Computing A Matrix Transpose of Multithreading for Queueing Parallel in Matlab Programming

I.A. Ismail
Dean of Faculty of Computer and Informatics,
Misr International University,
Cairo, Egypt.

G.S. Mokaddis
Prof .of Statistics Mathematics
Department of Mathematics,
Faculty of Science , Ain Shams University, Cairo, Egypt.

Mariam K.Metry
Engineer software of AOI Science
Student P h.D ,Faculty of Science, Ain Shams University, Cairo, Egypt.

ABSTRACT
This paper describes a A matrix operation (vector and transpose) can be performed in queueing parallel model by using multithreading software are showing. Multithreading is useful in reducing the latency by switching among a set of threads in order to improve the processor utilization. Closed queueing network model is suitable for large number of job arrivals. The model is validated by comparison of analytical parallel and simulation result.

Keywords
Closed Queueing Network, parallel programming, simulation and computing.

1. INTRODUCTION
Most of recent scalable shared memory architectures typically provide different combinations of latency reducing and such as caching and multithreading (**Bhaskar.V , 2009**).Multithreading is used for hiding long memory latency in multiprocessor systems, and aims to increase system efficiency. A number of threads are allocated to a processing node which switches thread contexts according to some context switch policy, such as switch on cache misses, synchronization locks or explicit remote references (***coarse or block multithreading***); or switch on each instruction or each cycle (***fine multithreading***). Finely multithreaded processors potentially provide low context switch overhead, but require a large number of threads and a complex thread scheduling technique supported by hardware in order to achieve high efficiency of pipelined execution. On a single threaded load the utilization of a fine multithreaded processor dramatically falls off. Multithreading technique allowed the conclusion that a block multithreaded (**Greenberg. A. G., 1991)** and **(V.Vlassov,1996**). Mathematical methods and simulation are used to analyze various architectural solutions in a MTA (**Mean Thread Analysis**) design. A few attempts of mathematical evaluation of block multithreaded MTA have resulted in deterministic analytical models and queuing models (**Scott Graham.G, 2002**) and (**Zhao.Y, 2006**). A technique of analytical modelling of such MTA is mainly based on the consideration of a set of thread states and state transitions. A thread, during its life time, cyclically passes through four main states: switching, running, suspended and ready. As a consequence, a processing node while executing a set of threads cyclically passes through three states: switching, running, and idle. The efficiency of the architecture is evaluated as the ratio of the total running time to the sum of the total switching, running, and idle time of the processing node. The distribution time for each state of a thread, except for the ready state, can be assumed to be fixed (***deterministic models***) or random (***probabilistic models***).A first order approximation for MTA efficiency is a set of *n* threads with the following fixed timing parameters: context switching overhead C , remote memory latency L and thread run length R , which is the number of cycles between two consecutive remote references resulting in context switch. The model predicts linear dependence of MTA efficiency remote memory latency EL on the number of threads n , MTA is saturated at the saturation point N_S . After this point efficiency as a function of n becomes constant E_S :

$$\left.\begin{array}{l}\text{Efficiency of single threaded architecture,}\\ \text{STA is } \mathrm{E}_1 = R/(R+L),\\ \\ \text{Efficiency of muli threaded architecture,}\\ \text{MTA is } \mathrm{E}_n = (E_L, E_S),\\ \\ \text{Linear region of MTA efficiency is}\\ \mathrm{E}_L = nR/(R+L+C),\\ \\ \text{Saturaion point is}\\ \mathrm{N}_S = \{1/(\mathrm{R}+\mathrm{C})\}+1.\end{array}\right\} \quad (1)$$

A queueing model for multithreaded architecture is executing a set of statistically identical threads. The presented solution of a closed queueing network is a case study for a particular state diagram for a thread execution cycle with context switch on local memory misses. The model illustrates a usage of queueing theory for evaluation of MTA .

2. PARALLEL PROGRAMMING MODEL
2.1. Threads model
Multithreading is similar to multiprocessing programming the difference is that a multithreaded program has a single processor which manages multiple threads of control executing asynchronously (**Ricardo Bianchini,1996**). The threads library provides function calls to calls to create

threads, control threads, terminate threads, control access to shared data through locking mechanisms, generate events and wait for events.

