j = 1. \tag{4}$$

So,

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

The arrival rate to SP_i $(1 \leq i \leq m)$ is

$$\begin{aligned} \lambda_i &= \lambda_0 p_i \\ &= \frac{\lambda}{p_0} p_i. \end{aligned} \tag{6}$$

The arrival rate to EP_j $(1 \leq j \leq n)$ is

$$\lambda'_j = \frac{\lambda}{p_0} p'_j. \tag{7}$$

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

$$\gamma_i = \frac{\lambda_i}{\mu_i} = \frac{\lambda p_i}{p_0 \mu_i}, \tag{8}$$

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

$$\gamma'_j = \frac{\lambda'_j}{\mu'_j} = \frac{\lambda p'_j}{p_0 \mu'_j}. \tag{9}$$

Consider the closed network of queues with single server shown in Figure 1. The state of the network is the number of tasks at each server in the network. It can be represented by an $(m+n)$-tuple vector, $s = (k_1, k_2, \ldots, k_m, k'_1, k'_2, \ldots, k'_n)$.

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 [1] (assuming that $0 < p_i < 1 \; \forall \, i = 1, 2, \ldots, m$, and $0 < p'_j < 1 \; \forall \, j = 1, 2, \ldots, n$).

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

$$\mathbf{T} = \begin{bmatrix} p_0p_1 & p_0p_2 & \cdots & p_0p_m & p'_1 & p'_2 & \cdots & p'_n \\ p_0p_1 & p_0p_2 & \cdots & p_0p_m & p'_1 & p'_2 & \cdots & p'_n \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_0p_1 & p_0p_2 & \cdots & p_0p_m & p'_1 & p'_2 & \cdots & p'_n \\ p_1 & p_2 & \cdots & p_m & 0 & 0 & \cdots & 0 \\ p_1 & p_2 & \cdots & p_m & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_1 & p_2 & \cdots & p_m & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{10}$$

Notice that all rows sum to 1, i.e., $\sum_{i=1}^{m} p_i = 1$ and $p_0 \sum_{i=1}^{m} p_i + \sum_{j=1}^{n} p'_j = p_0 + \sum_{j=1}^{n} p'_j = 1$.

The DTMC is finite, and if we assume that $0 < p_i \leq 1 \; \forall \; i = 1, 2, \ldots, m$ and $0 < p'_j \leq 1 \; \forall \; j = 1, 2, \ldots, n$, then the DTMC can be shown to be irreducible and periodic. Then, the relative visit count vector $\underline{\mathbf{v}} = (v_1, v_2, \ldots, v_m, v'_1, \ldots, v'_n)$ can be obtained by solving the system of equations,

$$\underline{\mathbf{v}} = \underline{\mathbf{v}}\mathbf{T}. \tag{11}$$

If we observe the system for a real-time interval of duration τ, then $v_i\tau$ can be interpreted to be the average number of visits to node i in that interval [1].

The terms $v_i\tau$ and $v'_j\tau$ represent the average number of visits to node i, $i = 1, 2, \ldots, m$ and node j, $j = 1, 2, \ldots, n$. In this sense, v_i can be thought of as a relative visit count to node i in the SPs and v'_j as a relative visit count to node j in the EPs. Thus, v_i and v'_j represent the relative throughputs of nodes i and j respectively. For the network of queues shown in Figure 1, (11) becomes

$$[v_1\, v_2\, \ldots v_m\, v'_1\, v'_2\, \ldots v'_n] = [v_1\, v_2\, \ldots v_m\, v'_1\, v'_2\, \ldots\, v'_n]\, \mathbf{T}. \tag{12}$$

The system of linear equations represented by (12) is

$$\begin{aligned}
v_1 &= v_1p_0p_1 + v_2p_0p_1 + \ldots + v_mp_0p_1 + v'_1p_1 + v'_2p_1 + \ldots + v'_np_1 \\
&= p_0p_1 \sum_{i=1}^{m} v_i + p_1 \sum_{j=1}^{n} v'_j \\
&\vdots \quad\quad \vdots \\
v_m &= p_0p_m \sum_{i=1}^{m} v_i + p_m \sum_{j=1}^{n} v'_j \\
v'_1 &= p'_1 \sum_{i=1}^{m} v_i \\
&\vdots \quad\quad \vdots \\
v'_n &= p'_n \sum_{i=1}^{m} v_i
\end{aligned} \tag{13}$$

