A Comparative Evaluation of Parallel Matlab and OpenMP Parallel Technologies

Abhay B. Rathod Sanjay M. Gulhane Vivek L. Faye Ketan A. Fulkar

***Abstract*— This paper present a comparison of an OpenMP and Parallel Matlab based on the parallel implementation of algorithm from the field of computer and mathematical applications. The focus of our study is on the performance of benchmark comparing OpenMP and Parallel Matlab.**

We take mathematical matrix multiplication problem execute it on OpenMP and Parallel Matlab. Based on this example, we conclude that OpenMP is the more suitable than Parallel Matlab due to its fast loading and speed of program. Another reason to consider OpenMP is freely available whereas Matlab we have to purchase. Therefore, for uses in research, OpenMP maturity and resulting richness of functions make it a viable alternative to Matlab.

In our simulation, we used Ubuntu 12.04 and Windows 8 operating system; a system with Intel Core i5 Dual core processor having thread count 4.

***Index Terms*—: OpenMP, Parallel MATLAB, Multicore, SPMD, Parallel Computing Toolbox**

I. INTRODUCTION

We are living in the era of Dual core, Quad-core, multi-core and many-core processors and our Desktop and Laptop computers are equipped with these processors. The question raised by the author in reference [2] What the parallel-processing Community has (failed) to offer the multi/many-core Generation". With the advent of multicore processors [14], it has become imperative to write parallel programs if one wishes to exploit the next generation of processors. So we should start thinking in parallel and execute our serial program in parallel.

There are various tools available for running the program in parallel we are considering only Parallel Matlab and OpenMp .

MATLAB is a popular choice for algorithm development in signal, image and numerical processing. While traditionally done using sequential MATLAB running on desktop systems in recent years there has been a surge of interest in running MATLAB in parallel to take advantage of multiprocessor and multicore systems [3].

OpenMP is a set of compiler directives and runtime library routines that can be used to extend FORTRAN and C to express shared memory parallelism [16].

In Section II introduces the parallel programming environment OpenMP [10], Parallel MATLAB [13] and discuss Multicore processor [9]. Experimental results are presented in Section III. Section IV presents future score and conclusion..

II. PRELIMINARIES

This section gives a brief introduction to Parallel MATLAB, OpenMP and Multicore.

A. Parallel MATLAB

MATLAB have a built-in interpreted language that is similar to C and HPF, and the flexible matrix indexing makes it very suitable for programming matrix problems [31]. Parallel MATLAB has been actively developed over the past several years, and there are several commercial and academic versions available in the market [3]. MATLAB R2010 comes with Parallel Computing Toolbox (PCT) and the Matlab Distributed Computing Server (MDCS).

Parallel computing in MATLAB split up the problem across multi core or multiple compute nodes in a variety of ways.

The parfor() Command

The PCT provides a simple way to parallelize MATLAB for-loops. The parfor () [3] command can be used to distribute the individual loop iterations across processors or cores without any additional code modifications. parfor () divides the loop iterations into groups so that each worker executes some portion of the total number of iterations. parfor-loops are also useful when you have loop iterations that take a long time to execute, because the workers can execute iterations simultaneously. You cannot use a parfor-loop when an iteration in your loop depends on the results of other iterations. Each iteration must be independent of all others [13]. Figure 1 shows the use of parfor command in Matlab using Matlab workers.

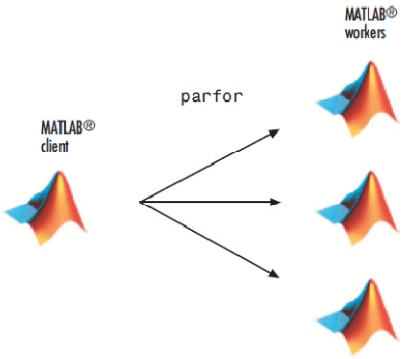

Fig. 1. Use of parfor command [13]

The general form of an parfor statement is:

parfor ii= 1:N

<statements>

end

The spmd() Command

For parallel matrix multiplication in Parallel Computing Toolbox of MATLAB we use spmd command which is data parallel command. The single program multiple data (spmd) construct lets you define a block of code that runs in parallel on all the workers (workers) in the MATLAB pool. The spmd block can run on some or all the workers in the pool [13].

