International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 4 Issue 2, February 2020 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Comprehensive Performance Evaluation on Multiplication of Matrices using MPI

Adamu Abubakar I[1], Oyku A[2], Mehmet K[3], Amina M. Tako[4]

[1,2,3]Computer and Electrical Electronics Engineering, Cyprus International University, Adana, Turkey
[4]Mathematics and Computer Science Department, Ibrahim Badamasi BABANGIDA University, Lapai, Nigeria

ABSTRACT
In Matrix-multiplication we refer to a concept that is used in technology applications such as digital image processing, digital signal processing and graph problem solving. Multiplication of huge matrices requires a lot of computing time as its complexity is O (n3). Because most engineering science applications require higher computational throughput with minimum time, many sequential and analogue algorithms are developed. In this paper, methods of matrix multiplication are elect, implemented, and analyzed. A performance analysis is evaluated, and some recommendations are given when using open MP and MPI methods of parallel of latitude computing.

Keywords: Message Passing Inteface, Performance evaluation, Matrix multiplication and Efficiency

How to cite this paper: Adamu Abubakar I | Oyku A | Mehmet K | Amina M. Tako "Comprehensive Performance Evaluation on Multiplication of Matrices using MPI" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-4 | Issue-2, February 2020, pp.833-838, URL: www.ijtsrd.com/papers/ijtsrd30015.pdf

IJTSRD30015

1. INTRODUCTION

Since the origin of parallel equipment and programming advances exploiter s are looked with the chaluminum lenge to pick an arrangement ming worldview most appropriate for the hidden PC design. With the current trgoal in line of scope of scope PC engineering towards group s of shared PC memory symmetric multi-proocessor (SMP) parallel programming systems have developed to help correspondence past a pro dimension. Simple programming inside one SMP hub can exploit the universally shared location space. Compiler for shared memory design more often than not bolster multi-strung execution of instrument of a program. Circle level parallelism can be abused by utilizing aggregating program order, for example, those characterized in the OpenMP standard (Dongarra et al, 1994; Alpatov et al, 1997). OpenMP gives a groin - and-join execution display in which a program Begin execution as a solitary string. This string executes successively until a parallelization mandate for a parallel district is discovered (Alpatov et al, 1997; Anderson et al, 1987). At this meter , the string makes a group of screw string and turns into the ace string of the new group (Chtchelkanova et al, 1995; Barnett et al, 1994; Choi et al, 1992). All strings execute the announcements until the remainder of the parallel district. Work - sharing mandates are given to isolate the execution of the encased code area among the strings. All strings need to synchronize toward the finish of parallel origination . The upside of OpenMP (vane ref.) is that a current code can be effortlessly parallelized by putting OpenMP mandates around sentence devouring circles which don't contain information conditions, leaving the source code unaltered. The disservice is that it is difficult for the client to enhance work process and memory get to. On a SMP bunch the topic passing programming worldview can be utilized inside and over a few hubs. MPI is a broadly acknowledged standard for message passing projects.

The key advantage of the MPI display is its entire authority over conveyance of information and simple synchronization, along these lines permitting real streamlining of information and relating work process.

The real misfortune is its successive applications that particularly needs an extensive and constrained measure of situating for precise parallelization particularly if its MPI-based.

1.1. Matrix-multiplication in serial (sequence)

It involves (2) two matrices A and B where number of columns of A and rows of B are same. If computed in sequential, it takes a time Say; O(n^3) and applying the algorithm:

```
for i=1 to n
for j=1 to n
c(i,j)=0
for k=1 to n
c(i,j)=c(i,j)+a(i,k)*b(k,j)
end
end
end
```

1.2. Matrix Multiplication in Parallel (OpenMp)
The Master node thread uses the outer loop between the slave nodes , thus each of these threads implements the matrix multiplication using a part of rows from the first matrix, when the threads multiplication are done the superior thread joins the sum result of the multiplied matrix.

1.3. Using MPI
The methodology of actualizing the consecutive calculation in parallel utilizing MPI can be isolated into the accompanying:

- Split the main ground substance push savvy to part to the distinctive C.P.U. s, this is performed by the school overlord processor.
- Broadcast the second framework to all centralized server.
- Each processor performs increase of the suggestion of the main lattice and the second grid.
- Each processor sends back the suggestion Cartesian item to the ace processor.

