PERFORMANCE COMPARISON OF ROW PER SLAVE AND ROWS SET PER SLAVE METHOD IN PVM BASED PARALLEL MATRIX MULTIPLICATION

Sampath S[1], Nanjesh B R[2], Bharat Bhushan Sagar[3], C K Subbaraya[4]

[1]*Research Scholar, Sri Venkateshwara University, Gajraula, Amroha, Uttarpradesh, INDIA,* ***23.sampath@gmail.com***
[2]*Department of Information Science and Engineering, Adichunchanagiri Institute of Technology, Chikmagalur, Karnataka, INDIA,* ***nanjeshbr@gmail.com***
[3]*Department of Computer Science and Engineering, Birla Institute of Technology, Noida, Uttarpradesh, INDIA,* ***drbbsagar@gmail.com***
[4]*Department of Computer Science and Engineering, Adichunchanagiri Institute of Technology Chikmagalur, Karnataka,, INDIA,* ***subrayack@gmail.com***

Abstract

Parallel computing operates on the principle that large problems can often be divided into smaller ones, which are then solved concurrently to save time by taking advantage of non-local resources and overcoming memory constraints. Multiplication of larger matrices requires a lot of computation time. This paper deals with the two methods for handling Parallel Matrix Multiplication. First is, dividing the rows of one of the input matrices into set of rows based on the number of slaves and assigning one rows set for each slave for computation. Second method is, assigning one row of one of the input matrices at a time for each slave starting from first row to first slave and second row to second slave and so on and loop backs to the first slave when last slave assignment is finished and repeated until all rows are finished assigning. These two methods are implemented using Parallel Virtual Machine and the computation is performed for different sizes of matrices over the different number of nodes. The results show that the row per slave method gives the optimal computation time in PVM based parallel matrix multiplication.

Keywords*: Parallel Execution, Cluster Computing, MPI (Message Passing Interface), PVM (Parallel Virtual Machine) RAM (Random Access Memory).*

----------***----------

1. INTRODUCTION

Parallel processing refers to the concept of speeding up the execution of a program by dividing the program into multiple fragments that can execute simultaneously, each on its own processor. Matrix multiplication is commonly used in the areas of graph theory, numerical algorithms, image processing and aviation. Multiplication of larger matrices requires a lot of computation time. This paper deals how to handle Matrix Multiplication problem that can be split into sub-problems and each sub-problem can be solved simultaneously using two methods of parallel matrix multiplication.

MPI (Message Passing Interface) is specification for message-passing libraries that can be used for writing portable parallel programs. In MPI programming, a fixed set of processes is created at program initialization. Each process knows its personal number. Each process knows number of all processes and they can communicate with other processes. Process cannot create new processes and the group of processes is static [11].

PVM (Parallel Virtual Machine) is a software package that allows a heterogeneous collection of workstations (host pool) to function as a single high performance parallel virtual machine. The PVM system consists of the daemon (or pvmd), the console process and the interface library routines. One daemon process resides on each constituent machine of the virtual machine. Daemons are started when the user starts PVM by specifying a host file, or by adding hosts using the PVM console [12].

This paper deals with the implementation of parallel application, matrix multiplication using recent versions of PVM, under PVM using PVM3.4.6 [12] for communication between the cores and for the computation. Because they are very much suitable to implement in LINUX systems

2. RELATED WORKS

Amit Chhabra, Gurvinder Singh (2010) [1] proposed Cluster based parallel computing framework which is based on the Master-Slave computing paradigm and it emulates the parallel

computing environment. Hai Jin et al (2001) [6] discussed the incentive for using clusters as well as the technologies available for building clusters. and also discussed a number of Linux-based tools such as MPI, PVM etc. and utilities for building clusters. Rafiqul Zaman Khan and Md Firoj Ali (2011) [2] represented the comparative study of MPI and PVM parallel programming tools in parallel distributed computing system. They described some of the features for parallel distributed computing system with a particular focus on PVM and MPI which are mostly used in today's parallel and distributed computing system. Sampath S et al (2012) [3] presented the framework that demonstrates the performance gain and losses achieved through parallel processing and made the performance analysis of parallel applications using this cluster based parallel computing framework. Rajkumar Sharma et al (2011) [5] evaluated performance of parallel applications using MPI on cluster of nodes having different computing powers in terms of hardware attributes/parameters. Cirtek P and Racek S (2007) [4] made the performance comparison of distributed simulation using PVM and MPI in which presented the possibilities of the simulation programs speedup using parallel processing and compared the results from an example experiments.