The body of a parfor-loop cannot contain an spmd statement, and an spmd statement cannot contain a parfor-loop [13].
The general form of an spmd statement is:
spmd
<statements>
end
MATLAB supports multithreading natively which can be broadly compared to the OpenMP [3] approach to parallelism.

B. OpenMP

Over the last decade, the Message Passing Interface (MPI) and OpenMP have become the two most dominant parallel programming models [1]. OpenMP is an Application Programming Interface (API) for developing multi-threaded applications in C/C++ or Fortran. OpenMP is a shared memory model of parallel hardware. Multi-core technology offers very good performance and power efficiency and OpenMP has been designed as a programming model for taking advantage of multi-core architecture[25]. So we use the OpenMP programming environment to parallelize Matrix Multiplication in multicore architectures.

Within the multi-core architecture there are varieties of parallel languages that can be utilized to run the parallel programs. The OpenMP is one of the most important libraries that could be applied successfully to run the parallel applications. Comparing with other parallel languages, OpenMP shows efficacy when it has been used with multi-core architecture processors [1].

In general OpenMP follows a fork-join execution model as shown in Fig. 2.

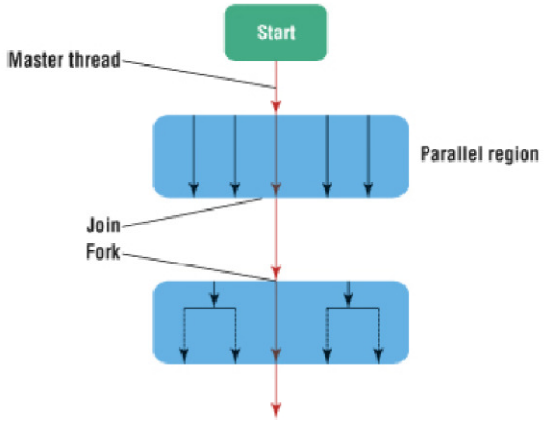

Fig. 2 Fork-join programming model of OpenMP [16].

OpenMP is sequential-coder friendly; that is, when a programmer has a sequential piece of code and would like to parallelize it, it is not necessary to create a totally separate multicore version of the program. Instead of this all-or-nothing approach, OpenMP encourages an incremental approach to parallelization, where programmers can focus on parallelizing small blocks of code at a time. The API also allows users to maintain a single unified code base for both sequential and parallel versions of code. Compiler directives allow programmers to specify which instructions they want to execute in parallel and how they would like the work distributed across cores. OpenMP directives typically have the syntax
"#pragma omp construct [clause [clause]…]."

C. Multicore

Microprocessor Microprocessor architecture has entered the multicore era [12]. Currently, there are two examples of multicore architectures: conventional multicore CPUs (typically with two eight-cores) and unconventional multicore processors such as GPGPUs (with tens or hundreds of cores) [3]. For the test case, we utilized multiple cores CPU.
Multicore architectures integrate multiple processing units into one chip to overcome the physical constraints of unicore architectures, and their exponentially growing power consumption [8].
Multicore architectures provide a new dimension to scale up the number of processing elements (cores) and, therefore, the potential computing capacity. Leading manufacturers are developing processor having more than two cores. IBM's Cell processor has eight cores (plus a master core). Sun Microsystems T2 processor also has eight cores, AMD's mainstream processors - Phenom families announced in 2007, have only four cores and the Dunnington processor announced by Intel in 2008 has six cores [8].

III. EXPERIMENTAL RESULTS

In this section, we present experimental results. For experimental setup, we have tested our system on four test cases. We compare the performance of sequential and parallel Matrix Multiplication with the OpenMP code on a multi-core CPU operating system is Ubuntu 12.4 operating system. The Matrix Multiplication problem again compared with the parallel computing toolbox of Matlab 2010 operating system is Windows 8.
The host machine used has Intel Core i5 1.6 GHz Dual processors. Each processor has Hyper-Threading [19] technology such that, each processor can execute simultaneously instructions from two threads. Overall numbers of cores are 2 and because of hyper threading thread count of host machine equal 4.
In 1988, Gustafson introduced the concept of scalable computing and the fixed-time speedup model [8]. The fixed-time speedup is defined as:

$$\text{Speedup} = \frac{\text{Sequential Time of Solving Scaled Workload}}{\text{Parallel Time of Solving Scaled Workload}}$$

Speedup > 1

A. Matrix Multiplication:
We consider the problem of computing the product C = A*B of two large, dense, matrices. A straight forward matrix

multiplication performs scalar operations on data items. We choose matrix multiplication, because of following two reasons [18]:

- Matrix Multiplication is widely used in numerical programming.
- Matrix Multiplication is a fundamental parallel algorithm with respect to data locality, cache coherency etc.