Usage

- the ace with (processor 0) peruses an information,
- the ace at that point sends the measure of information to all slaves,
- the slaves will distribute a memory,
- the ace next communicates a second lattice to a portion of alternate processors,
- the ace sends a few sections of first grid to different processors as needs be,
- all included processor plays out the underlying activity i.e duplication, the slave processors (all) send back their outcomes for ace to print.

2. Methodology and Specification of Tools used
One station with Pentium ifivesome processor with 2 .5 Gigacycle for every second and 4 GB memory is utilized to actualize sequential ground substance proliferation . Visual Studio loft 2008 with openMP store library is utilized as an environs for building, executing and testing intercellular substance duplication PC program . The program is tried utilizing Pentium i5 processer with 2.5 GHz and 4 GB memory.

A conveyed handling association with various phone number of processors is utilized, every processor is a 4 center processor with 2.5 Megahertz and 4 GB memory, the processors are associated however Visual Studio 2008 with MPI condition.

3. Cases
Variety sets of two matrices are selected (different sizes & data types) with each pair of matrix multiplied both in serial and parallel using both openMP and MPI domains, at the end then the average multiplication time is computed.

3.1. Case 1
Sequential matrix propagation program is tested using different size matrices. Different size matrices with different information case (integer, float, double, and complex) are chosen, 100 types of matrices with different data types and different sizing are multiplied. Table 1 display the norm results obtained in this experiment.

Table 1: Experiment 1 Results

Matrices size	Multiplication time in seconds
10*10	0.00641199
20*20	0.00735038
40*40	0.0063971
100*100	0.0142716
200*200	0.0386879
1000*1000	6.75335
1200*1200	11.889
5000*5000	2007
10000*10000	13000

3.2. Case 2
The times program is tested using small size matrices. Different size matrices with different data character (integer, air bladder, doubling , and complex) are chosen , 200 type of matrices with different data types and different size of it are multiplied using open MP environment . Table 2 shows the average resultant obtained in this experimentation.

Table 2: Experiment 2 Results

# of threads	10,10 (time in seconds)	20,20 (time in seconds)	40,40 (time in seconds)	100,100 (time in seconds)	200,200 (time in seconds)
1	0.00641199	0.00735038	0.0063971	0.0142716	0.0386879
2	0.03675360	0.06866570	0.0370589	0.0373609	0.0615986
3	0.06271470	0.06311360	0.0701701	0.0978940	0.0787245
4	0.07273020	0.06979990	0.0710032	0.0706766	0.079643
5	0.06772930	0.07232620	0.0673493	0.0699920	0.051531
6	0.06918620	0.07037430	0.0707350	0.0724863	0.0837632
7	0.07124480	0.07204210	0.0727263	0.0727355	0.0820219
8	0.74631600	0.07348000	0.0677064	0.0762404	0.0820226

3.3. Case 3
The matrix multiplication program is tested using big size matrices. Different size matrices with different information eccentric (integer, float, stunt man , and complex) are elect , 200 types of matrices with different data types and different sizes are multiplied using openMP environment with 8 thread . Table 3 shows the average results obtained in this experiment.

Table 3: Experiment 3 Results

Matrices size	Multiplication time (in seconds)
1000,1000	1.8377
1200,1200	3.19091
2000,2000	18.0225
5000,5000	508.1
10000,10000	333.3

3.4. Case 4

Matrix program is again tested using big size matrices. Different size matrices with different data type (integer, float, double, and complex) are chosen, 200 types of matrices with different data types and different size of it are multiplied using MPI environs with different number of mainframe . Table 4 shows the average results obtained in this experiment

Table 4: Experiment 4 Results

Number of processors	Multiplication time in second 1000x1000 matrices	Multiplication time in second 5000x5000 matrices	Multiplication time in second 10000x10000 matrices
1	6.96	2007	13000
2	5.9	1055	7090
4	3.3	525	3290
5	2.8	431	2965
8	2.1	260	1920
10	1.5	235	1600
20	0.8	119	900
25	0.6	91	830
50	0.55	52	292