2.2 Program transformations

We have identified the dependencies in a program, comes the issue of whether we can overcome the dependencies so that the program is queueing parallel execution.

- Forward Dependency.
- Backward Dependency.

2.2. A thread state diagram

We assume that the MTA have a typical structure including the processing element PE , a cache, and a main memory, part of which is shared. Assume that during its life time the thread cyclically passes through the following timed states:

- C : context switch, where the thread is being scheduled for execution. Thread run length of context switch takes CR cycles to activate a thread whose context resides in the processing element of the MTA , and the resident thread *CM* cycles to activate a thread from the main memory of the MTA (memory thread).
- r : run state, where the thread is executing during a time between two consecutive cache accesses.
- LC : cache memory access (cache latency).
- T : locality test, which is performed in the case of a cache miss to check if data resides in local or remote memory.
- *LM*: local memory access (local memory latency) which is performed in the case of a cache miss and if data is in the memory.
- *L*: Remote memory access a local memory miss (remote memory latency). The interval may include the time required to send a remote fetch request, communication round trip time, time needed to maintain cache coherency and load requested data to a register.
- t_r : ready state, where the thread is ready for execution. The duration of this state depends on behavior of other threads allocated to the node and does not need to be specified as an input parameter.

Assume also that a local memory miss may occur with probability P_{LM} (local memory miss ratio), and the probability of a cache miss is *PM* (cache miss ratio). illustrates thread state transitions. Execution of the newly reactivated thread C resumes from the state r , where the PE executes thread instructions passing through LC (cache access) and returning back to r if requested data is currently cached (probability $1-PM$). This loop continues until the cache misses with probability PM , and the thread passes to the locality-test state T where it checks locality of the requested address. In the case of a local memory miss (probability P_{LM}) the thread initiates a remote memory access and becomes suspended in the state L . In the case of a local memory hit (probability $1-P_{LM}$) the thread performs a local memory access LM and returns back to r .Assume that MTA executes a fixed number n of identical threads forever. In this case the MTA can be represented as a closed queuing network with a finite number n of circulating jobs (threads). The queueing network consists of a queue Q and two subnetworks called RN and LN . Each server of the queueing network corresponds to a thread state and is defined mean service time. For example, s the r state and is marked by the r service time. Assume that all timing parameters r , L , LM , LC , CR , CM and T have exponential distributions with corresponding means, and that the mean context switch time C is defined as:

$$C = \begin{cases} CR & ; \quad \text{if } n \le n_R, \\ CR \,.\, n_R + C_M \,.(n - n_R) & ; \quad \text{if } n > n_R, \end{cases} \tag{2}$$

where n_R is a number of contexts which may reside in the processing element of the MTA . It is difficult to analyze the queueing network because sub network RN can not serve more than one job at a time, i.e. not more than one thread can be active at any time. The behavior of the network with the above restriction can be described by the continuous time Markov chain. Each state of the chain is marked by a triple index (q, s, l) .

where:

- q : number of jobs in the queue Q , $i \in \{0,1,2,...,(n-1)\}$,
- s : the name (mean service time) of a server in the network where the job is located, $s \in \{C, r, LC, T, LM, l\}$,
- l : the number of jobs in sub network LN , $l \in \{0,1,2,...,n\}$.

The chain contains states and can be analytically explored using well known methods (**Ismail.A.I., 2000**)and (**Anoop Gupta,1991**). To define the limiting probabilities of states, the following system of equilibrium equations of our queueing network must be deduced as follows

$$0 = n\lambda_L P_{0,L,n} + \lambda_T P_{LM} P_{0,T,n-1},$$

$$0 = -(\lambda_C + \Phi) P_{i,C,n-i-1} + \Psi P_{i-1,C,n-1} + \Phi \lambda_T P_{LM} P_{i+1,T,n-i-2} + (1-\Psi) n \lambda_L P_{0,L,n},$$

$$0 = -(\lambda_r + \Phi)P_{i,r,n-i-1} + \Psi_{i-1,r,n-1} +$$
$$\lambda_C P_{i,C,n-i-1} +$$
$$\lambda_{LC}(1-P_M)P_{i,LC,n-i-1},$$