It is clear that only $(m+n-1)$ of the above $(m+n)$ equations are independent, since the sum of the first m equations is equal to the sum of the last n equations. Thus, this over-determined set of equations can be solved by choosing v_1 to be any real value that will aid us in our computations. The usual choices for v_1 are $\dfrac{1}{p_0}$, μ_1, or 1. Let us choose $v_1 = 1$. Adding the last n equations in

(13), $\sum_{j=1}^{n} v'_j = \sum_{j=1}^{n} p'_j \sum_{i=1}^{m} v_i = (1-p_0)\sum_{i=1}^{m} v_i$. Substituting this relation in (13) and rearranging, the relative throughput of node i $(2 \leq i \leq m)$ and node j $(1 \leq j \leq n)$ are

$$\begin{aligned} v_2 &= \frac{p_2}{p_1} \\ &\vdots \quad\quad \vdots \\ v_m &= \frac{p_m}{p_1} \\ v'_1 &= \frac{p'_1}{p_1} \\ &\vdots \quad\quad \vdots \\ v'_n &= \frac{p'_n}{p_1}. \end{aligned} \tag{14}$$

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

$$\rho_i = \frac{v_i}{\mu_i} = \frac{p_i}{p_1 \mu_i}. \tag{15}$$

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

$$\rho'_j = \frac{v'_j}{\mu'_j} = \frac{p'_j}{p_1 \mu'_j}. \tag{16}$$

Substituting (15) and (16) in the expression for steady-state probability [11]

$$\begin{aligned} p\left(k_1, k_2, \ldots, k_m, k'_1, k'_2, \ldots, k'_n\right) &= \frac{1}{C(N)} \prod_{i=1}^{m} \rho_i^{k_i} \prod_{j=1}^{n} \left(\rho'_j\right)^{k'_j} \\ &= \frac{1}{C(N)} \prod_{i=1}^{m} \left(\frac{p_i}{p_1 \mu_i}\right)^{k_i} \prod_{j=1}^{n} \left(\frac{p'_j}{p_1 \mu'_j}\right)^{k'_j}, \end{aligned} \tag{17}$$

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 [1]

$$C(N) = \sum_{s \in I} \prod_{i=1}^{m} \rho_i^{k_i} \prod_{j=1}^{n} \left(\rho'_j\right)^{k'_j}, \tag{18}$$

where the state space is

$$I = \left\{ \left(k_1, k_2, \ldots, k_m, k'_1, k'_2, \ldots, k'_n\right) | k_i \geq 0, k'_j \geq 0 \; \forall \; i, j \text{ and } \sum_{i=1}^{m} k_i + \sum_{j=1}^{n} k'_j = N \right\}, \tag{19}$$

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 (18) is very expensive and numerically unstable because the number of states grows exponentially with the number of customers and the number of service centers [1]. It is therefore necessary to derive computationally stable and efficient algorithms to obtain the value of the normalization constant, $C(N)$.

Let us consider the following polynomial in z.

$$\begin{aligned} G(z) &= \prod_{i=1}^{m} \frac{1}{1-\rho_i z} \prod_{j=1}^{n} \frac{1}{1-\rho'_j z} \\ &= \left(1+\rho_1 z+(\rho_1 z)^2+\ldots\right) \times \ldots \times \left(1+\rho_m z+(\rho_m z)^2+\ldots\right) \\ &\quad \times \left(1+\rho'_1 z+(\rho'_1 z)^2+\ldots\right) \times \ldots \times \left(1+\rho'_n z+(\rho'_n z)^2+\ldots\right). \end{aligned} \tag{20}$$

