ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 8, August 2016

An Analytical Study on effect of Parallelism on Multicore

Varsha Thakur[1], Sanjay Kumar [2]

[1] Research Scholar, Pt.Ravishankar shukla University ,Raipur Chhatisgarh ,India.

[2] Associate Professor, Pt.Ravishankar shukla University ,Raipur Chhatisgarh ,India

***Abstract*—** Parallel computing has been fast evolving research areain the last few decades. Parallel computing means simultaneous execution of the same task by dividing the task into subtasks, on multiple processors. This paper provides an overview of multicore architecture and shows the effect of parallelism in multicore. Performances was evaluated in a multicore on the basis of the run time of serial and parallel algorithms. To implement matrix multiplication algorithms C programming language with OpenMp Libraries was used under Linux and Windows environment..

***Index Terms*— Matrix Multiplication, OpenMp. Parallel Algorithm.**

I. INTRODUCTION

A Multicore processor is an Integrated Circuit in which more than one processor or core are included for performance improvement and Simultaneous processing of parallel jobs. Matrix multiplication is a well known mathematical term in a linear algebra which is used for solving large computation problem. We are living in the era of parallel computing where performance and efficiency are of fundamental importance [1]. OpenMp is used for parallelizing the sequential matrix multiplication. In rest of paper we have define some basic concept of OpenMP, Multicore Architecture and Matrix Multiplication Algorithms.

A. OPENMP

OpenMp is an API (Application Program Interface) that use multithreaded and shared memory parallelism. Openmp is basically divided into three parts first one is compiler directives, second is runtime library routines and third as environment variable. This API is an open specification for multiprocessing. OpenMp worked as a fork-join model where fork is master thread that use to create a team of parallel thread and join is used when the team of parallel threads complete their task they synchronize and terminate and left the master thread to execute sequential program. OpenMp visualize as parallel programming model on multicore architecture [3].

B. Muticore Architecture

A multicore places multiple processors on a single chip and each processor is called a core [2]. As we increase the capacity of chip placing m.ultiple processors on a single chip became practical. These architectural designs are known as Chip Multiprocessors (CMPs), chip Multiprocessors are known as Multicore. A multi-core processor is a logic circuit in which more than one processor is placed. In multicore multiple processors can execute parallel and increases performance. Multicore span threads which divide the tasks among cores. It can execute multiple tasks at single time. Multicore is shared memory processors, all processors shares the same memory. Multicores are becoming popular for both server and desktop processors. By the next decade, it is expected to have processors with hundreds of cores on a chip.

II. MATRIX MULTIPLICATION

Matrix multiplication is a mathematical binary operation that takes input as pair of matrices, and gives output of another matrix.

A. Sequential Matrix Multiplication

The sequential matrix multiplication is the fundamental basis for other matrix multiplication. Matrix multiplication between two matrices is possible when row sizeof first matrix match with column size of second matrix. The product of l X m matrix A with m X n matrix B is an l X n matrix C where element is defined as

$$C_{ij} = \sum_{m-1} a_{ik} b_{kj} \text{ where } 0 \le i < a, 0 \le j < c$$

Sequential matrix multiplication requires l * m*n addition and same number of multiplication so, time complexity of multiplication of matrix using sequential algorithm is O (N^3).

B. Parallel Matrix Multiplication

In last few decades various approaches has been proposed for implementation of matrix multiplication on shared memory architecture. All parallel algorithms are based on conventional sequential matrix multiplication. For parallel

matrix multiplication consider two nxn matrix A and matrix B. Partition the matrix in L blocks where (0<=i, j<root l) of size (n/root l) x (n/root l) each small matrix the n this small matrix mapped into root l X root l mesh of processors. The process initially stores Aij and Bij and compute Cij of result matrix. After computing the entire sub matrix, matrix A's block performed in each row and matrix B's performed in each coloum. Finally sub matrix multiplication and addition is performed. In parallel algorithm each element of matrix C is computed simultaneously. Time complexity of multiplication of nxn matrix using parallel algorithm is O (N^2).

C. Strassens Algorithms

Strassens algorithm is an algorithm used for matrix multiplication which is usually faster than other simple matrix multiplication [6]. Strassens algorithm partitions the data to reduce the number of multiplications .It uses divide and conquer approach for multiplication . In case of two by two matrix multiplication strassans algorithm requires 7 multiplication and 18 ad or subtract operation. , instead of the classical algorithm that does 8 multiplications and 4 additions. This multiplication requires matrix dimension and processors to be power of 2.

In this algorithm matrices are partitioned into equal sized block matrices such as.