$$0 = -(\lambda_{LC} + \Phi)P_{i,LC,n-i-1} + \Psi P_{i-1,LC,n-i}$$
$$+ \lambda_{Lc} P_M P_{LC,n-i-1},$$

$$0 = -(\lambda_T + \Phi)P_{i,T,n-i-1} + \Psi P_{i-1,T,n-i}$$
$$+ \lambda_{Lc} P_M P_{LC,n-i-1}, \quad (3)$$

where, $\Phi = \Phi(n-i-1)\lambda_L, \Psi = \Psi(n-i)\lambda_L$,

$$\Phi = \begin{cases} 1 & ; \quad \text{if} \quad i < n\text{-}1, \\ 0 & ; \quad \text{if} \quad i = n\text{-}1. \end{cases}$$

$$\Psi = \begin{cases} 1 & ; \quad \text{if} \quad i > 0, \\ 0 & ; \quad \text{if} \quad i = 0. \end{cases}$$

$$\lambda_C = 1/C \quad , \quad \lambda_r = 1/r \ ,$$
$$\lambda_{LM} = 1/LM \ ,$$
$$\lambda_L = 1/L \quad ;$$
$$i = 0,1,\ldots,n-1.$$

The conservation relation is:

$$\sum_{i=0}^{n-1} P_{i,C,n-i-1} + \sum_{i-0}^{n-1} P_{i,r,n-i-1} +$$
$$\sum_{i=0}^{n-1} P_{i,LC,n-i-1} +$$
$$\sum_{i=0}^{n-1} P_{i,LM,n-i=1} +$$
$$\sum_{i=0}^{n-1} P_{i,T,n-i+1} + P_{o,L,n} = 1. \quad (4)$$

Since service in the r, LC, T *and* LM in servers of the RN subnetwork is useful work, the probability of having the thread in the servers is interpreted as efficiency of the MTA:

$$E_n = 1 - \sum_{i=0}^{n-1} P_{i,C,n-i-1} + P_{o,L,n}. \quad (5)$$

We have analyzed the chain computationally using the matlab environment. Nevertheless, it is possible to simplify the analytical solution of the queueing network using the following steps.

1. Considering the RN subnetwork as a closed queueing network with one circulating job (***thread***), determine the following:

- Throughput (service rate $1/(R+C)$) of the RN subnetwork, where R is run length, i.e., the number of cycles between two consecutive local memory misses in a thread (context switches in the PE_S).
- Probabilities of having the job in each servers P_S $s \in \{C, r, LC, T, LM\}$

2. Derive from the above information the utilization of the MTA in saturation, i.e., 3.assuming that a running thread is always available on each context switch.
3. Assuming that the closed queueing network consists of $M/M/1$ queueing system with one server (with service rate $1/(R+C)$) and $M/M/n$ queueing system with n servers (each with service rate $1/L$), determine the utilization of the MTA as a function of the number of threads circulating in the queueing network.

2.3 Mean thread analysis MTA

We shall first consider sub network RN as a closed queueing network (with one circulating job), in which the *T* server is connected to the C server. It contains five servers : C, r, LC, T , and LM , which are numbered $1,\ldots,5$ respectively. The fictitious job source is numbered 0 and located between T and C servers. A matrix of transition probabilities A_T , which is derived from the structure of the closed queueing network in **Equation (5)** is

$$A_T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & (1\text{-}P_{LM}) & 0 & P_{LM} & 0 \\ P_{LM} & 0 & 0 & 0 & 0 & (1\text{-}P_{LM}) \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

transformation coefficients $\alpha_i, i = 1,...,5$ each of which is a mean number of having a thread in the corresponding server and can be defined from the following system of equations.

$$(P_{ii} - 1).\alpha_i + \sum_{j=1, j\neq i}^{s} P_{ji}.\alpha_j = 0 \quad ; \quad i=1,...,5$$

The solution is =

$$\begin{cases} \alpha_1 = 1, \\ \alpha_2 = \alpha_3 = 1/(P_M P_{LM}), \\ \alpha_4 = 1/P_{LM}, \\ \alpha_5 = (1\text{-}P_{LM})/P_{LM}. \end{cases}$$