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

$$G(z) = \sum_{N=0}^{\infty} C(N) z^N = C(0) + C(1)z + C(2)z^2 + \ldots \tag{21}$$

where $C(0)$ is defined to be equal to unity [1]. Since $C(N)$ is not a probability, $G(z)$ is not necessarily equal to unity. The polynomial $G(z) \geq 0 \;\forall\; z$. In order to derive a recursive relation for computing $C(N)$, we define

$$\begin{aligned} G_{i,j}(z) &= \prod_{k=1}^{i} \frac{1}{(1-\rho_k z)} \prod_{l=1}^{j} \frac{1}{(1-\rho'_l z)} \\ &= \frac{1}{(1-\rho_1 z)\,(1-\rho_2 z)\ldots(1-\rho_i z)\,(1-\rho'_1 z)\,(1-\rho'_2 z)\ldots\left(1-\rho'_j z\right)}, \end{aligned} \tag{22}$$

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

$$G_{i,j}(z) = \sum_{l=0}^{\infty} C_{i,j}(l) z^l \tag{23}$$

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

$$G_{1,0}(z) = \frac{1}{1-\rho_1 z}. \tag{24}$$

The generalized expression is

$$G_{i,j}(z) = G_{i-1,j}(z)\frac{1}{1-\rho_i z}. \tag{25}$$

Now, (25) can be rewritten as

$$\begin{aligned} G_{i,j}(z)\,(1-\rho_i z) &= G_{i-1,j}(z) \\ \Rightarrow \quad G_{i,j}(z) &= \rho_i z G_{i,j}(z) + G_{i-1,j}(z) \\ \Rightarrow \quad \sum_{l=0}^{\infty} C_{i,j}(l) z^l &= \sum_{l=0}^{\infty} \rho_i z C_{i,j}(l) z^l + \sum_{l=0}^{\infty} C_{i-1,j}(l) z^l. \end{aligned} \tag{26}$$

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

$$C_{i,j}(l) = \rho_i C_{i,j}(l-1) + C_{i-1,j}(l), \tag{27}$$

where $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, N$. The initialization is obtained from (23) and (24) as

$$C_{1,0}(l) = \rho_1^l \;\forall\, l = 0, 1, \ldots, N. \tag{28}$$

Also from (22), we have the coefficient of z^0 in $G_{i,j}(z)$ as unity. Hence,

$$C_{i,j}(0) = 1 \;\; \forall\, i = 1, 2, \ldots, m \;\text{ and } \forall\, j = 1, 2, \ldots, n. \tag{29}$$

2.2. **Utilization of the i^{th} device.** Consider a slight modification to the generating function $G(z)$ denoted by $H_i(z)$:

$$\begin{aligned} H_i(z) &= \left(\prod_{j=1(j\neq i)}^{m} \frac{1}{1-\rho_j z}\right)\left(\prod_{k=1}^{n} \frac{1}{1-\rho'_k z}\right)\left(\frac{1}{1-\rho_i z} - 1\right) \\ &= \left(1 + \rho_1 z + (\rho_1 z)^2 + \ldots\right) \times \ldots \times \left(1 + \rho_{i-1} z + (\rho_{i-1} z)^2 + \ldots\right) \\ &\quad \times \left(\rho_i z + (\rho_i z)^2 + \ldots\right)\left(1 + \rho_{i+1} z + (\rho_{i+1} z)^2 + \ldots\right) \times \ldots \\ &\quad \times \left(1 + \rho_m z + (\rho_m z)^2 + \ldots\right)\left(1 + \rho'_1 z + (\rho'_1 z)^2 + \ldots\right) \times \ldots \\ &\quad \times \left(1 + \rho'_n z + (\rho'_n z)^2 + \ldots\right). \end{aligned} \tag{30}$$

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 of the form $\rho_1^{k_1}\rho_2^{k_2}\ldots\rho_m^{k_m}\,(\rho'_1)^{k'_1}\,(\rho'_2)^{k'_2}\ldots(\rho'_n)^{k'_n}$, 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 (17), we see that the coefficient of z^N in $G(z)$ yields the marginal probability $P(N_i \geq 1)$, which is exactly the utilization $U_i(N)$. So,

$$H_i(z) = \frac{G(z)\left[\frac{1}{1-\rho_i z} - 1\right]}{\frac{1}{1-\rho_i z}} = \rho_i z G(z). \tag{31}$$

From (31), 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

$$U_i(N) = \rho_i \frac{C(N-1)}{C(N)} = \left(\frac{p_i}{p_1 \mu_i}\right) \frac{C(N-1)}{C(N)} \tag{32}$$

as the utilization of the i^{th} device among the SPs when there are N jobs in the system. Here, ρ_i is the relative utilization of node i $(1 \leq i \leq m)$.