Eyas El-Qawsmeh et al [7] presented a quick matrix multiplication algorithm and evaluated on a cluster of networked workstations consisting of Pentium hosts connected together by Ethernet segments. Petre Anghelescu [8] showed how the implementation of a matrix multiplication on a network computers can be accomplished using the MPI standard and presented extensive experimental results regarding the performance issues of matrix parallel multiplication algorithms. Various ways of matrix distribution among processors have been described here. Muhammad Ali Ismail et al [9] performed the concurrent matrix multiplication on multi-core processors. This study is a part of an on-going research for designing of a new parallel programming model SPC3 PM for multicore architectures. Ziad A.A. Alqadi, et al [10] conducted the performance analysis and evaluation of parallel matrix multiplication algorithms, In this work, a theoretical analysis for the performance and evaluation of the parallel matrix multiplication algorithms is carried out. However, an experimental analysis is performed to support the theoretical analysis results. Recommendations are made based on this analysis to select the proper parallel multiplication algorithms. In our work we do the comparison of row per slave and rows set per slave method which is implemented using PVM. We show that optimal computation time can be obtained using row per slave method of parallel matrix multiplication.

3. SYSTEM REQUIREMENTS

3.1 Hardware Requirements

- Processor: Pentium D (3 G Hz)
- Two RAM: 256MB and 1GB
- Hard Disk Free Space: 5 GB
- Network: TCP/IP LAN using switches or hubs

3.2 Software Requirements

- Operating System: Linux
- Version: Fedora Core 14
- Compiler: GCC
- Communication protocol: PVM
- Network protocol: Secure Shell

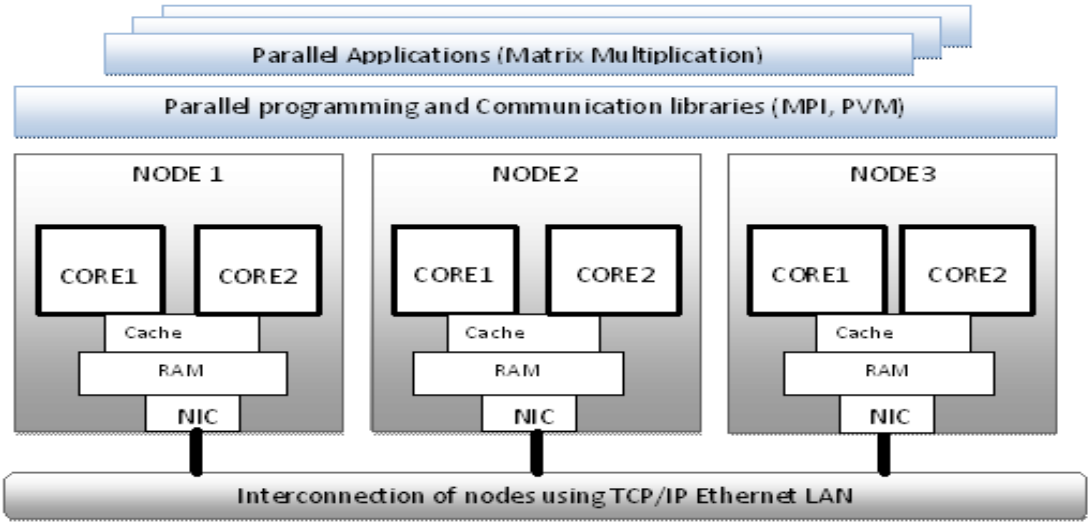

Fig 1: Cluster based parallel computing architecture

4. CLUSTER BASED PARALLEL COMPUTING ARCHITECTURE

Fig.1 shows the cluster based parallel computing architecture involving three nodes over which PVM based parallel applications can run. Desktop PC's are termed here as nodes which are connected together using Ethernet TCP/IP LAN to work as single high performance computing system. Each node contains two cores. Using the capacity of underlying nodes, the processes perform the parallel computation. One of the processes acts as master and remaining processes acts as slaves. For each process unique task ids or number will be generated for identifying processes in the communication world. The main problem is taken by the master process and assigns the task into slaves. Each slave send back the solutions of the assigned task.