The state of a closed queueing network is represented by an exponentially distribution of m jobs in k nodes, and the number of statesisequal to $(m+k-1)!/(m!(k-1)!)$. Since we have assumed that one job $(m-1)$ circulates only .The probability of each state is the probability of having the job in the corresponding server. The state probabilities can be defined using product form analysis and Erlang's formulas:

$$\left.\begin{array}{ll} P_C &= C/D, \\ P_r &= \alpha_2 r/D, \\ P_{LC} &= \alpha_3 LC/D, \\ P_T &= \alpha_4 T/D, \\ P_{LM} &= \alpha_5 LM/D. \end{array}\right\},$$

(6)

where,

$D = C + \alpha_2 r + \alpha_3 LC + \alpha_4 T + \alpha_5 LM$ The probability of having the job in any queueing server of RN except of the C server is the utilization of the MTA

where, $E_s = 1 - P_C = R/(C+R)$. (7)

$$R = \frac{r + L_C + [T + L_M + (1-P_{LM})]P_M}{P_M P_{LM}}.$$

(8)

is thread run length.

Note that the mean run length R , from the above **Equations (7) and (8)** can be used in the deterministic (***fixed***).

2.4.1. Efficiency of the MTA

It can be shown that the mean response time of the RN sub network is $C + R$ .Assume that subnetwork RN is represented as a server with service rate $1/(R+C)$. The network includes two queueing systems, $M/M/1$ and $M/M/n$ numbered 1 and 2. The number of states is $n+1$, where the number of circulating jobs (threads) is n . Transformation coefficients for this queueing network are $\alpha_1 = 1, \alpha_2 = 1.$ Probabilities of states can be derived in the same way as for the RN network :

$$\forall (n_1, n_2) \Rightarrow n_1 = 0,...,n; n_2 = 0,...,n \ ; \quad such\ that\ n = n_1 + n_2.$$

It follows that

$$P(n_1, n_2) = (C+R)^{n_1}.\frac{L^{n_1}}{n_2!}.(\sum_{i=0}^{n}(C+R)^{n-i}\frac{L^i}{i!})^{-1}$$

The probability of having at least one thread in RN is $1 - P(0,n)$. Since a service in the r, LC, T and LM servers of the RN subnetwork is the product efficiency of the MTA :

$$E_n = (1 - P(0,n)),$$

$$E_s = [1 - \frac{\eta^n}{n!}(\sum_{i=0}^{n}\frac{\eta^i}{i!})]^{-1}\frac{R}{R+C},$$

(9)

where ,

$$\eta = L/(C+R).$$

The mean number of running threads nR (utilization of the RN subnetwork), the mean number of ready threads $n.r$ (mean number of jobs in the queue), the mean number of suspended threads ns (utilization of the LN servers), and the mean ready time tr (waiting time in the queue Q) can be calculated using queuing theory:

$$\left.\begin{aligned} n_R &= 1 - P(0,n), \\ n_r &= \sum_{i=2}^{n} (i-1)\, P(i, n-i), \\ n_s &= \sum_{i=1}^{n} i\; P(n-i, i), \\ t_r &= \frac{(C+R)\, n_r}{n_R}, \end{aligned}\right\} \tag{10}$$

where,

$$n = n_R + n_r + n_s .$$

2.4.2. Efficiency of the MTA with fixed number of L servers

The closed queueing network in contains the L servers which model remote accesses. The number of the L servers k , represents the number of remote requests which can be serviced in parallel. This number can be fixed and independent of the number of executed threads, n . If k is fixed and, then a state probability (the probability of having n_1 jobs in the RN subnetwork and jobs in then LN subnetwork) is

$$P(n_1, n_2) = (C+R)^{n_1} \frac{L^{n_2}}{\beta(n_2)} \left(\sum_{i=0}^{n} (C+R)^{n-i} \frac{L^i}{\beta(i)} \right)^{-1},$$

where,

$$\beta(x) = \begin{cases} x! & ; \quad \text{if} \quad x \le k, \\ k!\, k^{x-k} & ; \quad \text{if} \quad x > k. \end{cases}$$

The **Equation (10)** is valid but the **Equation (7)** is transformed to:

$$E(n>k) = [1 - P(o,n)] E_s =$$

$$1 - \frac{\eta^n}{k!\, k^{n-k}} \left(\sum_{i=0}^{k} \frac{\eta^i}{i!} + \sum_{i=k+1}^{n} \frac{\eta^i}{k!\, k^{i-k}} \right)^{-1} \tag{11}$$

The minimum number of L servers, required to achieve a desired efficiency of the MTA is given workload can be obtained by repetitive computations of the efficiency.

