Computer Communications 31 (2008) 4098–4106

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

# A hybrid open queuing network model approach for multi-threaded dataflow architecture

Vidhyacharan Bhaskar [a,*], Kondo Hloindo Adjallah [b]

[a] Department of Electronics and Communication Engineering, SRM University, Kattankulathur, Kancheepuram District 603203, Tamilnadu, India
[b] Charles Delaunay Institute, Universite de Technologie de Troyes, 12 Rue Marie Curie, 10010 Troyes Cedex, France

ARTICLE INFO

*Article history:*
Received 24 April 2007
Received in revised form 19 August 2008
Accepted 20 August 2008
Available online 29 August 2008

*Keywords:*
Synchronization and execution processors
Multi-programming
Queue lengths
Response times
Throughput

ABSTRACT

Multi-threading has been proposed as an execution model for massively built parallel processors. Due to the large amount of potential parallelism, resource management is a critical issue in multi-threaded architecture. The challenge of multi-threading is to hide the latency by switching among a set of ready threads and thus to improve the processor utilization. Threads are dynamically scheduled to execute based on availability of data. In this paper, two hybrid open queuing network models are proposed. Two sets of processors: synchronization processors and execution processors exist. Each processor is modeled as a server serving a single-queue or multiple-servers serving a single-queue. Performance measures like response times, system throughput and average queue lengths are evaluated for both the hybrid models. The utilizations of the two models are derived and compared with each other. A mean value analysis is performed and different performance measures are plotted.

## 1. Introduction

Multi-threading is the ability of an operating system to execute different parts of a program, called threads, simultaneously. It is required to have the threads (a set of instructions) run at the same time without interfering with one another. A multi-threaded dataflow architecture is used in a hybrid closed queuing network model [1].

Dataflow means parallel execution. The dataflow model of computation only exposes the inherent parallelism and the parallelism can be exploited if multiple functional units or processing elements are available [2]. In this paper, threads are dynamically scheduled to execute based on the availability of data. A thread, after starting to execute, runs to completion without blocking and with a bounded execution time, because of which the threads must have a fixed execution time [3].

The dataflow architecture presented in this paper is called a scheduled dataflow architecture (SDF). In SDF, we have two kinds of processors. They are (i) synchronization processor (SP) and (ii) execution processor (EP). The SP is responsible for load/store operations. Data arrives at the SP in the form of threads from memory. The SP fetches the instructions from memory, schedules them, and sends them to the EP. It requires greater thread-level parallelism to achieve good performance, while super-scalar architectures require greater instruction level parallelism to achieve good performance [4]. The performance of the SDF scales better with a proper balance of workload among the functional units (SPs and EPs) [4].

Multi-threading achieves higher instruction rates on processors that contain multiple functional units (e.g., super-scalars and VLIW) or multiple processing elements (i.e., Chip Multiprocessors) [5–10].

It is necessary to find an appropriate multi-threaded model and implementation to achieve the best possible performance. An open queuing network model can be considered to model the SDF when the number of arrivals is small. In this paper, two hybrid open queuing network models depicting dataflow in SDF are presented. Section 2 describes the block diagram of the first hybrid model and discusses performance measures such as average queue length, response time, waiting time, visit ratios, utilizations and the simulation results. Section 3 describes the block diagram of the second hybrid model and discusses performance measures such as average queue length, response time, waiting time, visit ratios, utilizations and the simulation results. Section 4 provides a comparison between the utilization of the two networks. Finally, Section 5 presents the conclusions. In the Appendix, we present the state diagrams and steady-state balance equations for particular cases of hybrid queuing models, and the mean value analysis algorithm for the hybrid models in consideration.

## 2. Hybrid model 1

In Fig. 1, each synchronization processor (SP) is analogous to a server with a single-queue. The model contains $m$ SPs. Each

* Corresponding author. Tel.: +91 9884661184.
*E-mail addresses:* meetvidhyacharan@yahoo.com (V. Bhaskar), kondo.adjallah@utt.fr (K.H. Adjallah).

0140-3664/$ - see front matter Crown copyright © 2008 Published by Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2008.08.017

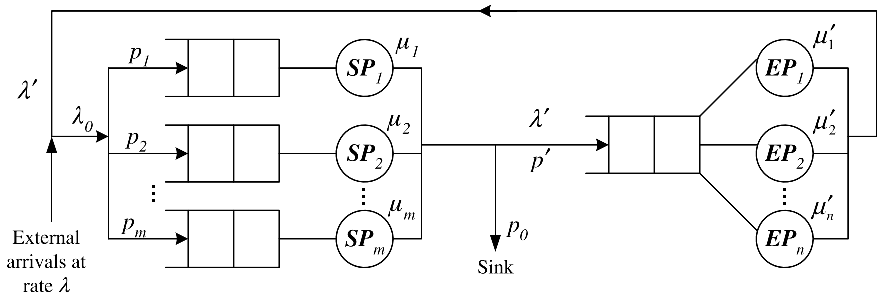

**Fig. 1.** Hybrid open queuing network model 1.

execution processor (EP) is modeled as a queuing network of $n$ servers denoted as $EP_j$, $(1 \leqslant j \leqslant n)$, serving a queue, $Q_{EP}$. Each queue in the network of $m$ queues has a server, $SP_i$ $(1 \leqslant i \leqslant m)$, and is denoted as $Q_{SP_i}$ $(1 \leqslant i \leqslant m)$.