  . It has a built-in interpreted language that is similar to C and HPF, and the flexible matrix indexing makes it very suitable for programming matrix problems [7].

TABLE 1
EXECUTION TIME FOR MATRIX MULTIPLICATION OF OPENMP AND PARALLEL MATLAB.

Matrix Order	Parallel Matlab		OpenMp	
M*N	Sequential	Parallel	Sequential	Parallel
2*2	0.0001s	0.7650s	0.000054s	0.000401s
3*3	0.0000s	0.5433s	0.000057s	0.000336s
4*4	0.0000s	0.5178s	0.000062s	0.000450s
10*10	0.0626s	0.5328s	0.000109s	0.000498s
100*100	0.0007s	0.6184s	0.018409s	0.010792s
1000*1000	0.4911s	1.2486s	9.803334s	6.980204s
3000*3000	11.643s	12.1814s	410.45334s	233.38s
5000*5000	28.1361s	37.9337s	2084.1065s	1213.364s

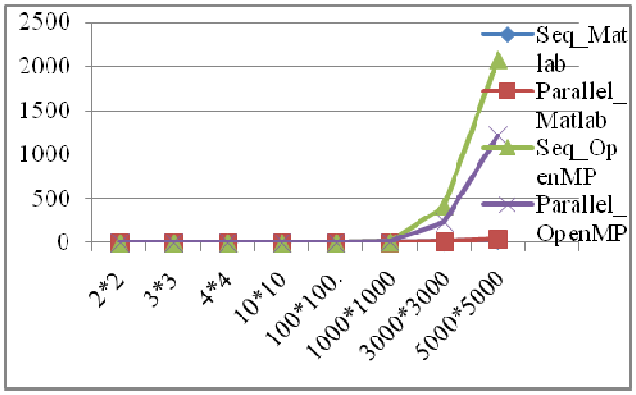

Fig.3 Execution Time for Matrix Multiplication of OpenMP and Parallel MATLAB

Speed-Up Comparison:

TABLE2
SPEED-UP OF SEQUENTIAL TO OPENMP & PARALLEL MATLAB

MATRIX ORDE	SEQ-TO-PARALLEL MATLAB	SEQ-TO-OPENMP
2*2	0.0001	0.1346
3*3	0.0000	0.1696
4*4	0.0000	0.1377
10*10	0.1174	0.2188
100*100	0.00113	1.7058
1000*1000	0.3933	1.4044
3000*3000	0.9555	1.7587
5000*5000	0.7417	1.7176

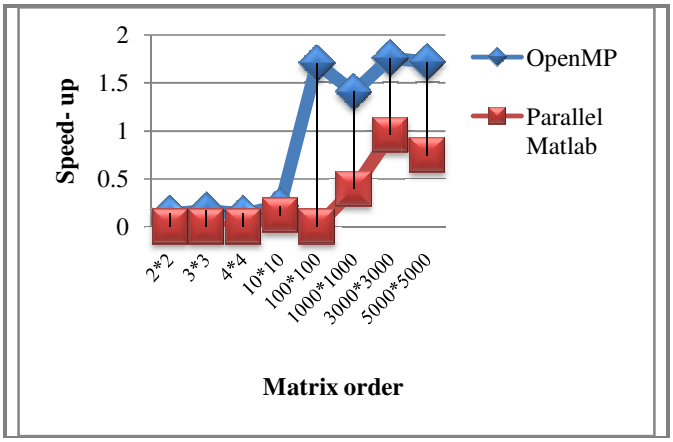

Fig. 4: Speed-Up Comparison

Journal Impact Factor-2.02
Indexed by- ProQuest, DOAJ and many International Bodies.

As the dimensions of the matrix increase, the execution time for sequential algorithm also increases by manifold as shown on Figure 3. After analyzing Table -1 and 2, Figure 3 and 4, we conclude that, given the Multi-core architecture, OpenMP shows good improvements for large matrix dimensions and gives very good Speed-Up factor and very less execution time. This can be evident from Table 1 & 2 that for matrix multiplication OpenMP is much better than Parallel Matlab.