Similarly, the utilization of the j^{th} device among the EPs when there are N jobs in the system is given by

$$U'_j(N) = \rho'_j \frac{C(N-1)}{C(N)} = \left(\frac{p'_j}{p_1 \mu'_j}\right) \frac{C(N-1)}{C(N)}. \tag{33}$$

Here, ρ'_j is the relative utilization of device j $(1 \leq j \leq n)$. The ratio of the actual utilizations of SPs and EPs when there are N jobs in the system is given by $\dfrac{U_i(N)}{U'_j(N)}$. From (32) and (33), the ratio of actual utilizations is

$$\frac{U_i(N)}{U'_j(N)} = \frac{\rho_i}{\rho'_j} = \left(\frac{\mu'_j}{\mu_i}\right)\left(\frac{p_i}{p'_j}\right). \tag{34}$$

Equation 34 explains the reason for calling ρ_i and ρ'_j "relative utilizations".

3. Single-server: Performance Measures

3.1. **Queue Lengths.** The probability that there are k or more jobs at node i is given by [1]

$$P\left(N_i \geq k\right) = \rho_i^k \frac{C(N-k)}{C(N)}. \tag{35}$$

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

$$E\left[L_i(N)\right] = \sum_{l=1}^{N} \rho_i^l \frac{C\left(N-l\right)}{C(N)}. \tag{36}$$

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

$$E\left[L'_j(N)\right] = \sum_{l=1}^{N} \left(\rho'_j\right)^l \frac{C(N-l)}{C(N)}. \tag{37}$$

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

$$\begin{aligned} L &= \sum_{i=1}^{m} E\left[L_i(N)\right] + \sum_{j=1}^{n} E\left[L'_j(N)\right] \\ &= \sum_{i=1}^{m}\sum_{l=1}^{N} \rho_i^l \frac{C(N-l)}{C(N)} + \sum_{j=1}^{n}\sum_{l=1}^{N} \left(\rho'_j\right)^l \frac{C(N-l)}{C(N)} \\ &= \frac{1}{C(N)} \left\{ \sum_{i=1}^{m}\sum_{l=1}^{N} \left(\frac{p_i}{p_1\mu_i}\right)^l C(N-l) + \sum_{j=1}^{n}\sum_{l=1}^{N} \left(\frac{p'_j}{p_1\mu'_j}\right)^l C(N-l) \right\}. \end{aligned} \tag{38}$$

3.2. **Normalization Constant.** Cross-multiplying in (38), 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

$$C(N) = \frac{1}{N} \sum_{l=1}^{N} C(N-l) \left[\sum_{i=1}^{m} \rho_i^l + \sum_{j=1}^{n} \left(\rho'_j\right)^l \right] \tag{39}$$

with the initial condition $C(0) = 1$. Equation 39 requires more arithmetic operations than (27) 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 use 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

$$R_{SP_i} = \frac{E\left[L_i(N)\right]}{\lambda_i} = \frac{1}{\lambda_0 p_i} \sum_{l=1}^{N} \rho_i^l \frac{C(N-l)}{C(N)} = \frac{p_0}{\lambda p_i} \sum_{l=1}^{N} \left(\frac{p_i}{p_1\mu_i}\right)^l \frac{C(N-l)}{C(N)}, \tag{40}$$

since $\lambda_i = \lambda_0 p_i \ \forall\ i = 1, 2, \ldots m$, $\rho_i = \dfrac{p_i}{p_1\mu_i}$ and $\lambda_0 = \dfrac{\lambda}{p_0}$.