In this algorithm matrices are partitioned into equal sized block matrices such as

$$A = \begin{vmatrix} a11 & a12 \\ a21 & a22 \end{vmatrix} \quad B = \begin{vmatrix} b11 & b12 \\ b21 & b22 \end{vmatrix} \quad C = \begin{vmatrix} c11 & c12 \\ c21 & c22 \end{vmatrix}$$

Where C=AB, Strassens define new intermediate matrices as follows

D1 :=(a11+a22) (b11+b2 2), D2 :=(a21+a2 2) b1 1, D3:=a1,1(b1 2-b2 2) D4:=a2 2(b21-b11), D5 :=(a11+a12) b22, D6 :=(a21-a11) (b11+b12), D7 :=(a12-a22) (b21+b22).

$C_{i,j}$ in terms of Dk is like this:

C11=D1+D4-D5+D7
C12=D3+D5
C21=D2+D4
C22=D1-D2+D3+D6

Strassens parallel matrix multiplication is possible since intermediate matrix are independent can be run simultaneously on machine with different processors. Parallel algorithm use task pool model to compute intermediate matrix the following diagrams illustrate the recursion and parallelism in Strassens algorithm [7].

Strassen algorithms complexity depending on execution time is O ($N^{2.8}$) [7].

III PERFORMANCE MEASUREMENTS

Performance of a parallel algorithm is measured using two factors speed-up and efficiency.

A. Speedup

In parallel computing, speedup refers to how much faster a parallel algorithm is run in parallel.

$$\text{Speed up} = \frac{\text{Sequential execution time}}{\text{Parallel execution time}}$$

Speed up means the ratio of run time before improvement (in sequential) to the run time after improvement(in parallel).

B. Efficiency

In parallel computing, efficiency refers to speed up divided by number of processors. Efficiency calculates among n available processors how much processors are used during execution.

$$\text{Efficiency} = \frac{\text{Sequential execution time}}{\text{Parallel execution time X processor used}}$$

IV. EXPERIMENTAL SETUP

For Experiment two computer systems are taken. First was with dual core with 1.83 GHz speed second was of Quad core with 2.13 GHZ speed alongwith two different operating system Linux and Windows 7. We have run the sequential matrix multiplication, parallel matrix multiplication and Strassens(Sequential and parallel) matrix multiplication on with two different computers one is dual core and other is quad core. Execution time was recorded as shown in Table I for dual core and Table II for Quad core and is analyzed graphically.

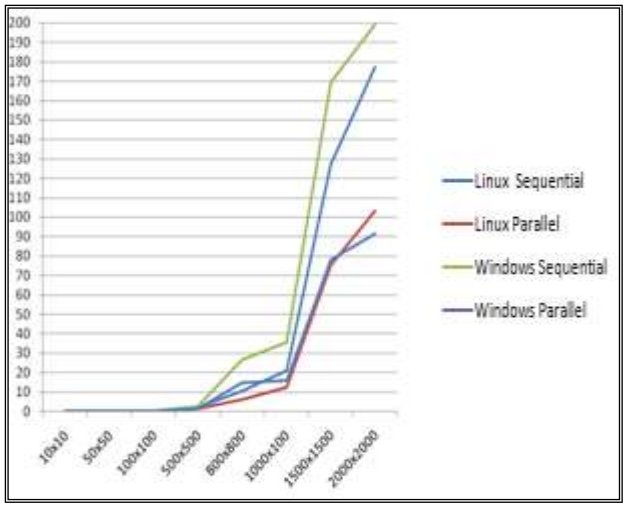
Table I: - Speed up of algorithms for dual core processors.

Matrix Size	Sequential			Parallel			Speed up		
	Linux	Windows	Linux Strassens	Linux	Windows	Linux Strassens	Linux	Windows	Linux strassens
10x10	.0012	.0031	.0021	.0023	.0045	.0035	0.521	0.68	.60
50x50	0.06	.019	.075	0.071	0.075	.071	.84	.25	1.0
100x100	0.01	0.011	0.01	.01	.01	.01	1.0	1.1	1.0
500x500	2.23	2.38	2.15	1.38	1.41	1.53	1.61	1.71	1.40
800x800	10.34	26.56	9.58	6.24	14.69	6.69	1.65	1.808	1.43
1000x1000	20.85	35.67	18.83	12.41	16.01	10.8	1.68	2.18	1.74
1500x1500	127.3	169.27	66.06	75.14	78.27	34.23	1.70	2.21	1.92
2000x2000	177.1	199.12	113.76	103.26	91.45	58.64	1.72	2.2	1.94