4. Discussion of the Simulations

From the gotten qualities in the past area we can classify the grids into (3):

- Small estimating lattices with measuring not as much as thou *unity 000
- Mid size grids with 1000*1000 ≤ estimate ≤ quint 000*5000
- Huge size lattices with size ≥ 5000*5000
- The accompanying suggestion can be announced:
- For little size grids, it is desirable over utilize successive lattice age .
- For medium size networks, it is desirable over utilize parallel lattice times utilizing openMP.
- For colossal size networks, it is desirable over utilize parallel lattice duplication utilizing MPI.
- Also it is prescribed to utilize cross breed parallel association (MPI with openMP) to duplicate grids with tremendous sizes.

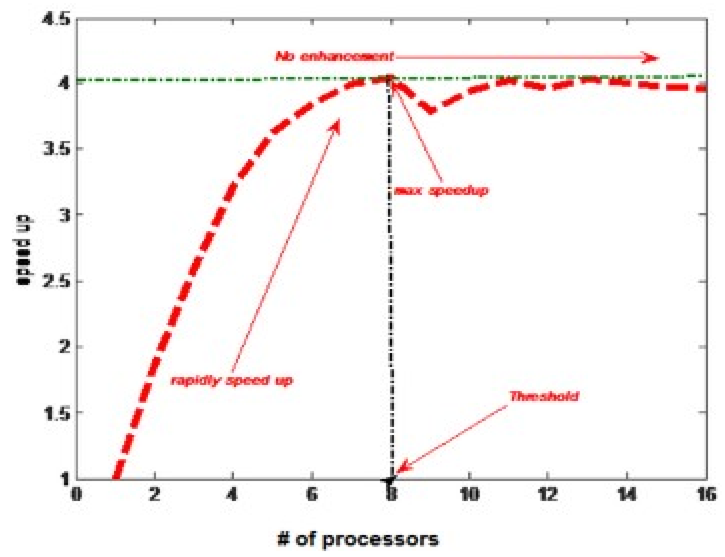
From the outcomes acquired in table 2 we can see the speedup of utilizing openMP is constrained to the quantity of real accessible physical impact in the PC framework as it is appeared in Table 5 and Fig. 1 separately.

Speed-up (i.e time) = the time of execution in 1 thread/but parallel time.

Table 5: Speedup results of Using OpenMP

Matrix size	#of threads = 1	#of threads = 8	Speedup
300,300	0.110188	0.097704	1.1278
400,400	0.314468	0.170006	1.8497
500,500	0.601031	0.277821	2.1634
600,600	1.14773	0.64882	1.7689
700,700	2.17295	0.704228	3.0856
800,800	3.16512	0.963983	3.2834
900,900	4.93736	1.37456	3.5920
1000,1000	6.69186	1.8377	3.6414
1024,1024	7.18151	1.97027	3.6449
1200,1200	12.0819	3.19091	3.7863
2000,2000	72.8571	18.0996	4.0253
2048,2048	74.7383	18.8406	3.9669

Figure 1: Maximum performance of using openMP



From the values obtained in tables 1 & 2 we see that our matrix multiplication time increased very fast when matrix size increases as we can see in figures 2 & 3