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

$$R_{EP_j} = \frac{E\left[L'_j(N)\right]}{\lambda'_j} = \frac{1}{\lambda_0 p'_j} \sum_{l=1}^{N} \left(\rho'_j\right)^l \frac{C(N-l)}{C(N)} = \frac{p_0}{\lambda p'_j} \sum_{l=1}^{N} \left(\frac{p'_j}{p_1\mu'_j}\right)^l \frac{C(N-l)}{C(N)}, \tag{41}$$

since $\lambda'_j = \lambda_0 p'_j \ \forall\ j = 1, 2, \ldots n$ and $\rho'_j = \dfrac{p'_j}{p_1\mu'_j}$.

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

$$\begin{aligned} R &= \sum_{i=1}^{m} R_{SP_i} + \sum_{j=1}^{n} R_{EP_j} \\ &= \frac{p_0}{\lambda}\left\{\sum_{i=1}^{m}\sum_{l=1}^{N}\left(\frac{\rho_i^l}{p_i}\right)\frac{C(N-l)}{C(N)} + \sum_{j=1}^{n}\sum_{l=1}^{N}\frac{\left(\rho_j'\right)^l}{p_j'}\frac{C(N-l)}{C(N)}\right\} \\ &= \frac{p_0}{\lambda}\left\{\sum_{i=1}^{m}\sum_{l=1}^{N}\left(\frac{p_i}{p_1\mu_i}\right)^l\frac{C(N-l)}{C(N)} + \sum_{j=1}^{n}\sum_{l=1}^{N}\left(\frac{p_j'}{p_1\mu_j'}\right)^l\frac{C(N-l)}{C(N)}\right\}. \end{aligned} \tag{42}$$

3.4. **Waiting times.** The total waiting time at all queues in the SPs is given by

$$E[W] = \sum_{i=1}^{m}\left(R_{SP_i} - \frac{1}{\mu_i}\right) = \sum_{i=1}^{m}\left(\frac{p_0}{\lambda p_i}\sum_{l=1}^{N}\left(\frac{p_i}{p_1\mu_i}\right)^l\frac{C(N-l)}{C(N)} - \frac{1}{\mu_i}\right), \tag{43}$$

where $\dfrac{1}{\mu_i}$ is the service time of SP_i $(1 \le i \le m)$. The total waiting time at all queues in the EPs is given by

$$\begin{aligned} E[W'] &= \sum_{j=1}^{n}\left(R_{EP_j} - \frac{1}{\mu_j'}\right) \\ &= \sum_{j=1}^{n}\left(\frac{p_0}{\lambda p_j'}\sum_{l=1}^{N}\left(\frac{p_j'}{p_1\mu_j'}\right)^l\frac{C(N-l)}{C(N)} - \frac{1}{\mu_j'}\right), \end{aligned} \tag{44}$$

where $\dfrac{1}{\mu_j'}$ is the service time of EP_j $(1 \le j \le n)$. The total waiting time in all the queues increases with N. The total waiting time in the system (SPs and EPs), $E[W_s]$, is given by

$$E[W_s] = \sum_{i=1}^{m}\left(\frac{p_0}{\lambda p_i}\sum_{l=1}^{N}\rho_i^l\frac{C(N-l)}{C(N)} - \frac{1}{\mu_i}\right) + \sum_{j=1}^{n}\left(\frac{p_0}{\lambda p_j'}\sum_{l=1}^{N}\left(\rho_j'\right)^l\frac{C(N-l)}{C(N)} - \frac{1}{\mu_j'}\right), \tag{45}$$

where $\rho_i = \dfrac{p_i}{p_1\mu_i}$ and $\rho_j' = \dfrac{p_j'}{p_1\mu_j'}$.