IV. CONCLUSION

We studied the behavior of parallel algorithms with respect to OpenMP and Parallel Matlab. The initial results we found were not satisfactory. But, as the number of input data size increased OpenMP gives good performance.

As the systems are equipped with multi-core architecture. So, OpenMP will be a viable option for cases such as matrix multiplication and other applications..

Parallel Matlab takes lot of time to start the matlabpool option and consume more space on hard disk and make system slower. All this adds overhead in Parallel Matlab.

Overall, we sum up our conclusion as

OpenMP > Sequential>Parallel Matlab

Where > indicates performance. As a future work, we will find algorithms, where OpenMP is more preferable over Parallel Matlab.

Future research work is required in the following problem areas: given an application program, we must check how useful OpenMP or Parallel Matlab is in heterogeneous environment consisting of multiple GPUs and multi-cores. Secondly, a library routine can be developed, which will port application program to CPU using OpenMP or Parallel Matlab or combination of these two technologies.

REFERENCES

[1] Burrows, Eva, and Magne Haveraaen. "A hardware independent parallel programming model." The Journal of Logic and Algebraic Programming 78.7 (2009): 519- 538.

[2] Träff, Jesper Larsson. "What the parallel-processing community has (failed) to offer the multi/many-core generation." Journal of Parallel and Distributed Computing 69.9 (2009): 807-812.

[3] Samsi, Siddharth, Vijay Gadepally, and Ashok Krishnamurthy. "Matlab for signal processing on multiprocessors and multicores." Signal Processing Magazine, IEEE 27.2 (2010): 40-49.

[4] Ganegoda, G. A. C. P., et al. "Jconqurr-a multi-core programming toolkit for java." International Journal of Computer and Information Engineering 3.4 (2009): 223-
230.

[5] M. Okrouhl´ık" Numerical methods in computational mechanics "Institute of Thermomechanics, Prague 2008 Last revision January 23, 2012

[6] Qiu, Xiaohong, et al. "High performance multi-paradigm messaging runtime integrating grids and multicore systems." e-Science and Grid Computing, IEEE International Conference on. IEEE, 2007.

[7] Ron Choy , Alan Edelman Computer Science AI Laboratory,"Parallel MATLAB: Doing it Right"Massachusetts Institute of Technology, Cambridge, MA 02139.

[8] Xian-He Sun_, Yong Chen "Reevaluating Amdahl's law in the multicore era" Computer Scienc Department, Illinois Institute of Technology, United States.

[9] Zhou, Zheng, et al. "Multicore Programming Guide." Signal Processing (ICSP), 2012 IEEE 11th International Conference on. Vol. 1. IEEE, 2012.

[10] Duran, Alejandro, et al. " OpenMP Application Program Interface Version 3.0 May 2008." Paralle Processing, 2009. ICPP'09. International Conference on. IEEE, 2009.

[11] Beck, Micah, Richard Johnson, and Keshav Pingali. " Parallel Programming Languages." Journal of Parallel and Distributed Computing 12.2 (1991): 118-129.

[12] Aiex, R. M., Martins, S. D. L., Ribeiro, C. C., & Rodriguez, N. D. L. R. (1998). Cooperative multi-thread parallel tabu search with an application to circuit partitioning. In Solving Irregularly Structured Problems in Parallel (pp. 310-331). Springer Berlin Heidelberg.

[13] Parallel Computing Toolbox™ User's Guide R2012a

[14] Park, Sungwoo, et al. "Parallel skyline computation on multicore architectures." Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on. IEEE, 2009.

[15] Wang, Lihui, and Xianchao Xu. "Parallel Software Development with Intel Threading Analysis Tools." Intel Technology Journal 11.4 (2007).

[16] Marowka, Ami. "Think parallel: Teaching parallel programming today." Distributed Systems Online, IEEE 9.8 (2008): 1-1.

[17] Sharma, Neeraj, and Matthias K. Gobbert. "A comparative evaluation of Matlab, Octave, FreeMat, and Scilab for research and teaching." Department of Mathematics and Statistics, University of Marylan Baltimore County, Technical Report HPCF–2010–7 (2010).

[18] Thouti, Krishnahari, and S. R. Sathe. "Comparison of OpenMP & OpenCL Parallel Processing Technologies." arXiv preprint arXiv:1211.2038 (2012)

[19] IntelHyper-Threading Technology, http://www.intel.com