3. THE PERFORMANCE MEASURES OF QUEUEING PARALLEL COMPUTING

We can be described the performance measures of queueing parallel computing under MTA (**Scott Graham.G, 2002**).

- **Queue lengths**

The probability that there are k or more job i is given by

$$P(N_i) = \rho_i^k \frac{C(N-1)}{C(N)} \quad ; \quad i \ge k \tag{12}$$

The average queue length at i in the system is given by:

$$E[L_i(N)] = \sum_{i=1}^{N} \rho_i \frac{C(N-1)}{C(N)} \tag{13}$$

The expected total number of jobs (threads) in the system:

$$L = \sum_{i=1}^{m} E[L_i(N)] + \sum_{j=1}^{n} E[L'_j(N)]$$

$$L = \frac{1}{C(N)} \left\{ \sum_{i=u}^{m} \sum_{u=1}^{N} \left(\frac{P_i}{P_l \mu_i}\right)^u C(N-1) + . \right.$$
$$\left. \sum_{j=l}^{n} \sum_{u=1}^{N} \left(\frac{P'_j}{P_l \mu'_j}\right)^u C(N-1) \right\} \tag{14}$$

- **Response time**

The average response time of task in node j is given by

$$R_{sys} = \frac{E[L_i(N)]}{\lambda_i} =$$
$$\frac{1}{\lambda_0 P_i} \sum_{l=1}^{N} (\rho_j) \frac{C(N-1)}{C(N)}$$
$$R_{sys} = \frac{E[L_i(N)]}{\lambda_i} =$$
$$\frac{1}{\lambda_0 P_i} \sum_{l=1}^{N} \left(\frac{P'_j}{P_1 \mu'_j}\right) \frac{C(N-1)}{C(N)} \tag{15}$$

$$\forall i = 1,2,...,m, \qquad \rho_j = \frac{P_0}{\lambda P_j'} .$$

The total response time of the system is given by

$$R_S = \sum_{i=1}^{m} R_{sys_i} + \sum_{j=1}^{n} R_{sys_j}$$

$$R_S = \frac{P_0}{\lambda}\left[\sum_{i=1}^{m}\sum_{u=1}^{N}\left(\frac{\rho_i^u}{P_i}\right)\frac{C(N-1)}{C(N)} + \sum_{j=1}^{n}\sum_{u=1}^{N}\frac{(\rho_j')^u}{P_j'}\frac{C(N-1)}{C(N)}\right] \quad (16)$$

- **Waiting times**

The total waiting time in the system is given by

$$E[W_S] = \sum_{i=1}^{m}\left(\rho_i^u \frac{C(N-1)}{C(N)} - \frac{1}{\mu_j}\right) + \sum_{j=1}^{n}\left(\frac{P_o}{\lambda P_j'}\sum_{i}^{N}(\rho_j')\frac{C(N-1)}{C(N)} - \frac{1}{\mu_j}\right) \quad (17)$$

4. NUMERICAL RESULTS

We used double 32 bits,

This is the specifications of my personal computer ***(PC***) are computing a matrix operations of multithreading for queueing parallel. We studied the operations of the 4×4 dimensional transpose and inverse matrix mathematics and four threads. We are important to studies some operations matices for CPU (*Processor = Intel (R) Core(TM) i7-2600 3.4 GHz and Ram= 8 GB DDR3*) of computer system. Show the analytical results MTA both sequential computing (simulation technique) and parallel computing for multithreading in computer system by using matlab programming.

First, we discuss to compute the operation of the 4×4 dimensional transpose matrix mathematics under 4 threads in by using **Gauss Jordan** method in the following this example:

Assuming the matrix is given by

$$A = \begin{bmatrix} 1 & 0 & 5 & 6 \\ 2 & 3 & 0 & 4 \\ 1 & 4 & 2 & 2 \\ 1 & 1 & 2 & 1 \end{bmatrix}$$

Applied MTA program, we note that vector matrix and matrix transpose stored in the thread of one row to equal =

[1 2 1 1 ; 0 3 4 1 ; 5 0 2 2 ; 6 4 2 1].