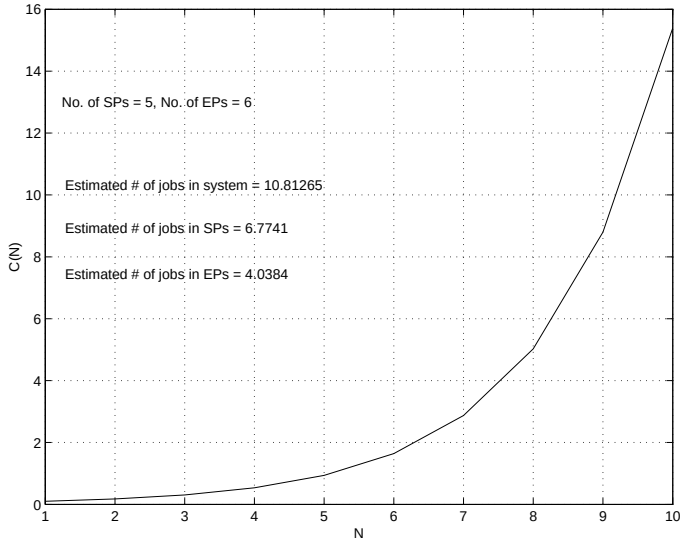

FIGURE 2. Normalization constant of a closed queuing network versus the total number of jobs

3.5. **Utilizations.** The steady-state probability of having all jobs served by the EPs is given by

$$p\left(0,0,\ldots,0,l_1',l_2',\ldots,l_n'\right)=\frac{1}{C(N)}\prod_{j=1}^{n}\left(\rho_j'\right)^{l_j'}=\frac{1}{C(N)}\prod_{j=1}^{n}\left(\frac{p_j'}{p_1\mu_j'}\right)^{l_j'}, \tag{46}$$

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

$$U_0=1-p\left(0,\ldots,0,l_1',l_2',\ldots,l_n'\right)=1-\frac{1}{C(N)}\prod_{j=1}^{n}\left(\frac{p_j'}{p_1\mu_j'}\right)^{l_j'}. \tag{47}$$

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 $(1 \le i \le m)$ is given by

$$E\left[T_i(N)\right]=\mu_i p_0 U_i(N)=\mu_i p_0 \rho_i \frac{C(N-1)}{C(N)}. \tag{48}$$

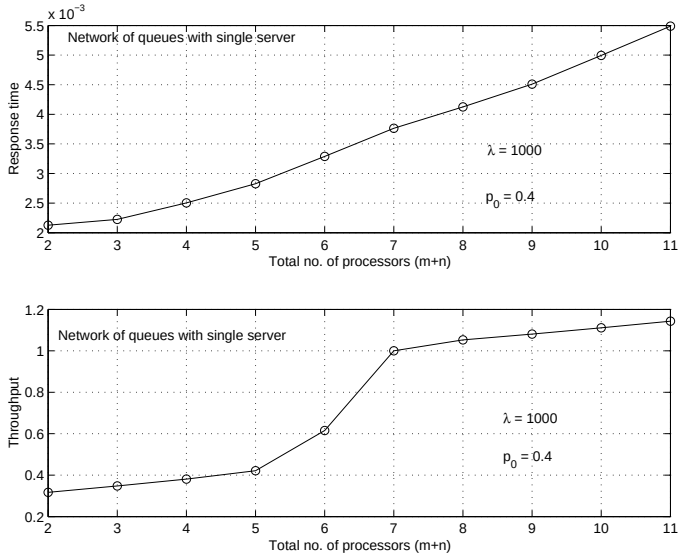

FIGURE 3. Response time and throughput versus the total number of processors (SPs and EPs)

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

$$E\left[T'_j(N)\right] = \mu'_j p_0 U'_j(N) = \mu'_j p_0 \rho'_j \frac{C(N-1)}{C(N)}. \tag{49}$$

Now, the system throughput is provided only by the contribution of the SPs [1]. The system throughput is given by

$$E\left[T(N)\right] = \sum_{i=1}^{m} E\left[T_i(N)\right] = \sum_{i=1}^{m} \mu_i p_0 \rho_i \frac{C(N-1)}{C(N)} = \sum_{i=1}^{m} \frac{p_i p_0}{p_1} \frac{C(N-1)}{C(N)}. \tag{50}$$

4. Simulation Results

A simulation is performed for the queuing model. The number of synchronization processors (m) is chosen to be equal to 5, and the number of execution processors (n) is chosen to be equal to 6. The service rates, μ_i and μ'_j, and the probability of getting serviced, p_i ($1 \leq i \leq m$) at SPs and p'_j ($1 \leq j \leq n$) at EPs, are appropriately chosen such that there is 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), as the total number of processors, $(m + n)$, increases.

