CSCanada
Studies in Mathematical Sciences
Vol. 5, No. 1, 2012, pp. [41–48]
DOI: 10.3968/j.sms.1923845220120501.1366

ISSN 1923-8444 [Print]
ISSN 1923-8452 [Online]
www.cscanada.net
www.cscanada.org

Compare Inverse Matrix Between Sequential and Parallel for Multithreading with Queueing Network

I.A. Ismail[a], G.S. Mokaddis[b], S.A. Metwally[b] and Mariam K. Metry[c],*

[a] Department of Computer Science, Faculty of Computer Science and Information System, 6 October University, Cairo, Egypt.
[b] Department of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt.
[c] Researcher of Queueing Systems & Parallel programming and Engineer software of AOI Cairo, Egypt.

* Corresponding author.
Address: Researcher of Queueing Systems & Parallel programming and Engineer software of AOI Cairo, Egypt; E-Mail: mari25eg@yahoo.com

Received: June 16, 2012/ Accept: August 15, 2012/ Published: August 31, 2012

Abstract: This work deals with minimizing the computing time for matrix inversions used in the queueing system models or otherwise. The time is reduced considerably and is proportional to the number of the used threads in parallel.

Key words: Closed queueing network; Inverse matrix; Parallel programming and computing

Ismail, I.A., Mokaddis, G.S., Metwally, S.A., & Metry, M.K. (2012). Compare Inverse Matrix Between Sequential and Parallel for Multithreading with Queueing Network. *Studies in Mathematical Sciences, 5*(1), 41–48. Available from http://www.cscanada.net/index.php/sms/article/view/j.sms.1923845220120501.1366 DOI: 10.3968/j.sms.1923845220120501.1366

1. INTRODUCTION

The inverse matrix frequently used in queueing parallel model by using multithreading software. Queueing for 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 [5] and [2]. Performance measures such as average response times and average system throughput are derived to against the total number of processors in the closed queueing network model. The model is validated by comparison of analytical parallel and sequential result.

2. DESCRIPTION INVERSE MATRIX

2.1. By Partitioning for Inverse Matrix

The matrix inversion is given for nonsingular numerical matrix A, we can partition A into four sub matrices: Where $r = n/2$

$$A = \begin{bmatrix} \alpha_{11}(r,r) & \alpha_{12}(r,r) \\ \alpha_{21}(r,r) & \alpha_{22}(r,r) \end{bmatrix}$$

The orders of the sub matrices are indicated in parentheses; and, n is the order of square matrix A. We seek the inverse in the form of a four block matrices A^{-1} also

$$A^{-1} = \begin{bmatrix} \beta_{11}(r,r) & \beta_{12}(r,r) \\ \beta_{21}(r,r) & \beta_{22}(r,r) \end{bmatrix}$$

Then, since $AA^{-1} = E$, by multi-plying the matrices we get four matrix equations

$$\begin{aligned} \beta_{11}\alpha_{11} + \beta_{12}\alpha_{21} &= E_r, \\ \beta_{11}\alpha_{12} + \beta_{12}\alpha_{22} &= 0, \\ \beta_{21}\alpha_{11} + \beta_{22}\alpha_{21} &= 0, \\ \beta_{21}\alpha_{12} + \beta_{22}\alpha_{22} &= E_s, \end{aligned} \tag{1}$$

where E_r and E_s are the unit matrices of appropriate orders. Solving this system, we determine the blocks of matrix A^{-1}. In solving in Equation (1).

$$\begin{aligned} \beta_{11} &= {\alpha_{11}}^{-1} + X\theta^{-1}Y, \\ \beta_{12} &= X\theta^{-1}, \\ \beta_{21} &= Y\theta^{-1}, \\ \beta_{11} &= \theta^{-1}. \end{aligned}$$

From Equation (1) demine the blocks of matrix A^{-1} in the following scheme:

	α_{21}	α_{22}
$X = \alpha_{21}^{-1}\alpha_{22}$	α_{11}^{-1}	α_{12}
θ^{-1}	$Y = \alpha_{21}\alpha_{11}^{-1}$	$\theta = \alpha_{22}^{-Y}\alpha_{12}$

and the finial inverse matrix A^{-1} is given as follows

$$A^{-1} = \begin{bmatrix} \alpha_{11} - 1 + X\theta^{-1}Y & -\mathrm{X}\theta^{-1} \\ -\theta^{-1}Y & \theta^{-1} \end{bmatrix} \tag{2}$$

2.2. Solving Systems for Inverse Matrix

Let the matrix is solving linear systems for determining non zero and numerical be non singular. A frequently quoted test for invertability of matrix is based on the concept of the **determinant**. $[A]$ is invertible if and only if $[A] \neq 0$. Numerical method solving linear systems may be divided into two types **direct** and textbfiterative [8]. Computing the inverse matrix in queueing network model is obtained by solving linear equations with constant vector under where $[A][X] = I_i$ instantaneously where I_i is the unit vector with i^{th} row= 1,

$$I_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}; \; I_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}; \; I_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}; \; I_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

3. PARALLEL PROGRAMMING MODEL

Mathematical methods and simulation are used to analyze various architectural solutions in MTA (Mean Thread Architectural) design. A few attempts of MTA mathematical evaluation of block multithreaded have resulted in deterministic analytical models and queuing models [7,9]. A technique of analytical modeling of such is mainly based on the MTA consideration of a set of thread states and state transitions.

