An optimal parallel algorithm for finding roots of linear equations system

Mehrdad Hashemi [1], Shabnam Ahari Abeshahmad[2]

[1]*Baku State University, Baku, Azerbaijan*

[2]*Islamic Azad University, Ahar, Iran*

Mehrdad.hashemi@mail.ru, sh_ahari@yahoo.com.

Abstract. *In this paper we present an optimal parallel algorithm which finds roots of Linear Equations system. Suggested algorithm works based on Multi Processor architecture of parallel computers. Special method of parallel algorithm which named Pipe Lined, is preferred in our algorithm. Here we use Shared Memory programming method and give each equation to one process that can be executed by one physical processors. Memory contention and other inevitable problems by using of shared memory programming method have been considered in the algorithm.*

Keywords

System of linear equations, parallel algorithm, pipe line method, shared memory.

1 Introduction

Consideration to actuality and importance of the computational and engineering problems which contains the system of linear equations, from around the world, there are many serial and parallel solutions are suggested for solving them [1,2,3,4,5]. We can find the system of linear equations in many physical, industrial and mechanical problems too. In these problems if we use serial structural systems, also the super computers can't operate effectively at all moments.

As we know, by using n-processors we can increase of the processing power of computational systems in n-fold. This was the main idea in invention of parallel computing. On the other hand, by increasing number of equations, amount of computations and size of "n" will be increased and therefore it is better to use parallel algorithms for solving it [6]. According to importance of this problem all experts want to find an optimal solution method for solving linear equations system.

2 Theoretical aspect of the problem

One of techniques which used in parallel programming is named "Back Substitution". Every system of linear equation composed of the relationship between unknown amounts "$x_1,x_2,...,x_n$" . According to the mothod Gauss-Jordan, the unknown "x_i" can be founded just after knowing "$x_1,x_2,...,x_{i-1}$" easily if we convert the system into lower triangular.

Each parallel algorithm which suggested for solving of this problem, has own superiority among the others. These superiorities related on the architecture, topology and techniques which applied in algorithms. In suggested algorithm we use 2-dimensional array A for keeping amounts of "$a_{i,j}$" and 1-dimensional B and X arrays for keeping amounts of "b_i" and "x_i" in shared memory of system [7].

In our parallel algorithm, each equation will given to one process which executed by one physical processor of multiprocessor system. Since i-th equation containes "$x_1,x_2,...,x_i$" we can give this equation to i-th process. But it should be attended that each process for executing it's own operations should wait finishing operations of all previous processes. General furmula for finding each "x_i" is: $Xi = (b_i - a_{i1}x_1 - a_{i2}x_2 - a_{i\,i-1}x_{i-1}) a_{ij}$

But we can't convert all instructions of non parallel program to parallel easily. We use Multi pascal language created by B. P. Lester for simulation and evaluation of suggested algorithm and therefore we use it's instructions for parallelization of serial program. In this situation it may seems that the algorithm will executed in non parallel structure. But still there is a chance for parallelism.

Providing the interprocesses relation network and also keeping synchronazation between them is produced by the capability of multi pascal language which named "channel variable". Channels create "pipe lined" category of parallel algorithms. This method is the special form of "producer-consumer" topic which discussed in operating systems[8]. According to this structure, the data is flowing through the channel. After executing of determined operations on the data,

they will arrive to end of channel and will give us the final result contains amounts if all "x_i"s. Processes are collected and compose one channel. Each process in the mentioned channel, execute it's own operations with others at the same time. But this operations executed on different data and this is the parallelism in our computational problem.

Since this will be occured on the different data of arrays, they have not to wait for each other in all moments. In the channel each process get the result of process on the left hand and execute determined operations on them and give own result to process on the right hand.

3 Implementation

The other capability of multi pascal language is that this simulation language can collect channels and compose a array of them. In this way, we can apply some operations such write, read and numbering on them.