The model shown in Fig. 1 is an open queuing network of $(m+1)$ queues. Each queue served by $SP_i$ $(1 \leqslant i \leqslant m)$ is M/M/1 in nature. The first M in the M/M/1 notation denotes that the arrivals to the queues are memoryless. The arrivals are random in nature and they form a Poisson process at a constant arrival rate. The second M denotes that the service times are memoryless. The service times have an exponential distribution. The queuing discipline is first come first served (FCFS) [11]. There are no limits on the size of the queue or on the population from which the jobs come.

Arrivals to the SP nodes occur from outside the network at a rate $\lambda$, or through feedback. Let $\mu_1, \mu_2, \ldots, \mu_m$ be the service rates of each of the SPs, and let $\mu'_1, \mu'_2, \ldots, \mu'_n$ be the service rates of each of the EPs. Similarly, let $\lambda_1, \lambda_2, \ldots, \lambda_m$ be the arrival rates to each of the SPs, and let $\lambda'$ be the arrival rate to $Q_{EP}$.

Let the total arrival rate to all SPs be $\lambda_0$. From Fig. 1,

$$\lambda' = \lambda_0 p', \tag{1}$$

where $p'$ is the probability of requesting service from $EP_j (1 \leqslant j \leqslant n)$. For steady-state to exist in the network, the total arrival rate to all SPs is the sum of the external arrival rate and that coming from feedback. So,

$$\lambda_0 = \lambda + \lambda' = \lambda + \lambda_0 p'. \tag{2}$$

Rearranging (2), we have

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

From Fig. 1, we also have

$$p_0 + p' = 1, \tag{4}$$

where $p_0$ is the probability that no service is requested from the EPs. Substituting (4) into (3), we have

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

Eq. (5) must hold good for steady-state to exist.

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

$$\lambda_i = \lambda_0 p_i = \frac{\lambda}{p_0} p_i, \tag{6}$$

where $p_i$ $(1 \leqslant i \leqslant m)$, is the probability of arrival at the $i^{th}$ queue, requesting service from $SP_i$. The utilization of each $SP_i$ $(1 \leqslant i \leqslant m)$ is

$$\rho_i = \frac{\lambda_i}{\mu_i} = \frac{\lambda p_i}{p_0 \mu_i}, \tag{7}$$

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

$$\rho'_j = \frac{\lambda'}{\mu'_j} = \frac{\lambda p'}{p_0 \mu'_j}. \tag{8}$$

The average number of times $Q_{SP_i}$ $(1 \leqslant i \leqslant m)$ is revisited is [11]

$$V_i = \frac{p_i}{p_0}. \tag{9}$$

The average number of times $Q_{EP}$ for server $EP_j$ $(1 \leqslant j \leqslant n)$ is revisited is

$$V'_j = \frac{p'}{p_0}. \tag{10}$$

## 2.1. Performance measures

The performance of the hybrid model can be measured by the average queue lengths, average waiting times, average response times, and the average number of jobs in the system. For the queues in the model which are M/M/1, the average queue length (number of jobs in service + number of jobs waiting to be served) in $SP_i$ $(1 \leqslant i \leqslant m)$ is

$$L_i = \frac{\rho_i}{1 - \rho_i} = \frac{\lambda p_i}{\mu_i p_0 - \lambda p_i}. \tag{11}$$

The following discussion holds good for queues in the model which are M/M/$n$/$\infty$. The M/M/$n$/$\infty$ queuing system at time $t$ is a birth–death process with the following parameters:

$$a_{k'} = \lambda', \quad k' \geqslant 0, \tag{12}$$

$$d_{k'} = \begin{cases} k'\mu' & 0 < k' < n \\ n\mu' & k' \geqslant n. \end{cases} \tag{13}$$

In (12), $a_{k'}$ is the arrival rate to the M/M/$n$/$\infty$ queuing system, and in (13), $d_{k'}$ is the departure rate from the M/M/$n$/$\infty$ queuing system. Since there are infinite number of waiting positions in this queue, we consider an infinite number of jobs arriving at this queue. The departure rate depends on the number of busy servers at a time. If there are $k' < n$ servers busy in the system, the average service (departure) rate is $d_{k'} = k'\mu'$, and if all the $n$ servers are busy in the system, $d_{k'} = n\mu'$. The departure parameter is state dependent. Note that the steady-state probability of having no jobs in the system is

$$\pi'_0 = \left[ \sum_{l=0}^{n-1} \frac{(n\rho')^l}{l!} + \frac{(n\rho')^n}{n!(1-\rho')} \right]^{-1}, \tag{14}$$

where $\rho' = \frac{\lambda'}{n\mu'} < 1$ is the traffic intensity of the M/M/$n$/$\infty$ queueing system (for EPs) and $\frac{1}{\mu'}$ is the average service time of the EPs [12].

The average queue length of the M/M/$n$/$\infty$ queuing system (for EPs) is [13]

$$L'_n = n\rho' + \frac{\rho'(n\rho')^n}{n!(1-\rho')^2} \pi'_0. \tag{15}$$

The average response time can be computed from Little's theorem [11]. The average response time in $SP_i$ $(1 \leqslant i \leqslant m)$ is

$$R_i = \frac{L_i}{\lambda_0 p_i} = \frac{p_0}{\mu_i p_0 - \lambda p_i}, \tag{16}$$