3.1. Data Dependency Analysis

Dependency among program segments arises from primarily three sources:

(a) Control,

(b) Data and

(c) Resources.

3.2. Threads Model

Multithreading is similar to multiprocessing programming [1,6]. It is to be born in mind that the number of threads was taken into consideration equals the number of servers or queues considered. The difference is that a multithreaded program has a single processor which manages multiple threads of control executing asynchronously. 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.

3.3. 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.

4. THE PERFORMANCE MEASURES OF QUEUEING PARALLEL COMPUTING

We can be described the performance measures of queueing parallel computing under MTA [3,4,7]. We note that the number of threads = the number of servers $= N$.

4.1. Queue Lengths

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

$$P(N_i) = \rho_i^k \frac{C(N-1)}{C(N)}; \quad i \geq k \tag{3}$$

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

$$E[L_i(N)] = \sum_{i=1}^{N} \rho_i \frac{C(N-1)}{C(N)} \tag{4}$$

The expected total number of servers (threads) in the system: $L = \sum_{i=1}^{m} E[L_i(N)] + \sum_{j=1}^{n} E[L'_j(N)]$

$$L = \frac{1}{C(N)} \left[\sum_{i=u}^{m} \sum_{u=1}^{N} \left(\frac{P_i}{P_l \mu_i}\right)^u C(N-1) + \sum_{j=l}^{n} \sum_{u=1}^{N} \left(\frac{P'_j}{P_l \mu'_j}\right)^u C(N-1) \right] \tag{5}$$

4.2. Response Time

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

$$R_{sys} = \frac{E[L_i(N)]}{\lambda_i} = \frac{1}{\lambda_0 P_i} \sum_{l=1}^{N} (\rho_j) \frac{C(N-1)}{C(N)}$$

$$R_{sys} = \frac{E[L_i(N)]}{\lambda_i} = \frac{1}{\lambda_0 P_i} \sum_{l=1}^{N} \left(\frac{P'_j}{P_1 \mu'_j}\right) \frac{C(N-1)}{C(N)} \tag{6}$$

where $\forall i = 1, 2, ..., m, \quad \rho_j = \dfrac{P_0}{\lambda P'_j}$. The total response time of the system is given by

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

$$R_s = \frac{P_0}{\lambda} \left[\sum_{i=1}^{m} \sum_{u=1}^{N} \left(\frac{\rho_i^u}{P_i}\right) \frac{C(N-1)}{C(N)} + \sum_{j=1}^{n} \sum_{u=1}^{N} \frac{(\rho'_j)^u}{P'_j} \frac{C(N-1)}{C(N)} \right] \tag{7}$$

4.3. Waiting Times

The total waiting time in the system is given by

$$E[W_s] = \sum_{i=1}^{m} (\rho_i^u \frac{C(N-1)}{C(N)} - \frac{1}{\mu_j}) + \sum_{j=1}^{n} (\frac{P_o}{\lambda P'_j} \sum_{i}^{N} (\rho'_j) \frac{C(N-1)}{C(N)} - \frac{1}{\mu_j}) \qquad (8)$$

5. NUMERICAL RESULTS

Using double precision 32 Bits in the processor in matlab programming.

	Specifications
Processor	Min. Intel (R) Core(TM) i7-2600 3.4 GHz Processor
Memory(Ram)	8 GB DDR3
Storage	Min. 2 TB SATA III/7200
Operating System	Windows Starter Edition License (Media is not required)

This is the specifications of my personal computer (PC). We studied theoperations of the 4×4 dimensional inverse matrix mathematics and four threads. Show the analytical results MAT both sequential computing (simulation technique) and parallel computing for multithreading in computer system by using matlab programming. We start determining the inverse of our 4×4 matrix by solving $Ax = b_i$, where b_i is the column matrix $= \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

The number 1 lies in the ith row. Thus, we have to solve n linear systems from $i = 1$ to n. Each solving represent of column of the required matrix A^{-1}.

Observing the above steps are independent we can execute them in queueing parallel. Each step includes the computation of one column of the inverse matrix, to finally get the columns of the inverse matrix simultaneously.