Table II: - Speed up of algorithms for Quad core processors

Matrix size	Sequential			Parallel			Speed up		
	Linux	windows	LinuxStrassens	Linux	windows	Linux Strassens	Linux	Windows	Linux Strassens
10x10	0.05	0.019	0.014	0.061	0.026	0.031	.81	.73	.45
50x50	0.23	0.08	0.08	0.03	0.010	0.12	.80	.80	.66
100x100	0.6	0.018	.5	.56	.02	.5	1.07	.91	1.0
500x500	1.13	2.1	1.19	.705	1.23	1.02	1.61	1.707	1.16
800x800	7	8.36	6.59	4.31	4.81	3.51	1.62	1.73	1.88
1000x1000	11	15.34	9.87	6.689	7.94	5.23	1.66	1.92	1.89
1500x1500	36.91	51.32	30.46	22.09	27.11	16.23	1.71	1.91	1.87
2000x2000	89.25	122.85	75.81	51.07	68.52	39.81	1.75	1.95	1.93

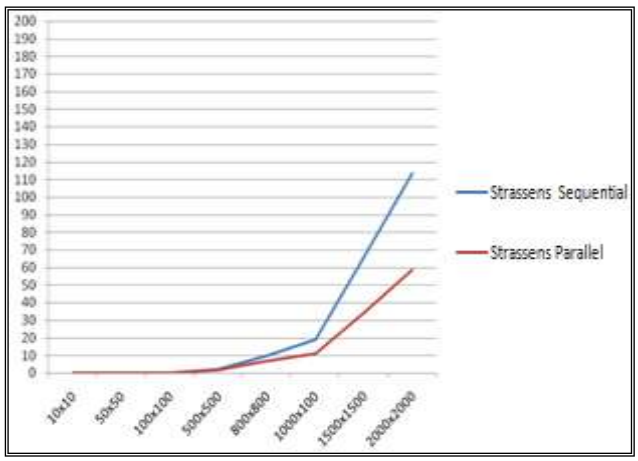
Graph 1 was plotted for time taken during execution of sequential and parallel algorithm in linux and windows platform in dual core.



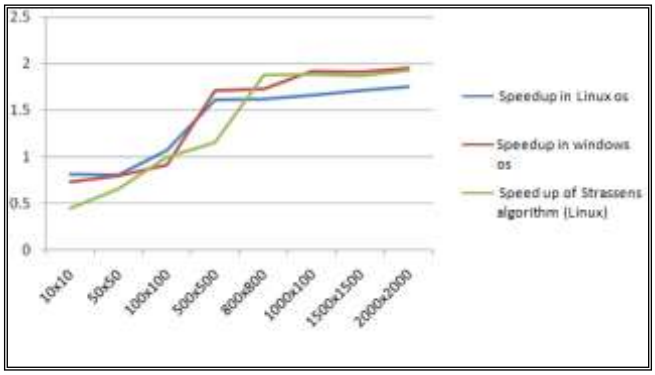
The horizontal axis represents matrix size and vertical axis represents running time in milliseconds.

Graph 2 was plotted for time taken during execution of sequential and parallel strassens algorithm in linux and windows platform in dual core.

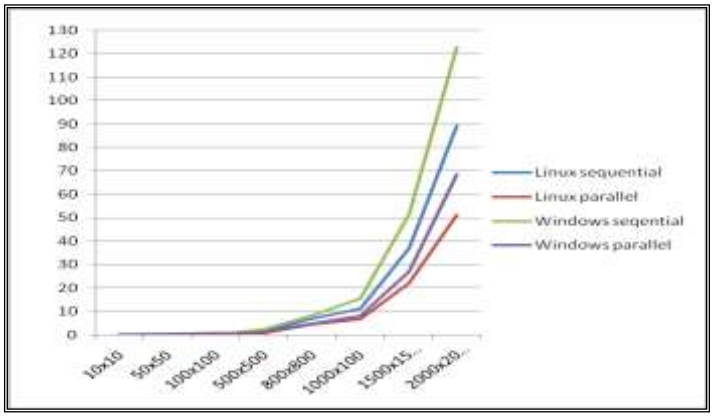
The horizontal axis represents matrix size and vertical axis represents running time in milliseconds. Graph 3 was plotted for speed up in linux and windows platform in dual core



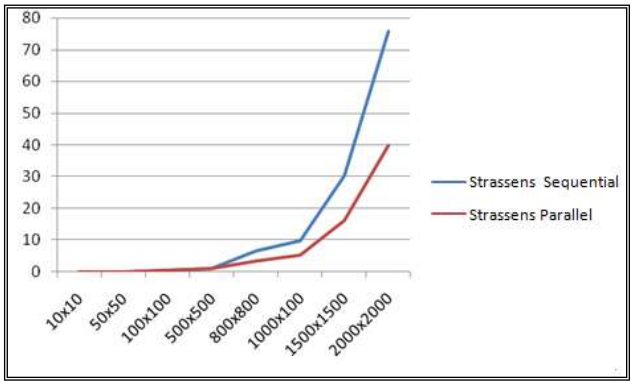
.