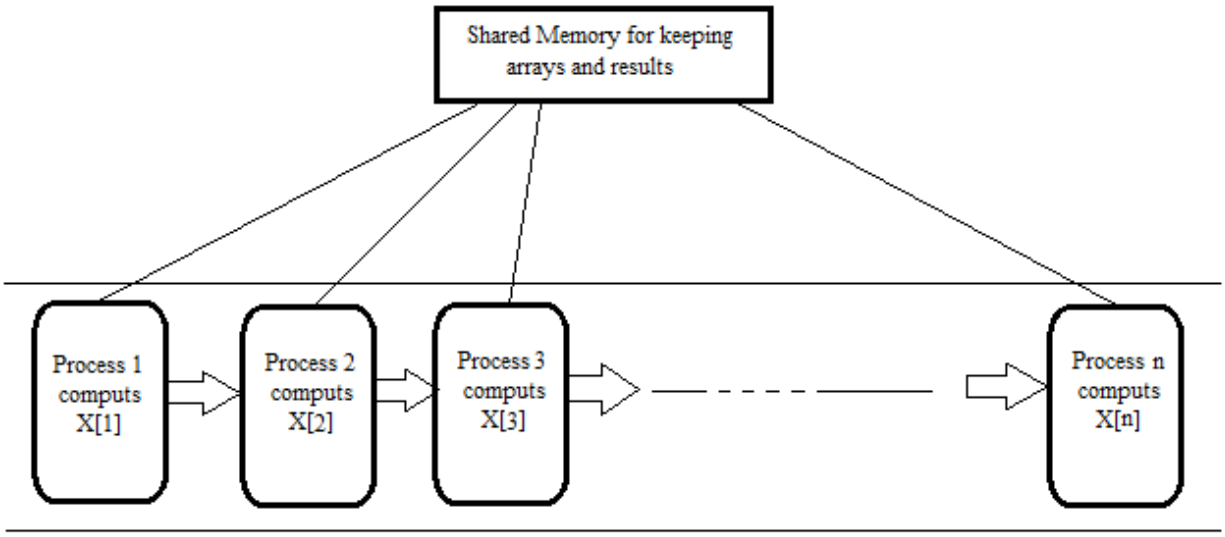

fig. 1. The structure of Pipe Lined method in parallel solving of linear equations system

In the figure 1 you can see the structure of pipe lined method for solving disccused problem. Because of using shared memory programming technique in our algorithm, all of processes at the same time write their own result in the shared memory of system by sending the result to process on the right. We should attend to memory contention and other problems which we may confront by using shared memory.

For solving related problem we can devide the memory into modules for avoiding memory contention and memory hot spots in our computations. Also we can use local cache memory for each physical processor to keep the latest copy of informations in the pipe. Using any of these methods for achiving the best memory status is depended on the computational problem and architecture which we applied in desiging of parallel algorithm[9][10].

4 Conclution

For creation of pipe line, we used an array composed of channels in our program. Because of unical structure of pipe we assumed each equation to one physical processor for finding unknowns. Since all processes has a communication link with shared memory of system, all of them can get needed known data from memory and also put own results on it for using other processes. The data flows from left to right through the channel. This strategy will be continued when the data arrive to the end of channel. At the end of channel x[n] will be computed and the problem will be solved.

As it seems, there is both sequential and parallel structure in the algorithm. This factor icrease the general effectivity of the algorithm. By increasing of the number of process in the algorithm, the process with largest number among all, will execute more operation than others. Knowing that our system is lower triangular, we can say that most number of computations are executed by final process and the most waiting time is occured on this process which execute "n" adding and multiplicating operations.

In other word, the needed executive time of suggested algorithm is equal to the time for tramsmition of x[1] to final proscess (or processor). Thus according to rules for computation of complexity, we can say that the complexity of given algorithm by us is equal to O(n) [11].

References

[1] M.Otelbaev, D.Zhusupova, B.Tuleuov: On a method of Parallel Computation for solving linear algebraic system with ill-conditioned matrix. *L. N. Gumilev Eurasian national University, Astana, Kazakhstan,* 10 p.

[2] R. Mehmood, J. Crowcroft: Parallel iterativ solution method for large sparse linear equation systems, *University of Cambridge. Computer Laboratory, No 650, Cambridge, UK,* 20 p , 2005.

[3] S. H. Abbas: Parallel Solution of Linear System of Equations on Transputer Array. *Applied Mathematics & Information Sciences, No 2, University of Bahrain,* 10 p , 2008.

[4] J. Michael. Quinn: Parallel Programming in C with Mpi and Openmp. *Oregon State University , McGraw Hill , USA,* 2004 , 529 p.

[5] R. Smith. Justin: The Design and Analysis of Parallel Algorithms, *Drexel University, Philadelphia, USA* , 1995 , 462 p.

[6] M. Tanavosh: Parallel Programming. *Naghoos press, Tehran, Iran,* 2009, 549 p.

[7] G. AKL. Selim: The Design and Analysis of Parallel Algorithms. *Queen's University , Kingstone , Canada* , 1989 , 401 p.

[8] W. Stallings: Operating Systems. *Pearson Education, USA,* 2006, 818 p.

[9] M. Tanavosh: Parallel Programming. *Naghoos press, Tehran, Iran,* 2009, 549 p.

[10] M. Hashemi: The problem of memory contention in multiprocessor systems. *Scientific conference of applied mathematics and cybernetics faculty, Baku State University, Azerbaijan*, 2011, 3 p.

[11] R. E. Neapolitan, Naimipour K: Foundation of Algorithms (using C++ pseudocodes). *translated by E. Jafarnejad, Mashhad, Iran,* 2002, 413 p.