Figure 2: Comparing between 8 and 1 threads results

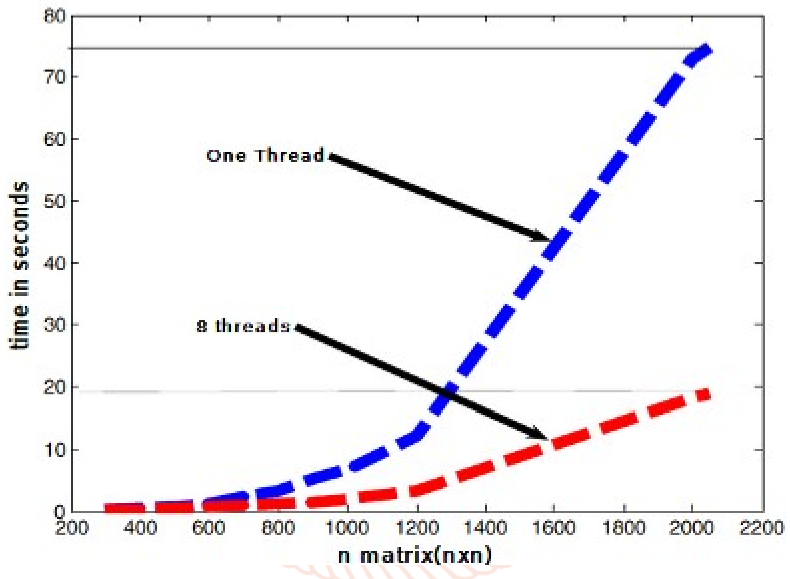
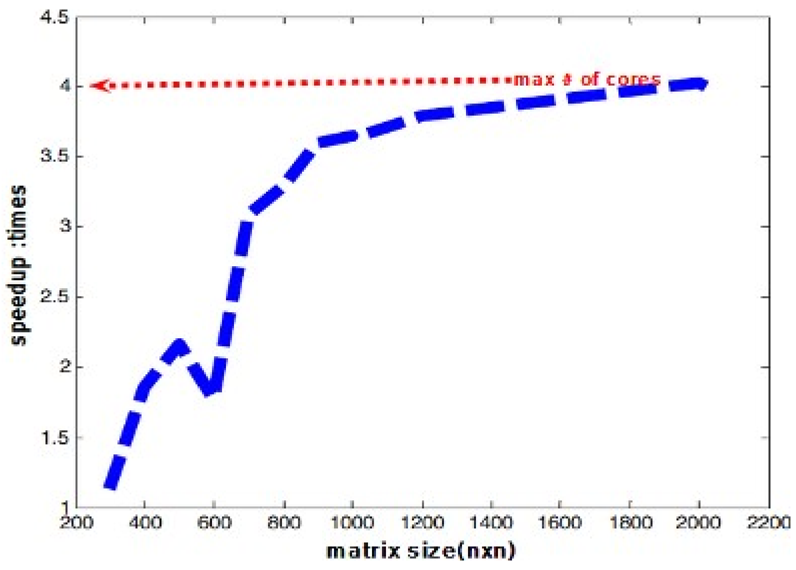


Figure 3: Relationship between the speedup and the matrix size



Results in Table 4 shows that we can ompute the time/speedup using the MPI model and the system efficiency is determined thus:

System Efficiency = the Speedup/no of processors

The obtained results are given in Table 6:

Table 6: Speedup and efficiency of using MPI

Number of processors	Speedup of Multiplication 1000x100 matrices	Efficiency	Speedup of Multiplication 5000x5000 matrices	Efficiency	Speedup of Multiplication 10000x10000 matrices	Efficiency
1	1	1	1	1	1	1
2	1.17	0.585	1.9	0.38	1.83	0.92
4	2.9	0.725	3.8	0.95	3.9	0.98
5	2.46	0.492	4.66	0.93	4.4	0.88
8	3.29	0.411	7.72	0.96	6.77	0.85
10	4.6	0.46	8.54	0.85	8.13	0.81
20	8.63	0.43	16.87	0,84	14.44	0.72
25	11.5	0.45	22.05	0.88	15.66	0.63
50	12.55	0.251	38.6	0.77	44.52	0.89

In table 6, increase in the no of processors in MPI doamin allows the speed-up of Matrix- Multiplication and gives access to poor system efficiency snapshots in figure 4, 5 & 6.