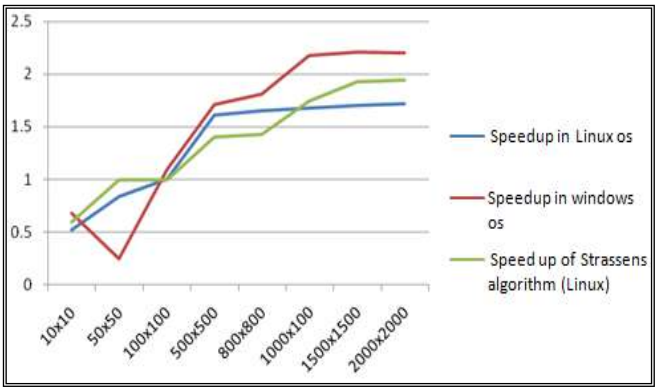
The horizontal axis represents matrix size and vertical axis represents running time in milliseconds.Graph 4 was plotted for time taken during execution of sequential and parallel algorithm in linux and windows platform in quad core.



The horizontal axis represents matrix size and vertical axis represents running time in milliseconds.Graph 5 was plotted for time taken during execution of sequential and parallel strassens algorithm in linux and windows platform in quad core.



The horizontal axis represents matrix size and vertical axis represents running time in milliseconds. Graph 6 was plotted for speed up in linux and windows platform in quad core



.

III. CONCLUSION

In this experiment execution time and speedup for sequential, parallel and strassans matrix multiplication was calculated. It is clear from the graph as OpenMp parallaize sequential program performance get increases for higher order matrix while for lower order matrix execution time with parallalization is more than sequential multiplication.Strassans algorithm is faster than conventional algorithms. The operation in strassens multiplication is decreases by which it reduced time but increases the space complexity. It is also observed that matrix multiplication in linux environment is faster than windows environment and quadcore take less computational time than dual core. Beyond a certain optimum problem size only parallelization is effective below that point because of communication overhead sequential algorithm on sequential machines will give better results.

.

REFERENCES

[1] . Keqin L., Yi P., Si Qing Z.,"Fast and Processor Efficient Parallel Matrix Multiplication Algorithms on a Linear Array With a Reconfigurable Pipelined Bus System", IEEE Transaction on parallel and distributed, VOL. 9, AUG 1998, pp 705-720.
[2] Cameron, H., Tracy, H., Professional Multicore programming, Wiley publication,2008.
[3] Venkatesan P., Harish B., S. Sarholz, Proceedings of the 3rd international workshop on OpenMP "A Practical Programming Model for the Multi-core Era",2008.
[4] Rose M. P., "A Parallel Approach for MatrixMultiplication on the TMS320C4x DSP", Digital Signal Processing Semiconductor Group,Texas Instruments**,** Feb 1994.
[5] . http://en.wikipedia.org/wiki/Matrix_multiplication
[6] http://en.wikipedia.org/wiki/Strassen_algorithm
[7] Juby M., R. Vijaya K., "Comparative Study of Strassen's Matrix Multiplication lgorithm", International Journal of Computer Science And Technology IJCST Vol. 3, Issue 1, Jan. - March 2012.
[8] Sara R.,"Toward an Optimal Algorithm for Matrix Multiplication", From SIAM News, Vol. 38, No. 9,November 2005.

Varsha Thakur received his MCA. from NIT (FORMLY GEC)Raipur Chhattisgarh, India and is currently enrolled as a Ph.D. scholar in the School of Studies of Computer Science and Information Technology , Pt. Ravishankar Shukla University Raipur, Chhattisgarh , India. She is having 6 Year Teaching Experience. Her research area is Load balancing methodologies in parallel and distributed computing.

Dr. Sanjay Kumar received his B. E from Govt.Engg.college, M.E. (Computer Science & Engg.) from Motilal Nehru Regional Engineering College, Allahabad in 2000 and Ph.D. (Computer Science & Engg.) from Ravishankar Shukla University, Raipur in 2005. He is presently serving as Associate professor and Head, Computer Science and Information Technology School of Studies, Pt.Ravishankar Shukla University Raipur Chhattisgarh, India. He has about 15 years of experience in teaching and 6 yr in others. His is current research interest includes Networking and parallel and distributed.