And the transpose matrix is given by

$$A_{transpose} = \begin{bmatrix} 1 & 2 & 1 & 1 \\ 0 & 3 & 4 & 1 \\ 5 & 0 & 2 & 2 \\ 6 & 4 & 2 & 1 \end{bmatrix}.$$

Table 1: Compare between the computing sequential and computing parallel of response time in queueing network for multithreading.

Response Time (second)	Sequential	Parallel
	1.03245	0.75412

Figure1: The response time by second

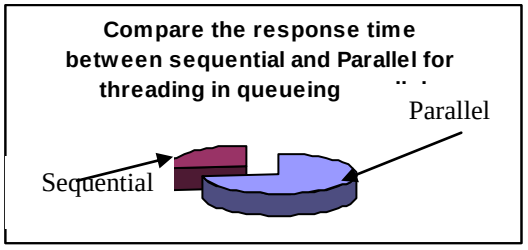
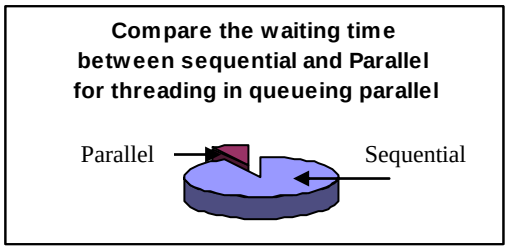


Table 2: Compare between the computing sequential and computing parallel of waiting time in queueing network for multithreading.

Waiting Time (Microsecond)	Sequential	Parallel
	92832	4539

Figure 2: The waiting time by micro second.



By Table1 and 2 and also Figure1 and2 of the 4×4 dimensional matrix mathematics under four threads, the parallel queueing model is the fastest running time and to reduce waiting time or elapsed time for multithreading of queueing theory in computer system (CPU).

5. CONCLUSIONS

In this chapter, the model is a closed queuing network of MTA. The instruction streams are executed simultaneously (multithreading) to minimize the loss of CPU cycles. An algorithm is used to compute the normalization constant as a function of the degree of multiprogramming (number of active jobs) in the queueing model. The system performance measures are derived knowing the normalization constant. The response time is found to increase with the total number of processors, and the response time is found to increase with the degree of multiprogramming. In the closed queueing network with circumstances, an increase in the service rates of the computer system (CPU). Finally, the optimum number of multithreading model achieves to minimize waiting (elapsed) time per microseconds under

increasing the total number of processors of computer system (CPU).

6. REFERENCES

[1] Anoop Gupta , John Hennessy , Kourosh Gharachorloo , Todd Mowry , Wolf-dietrich Weber, Comparative Evaluation of Latency Reducing and Tolerating Techniques, CiteSeex ,1991.

[2] Bhasker.V., "A closed queueing network model with single servers for multithread architecture ".plied Mathematical Modelling Vol.33, PP.3599-3616,2009.

[3] Greenberg. A. G., Lubachevsky B. D., and Mitrani.I., "Algorithms for unbo- undedly parallel simulation s".ACM transactions on Computer Systems Vol.9 , No.3 , PP. 201–221. 1991.

[4] Ismail.A.I. and Elbehady .E.E.," Finding computable Green's function for parallel planes and an open rectangular channel flow". Egyptian computer science journal, vol.6, No.1, PP. 36-44, 2000.

[5] Ricardo Bianchini , Beng-hong Lim, "Evaluating the Performance of Multithreading and Prefetching in Multiprocessors", IBM,1996.

[6] .ScottGraham.G.andKenneth.C.Sevcik.,Quantiative System performance computer analysis using queueing network models, prentice-hall inc., upper sadd river, NJ,USA. 1992.

[7] Zhao.Y. and SimchiLevi.D, "Performance analysis and evaluation of assemble-to-order systems with stochastic sequential lead times ". Operations Research Vol. 54, No. 4, PP.706–724,2006.

[8] V. Vlassov , L-E Thorelli , A. Kraynikov, A Queuing Model of Multithreading: A Case Study, CiteSeex,1996.