The following example on finding the inverse of 4×4 matrixes is:

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

The inverse matrix is above method to this matrix A^{-1}, we get

$$A^{-1} = \begin{bmatrix} -0.2278 & -0.0506 & 0.0633 & 0.1519 \\ 0.3291 & -0.0380 & -0.2025 & 0.1139 \\ -0.5190 & 0.3291 & 0.0881 & 0.0127 \\ 1.0886 & -0.2025 & 0.2532 & -0.3924 \end{bmatrix}$$

We compute the response time of the inverse matrix in queueing network model by three methods in Table 1 and Figure 1.

Table 1
Compute the Response Time per Second R_S

	Sequential	Parallel
Partitioning inverse	1.789	0.9482
Linear solving systems	1.3432	0.7531

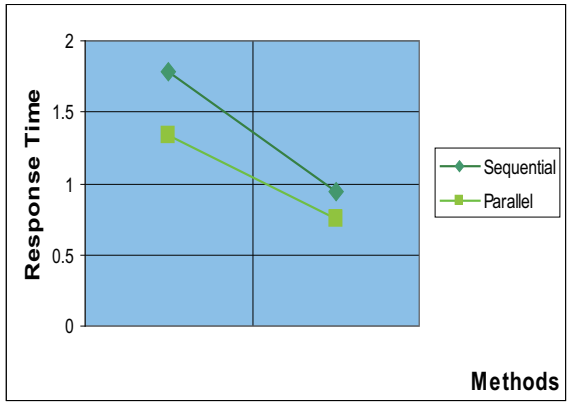

Figure 1
Compute the Response Time per Second

We compute the waiting of the inverse matrix in queueing network model by three methods in Table 2 and Figure 2.

Form the above results for the case of 4×4 dimensional inverse matrix mathematics with constant vector matrix. It turns out that using fourth thread. Finding the linear solving systems is the best partitioning inverse systems.

In general the 4×4 dimensional matrix mathematics under four threads, we proved the parallel computing is the better than the sequential (simulation) technique of using inverse matrix for multithreading in queueing theory. 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).

6. CONCLUSIONS

By adopting the parallel computing for inverting $n \times n$ matrices, we could see that the time needed to compute the inverse is much less than the standard used methods. Therefore, using thistechnique in multiserver queueing system minimizes the computing time considerably. Finally, the optimum number of multithreading model for inverting $n \times n$ matrices achieves to minimize waiting (elapsed) time per microseconds under increasing the total number of processors of computer system (CPU).

Table 2
Compute the Waiting Time per Microsecond E[W_S]

	Sequential	Parallel
Partitioning inverse	5840	1865
Linear solving systems	2000	1300

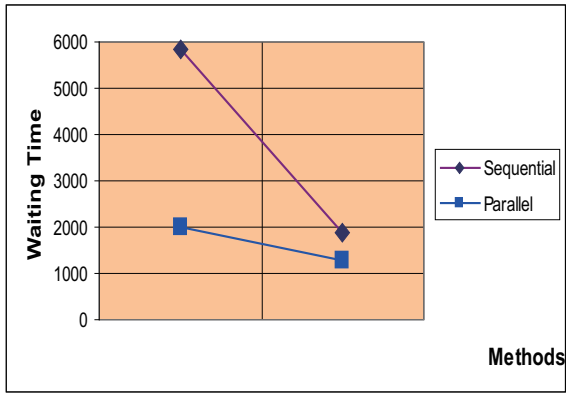

Figure 2
Compute the Waiting Time per Microsecond

REFERENCES

[1] Bhasker, V. (2009). A closed queueing network model with single servers for mulltithread architect-ture. *Applied Mathematical Modelling, 33*, 3599-3616.

[2] Greenberg, A.G., Lubachevsky, B.D., & Mitrani, I. (1991). Algorithms for unboundedly parallel simulations. *ACM Transactions on Computer Systems, 9*(3), 201-221.

[3] Hassan, A.N., & Metry, M.K. (2009). Performance analysis of multiclass queueing network model with positive Poisson batch sizes following discrete review policy. *Model Assisted Statistics and Applications, 4*(1), 5-13.

[4] Ismail, A.I., & Elbehady, E.E. (2000). Finding computable Green's function for parallel planes and an open rectangular channel flow. *Egyptian Computer Science Journal, 6*(1), 36-44.

[5] Ismail, I.A., Mokaddis, G.S., & Metry, M.K. (2012). Computing a matrix transpose of multithreading for queueing parallelin matlab programming. *International Journal of Computer Applications, 49*(5), 41-47.

[6] Sasikumar, M., & Raviprakash, P. (2003). *Introductions to parallel processing.* Prentice Hall of India.

[7] Scott, G.G., & Kenneth, C.S. (1992). *Quantitative system performance computer system analysis using quequeing network models.* NJ, USA: Prenticehall. Inc., Uppersaddleriver.

[8] Shan, K.S. (1973). *Computer applications of numerical methods.* Philippines.

[9] Zhao, Y., & SimchiLevi, D. (2006). Performance analysis and evaluation of assemble-to-order systems with stochastic sequential lead times. *Operations Research, 54*(4), 706-724.