Figure 4: Multiplication time for 10000x10000 matrices

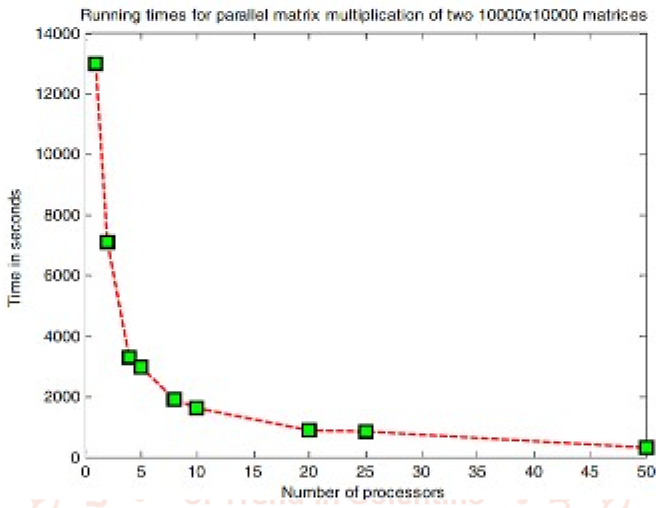


Figure 5: Speedup of multiplication for 10000x10000 matrices

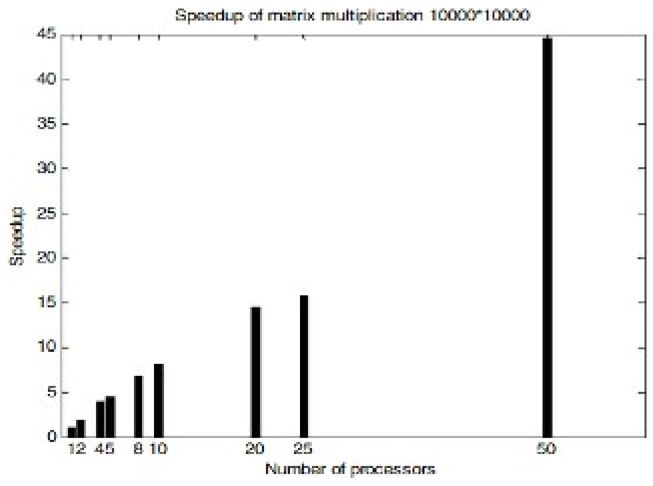
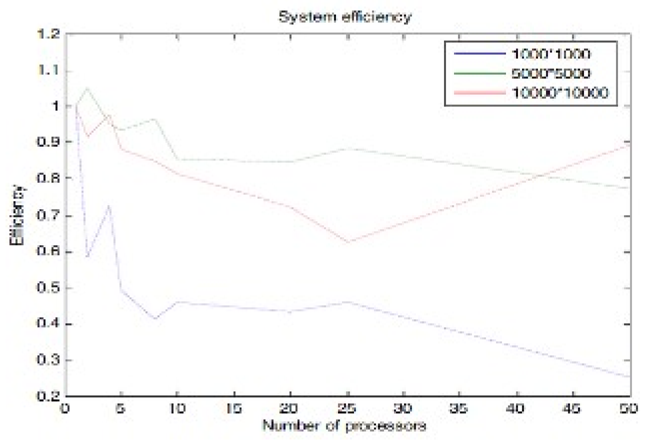


Figure 6: System efficiency of matrices multiplication



5. Conclusions

After a watchful report we can state that consecutive augmentation of network is achievable for frameworks with little length sizes. As it were the Open-MP is an ideal technique to utilize while figuring parallel framework augmentation that has some great number of sizes, the speedup is accessible to the quantity of input physical no of centers. Message Passing is best utilized in the parallel framework duplication with the grids that has enormous sizes, there is plausibility that we can expand the speedup however might be a burden and will clearly influence the framework's execution and effectiveness. As a method for future work we can prescribe a half and half parallel framework just to stay away from the issues in more mind boggling computational issues and get remarkable discoveries.

References

[1] Chen Guoliang, An Hong, Chen Ling, Zheng Qilong, Shan Jiulong, Parallel Algorithm Practice, Beijing: Higher Education Press, pp. 24, 2004. Show Context

[2] J. Michael, Quinn parallel programming in C with MPI and Open MPI, Beijing: Tsinghua University Press, pp. 274-281, 2005.

[3] Zhou Weiming, Multi-core computing and programming, Wuhan: Huazhong University of Science and Technology Press, pp. 92-94, 2009.

[4] Dou Zhihui, High Performance Computing of parallel programming techniques-MPI parallel programming, Wuhan: Huazhong University of Science and Technology Press, pp. 92-94, 2009.

[5] Zhang Linbo, Chi Xuebin, Mo Zeyao, Li Ruo, Introduction to Parallel Computing, Beijing: Tsinghua University Press, pp. 164, 2006.

[6] Beijing: Tsinghua University Press, pp. 168-191, 2007.