TABLE 1. Total number of jobs in SPs, EPs, and system

(m,n)	L_1	L_2	$L = L_1 + L_2$
(1,1)	5.02496	0.17475	5.19971
(1,2)	10.0499	0.16733	10.2172
(2,2)	3.3978	2.2136	5.6114
(3,2)	2.7293	3.3204	6.04986
(3,3)	4.0939	2.9077	7.0016
(3,4)	5.4585	2.64335	8.10189
(4,4)	4.78375	3.52447	8.30823
(5,4)	4.51612	4.40559	8.92172
(5,5)	5.64516	4.150641	9.7958
(5,6)	6.77419	4.0346	10.81265

The normalization constant is computed from (39) by pre-computing the utilizations at each of the nodes in SPs and EPs. The normalization constant is plotted against N, the total number of jobs in the system in Figure 2. Knowing the normalization constant, the total system response time is computed from (42).

The system throughput is obtained from (50) 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. In Figure 3, the response time is also plotted against the total number of processors. As the total number of processors increases, the response time increases for a closed queuing network model. An increase in the number of functional units implies that additional main memory has to be purchased. Some programs (jobs) may be allowed to execute in only the address space allotted to them, especially in systems employing paged virtual memory. When a page fault occurs, additional memory is needed for a program. With the increased frequency of page faults, the system response time increases. On the other hand, an increased 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.

5. CONCLUSIONS

In this paper, we have 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 number of jobs in SPs, EPs, and in the system are tabulated for different

(m, n). The response times and throughput are found to increase with the total number of processors. The system throughput approaches an optimum value when the number of (synchronization + execution) processors is greater than or equal to ten. 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] K. Trivedi, *Probability & Statistics with Reliability, Queuing and Computer Science Applications*. New Jersey: Prentice-Hall, 1982.

[2] V. Bhaskar and L. Joiner, "Modelling scheduled dataflow architecture: An open queuing network model approach," *International Journal of Pure and Applied Mathematics*, vol. 18, no. 3, 2005.

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

[4] W. Grunewald and T. Ungerer, "A multithreaded processor design for Distributed Shared Memory (DSM) system," *Proc. of the Intl Conference on Advances in Parallel and Distributed Computing*, 1997.

[5] M. Lam and R. Wilson, "Limits of control flow on parallelism," *Proc. of the 19th Intl Symposium on Computer Architecture (ISCA-19)*, pp. 46–57, May 1992.

[6] S. Sakai, "Architectural and software mechanisms for optimizing parallel computations," *Proc. of 1993 Intl Conference on Supercomputing*, July 1993.

[7] K. M. Kavi, J. Arul, and R. Giorgi, "Execution and cache performance of the scheduled dataflow architecture," *Journal of Universal Computer Science*, Oct. 2000.

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

[9] K. M. Kavi, R. Girogi, and J. Arul, "Scheduled dataflow: Execution paradigm, architecture, and performance evaluation," *IEEE Transactions on Computers*, vol. 50, no. 8, pp. 834–846, Aug. 2001.

[10] L. Kleinrock, *Queuing Systems, Volume II: Computer Applications*. John Wiley & Sons Inc., 1976.

[11] W. Gordon and G. Newell, *Closed Queuing Systems with Exponential Servers*. Oper. Res. 15, 1947.

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

Institute of Computer Science and Engineering of Troyes, Universite de Technologie de Troyes, 12 Rue Marie Curie, 10010 Troyes Cedex, FRANCE
E-mail address: Kondo.Adjallah@utt.fr

Department of Electrical & Computer Engineering, University of Alabama in Huntsville, Huntsville, AL 35899 USA
E-mail address: ljoiner@ece.uah.edu