5. ROWS SET PER SLAVE METHOD OF MATRIX MULTIPLICATION

The operations involved in Rows set per slave method of matrix multiplication are as follows: Master finds average number of rows to be sent to each slave and extra rows. Then Master finds the number of rows to be sent in a rows set of matrix A and Send the each set of rows along with the offset to the available slaves. Assigning is done serially from first slave to the last slave. Slaves computes the rows set of resultant matrix C and send back the solution. Slave uses entire matrix B and rows set assigned to it for computation. Receiving of solution is done serially from first slave to last slave. Master receives the solution of subtasks from each slave which is the part of resultant matrix C along with the offset. The Row per slave based algorithm for master and slave side operations shown in Fig 2 and 3 respectively.

6. ROW PER SLAVE METHOD OF MATRIX MULTIPLICATION

The operations involved in Row per slave method of matrix multiplication are as follows: Master sends one row of the first matrix (matrix A) and the one count value which varies from 0 to size of matrix-1 to each slave. Slaves as soon they receive a row of first matrix, computes the resultant row of matrix C using received row of matrix A and predefined Matrix B in them. Finally slaves send back the resultant row of Matrix C to master along with their tid and count value. But initially, master starts receiving resultant rows only after the assignment of single row to all available slaves is finished. Master then receives the row of resultant matrix, count value and tid from the process which finished its computation and it is set free. This tid is used for assigning next task to the slave process. Then master copies row just calculated into C matrix in a correct order using count value. This procedure is repeated until all the rows of Matrix A is finished assigning to slaves. The Row per slave methodology of master and slave side operations is shown in Figure 4a and 4b respectively. The operations involved in getting the resultant matrix are shown in Fig. 4.c with simple example.

7. RESULTS AND DISCUSSION

We compared the new row per slave and traditional rows set per slave methods using the parallel computing tool PVM3.4.6. Computation time is taken for different sizes of input matrices and for executions over different number of nodes. Table 1 shows the computation time taken for rows set per slave and row per slave based matrix multiplication using PVM. Comparison of these two methods over single, two and three nodes using PVM are shown in figs 5, 6, 7. The row per slave method of matrix multiplication is taking less computation time compared to rows set per slave method. The rows set per slave method takes more computation time as it does assigning sub tasks and retrieving solution serially and in terms of rows set

```
define matrix a, initialize matrix c
start timer
if taskid ←MASTER
        Initialize input matrices elements
        start timer
        averow ←NRA/numworkers
        extra ← NRA%numworkers
        set offset as zero
        mtype ←FROM_MASTER
        for i ←0 to NPROC
                rows ← i <= extra ? averow+1 : averow
                send offset, matrix A's subset and Matrix B
                offset ←offset + rows;
        end for
        /* Receive results from worker tasks */
        mtype ← FROM_WORKER
        for i ←1 to NPROC
                receive offset, portion of resultant matrix C using PVM_Recv
        end timer
        print resultant matrix C
end if
```

Fig 2: Algorithm for Mater side operations using rows set per slave method

```
define matrix B
get parent id, so we know where to receive from
mtype ←FROM_MASTER
receive offset, matrix A's subsetand Matrix B for k←0 to NCB
        for i←0 to rows
                c[i][k] ← 0.0
                for j←0 to NCA
                        c[i][k] ← c[i][k] + a[i][j] * b[j][k]
                end for
        end for
end for
mtype = FROM_WORKER
send offset, set of rows portion of resultant matrix C
```

Fig 3: Algorithm for Slave side operations using rows set per slave method

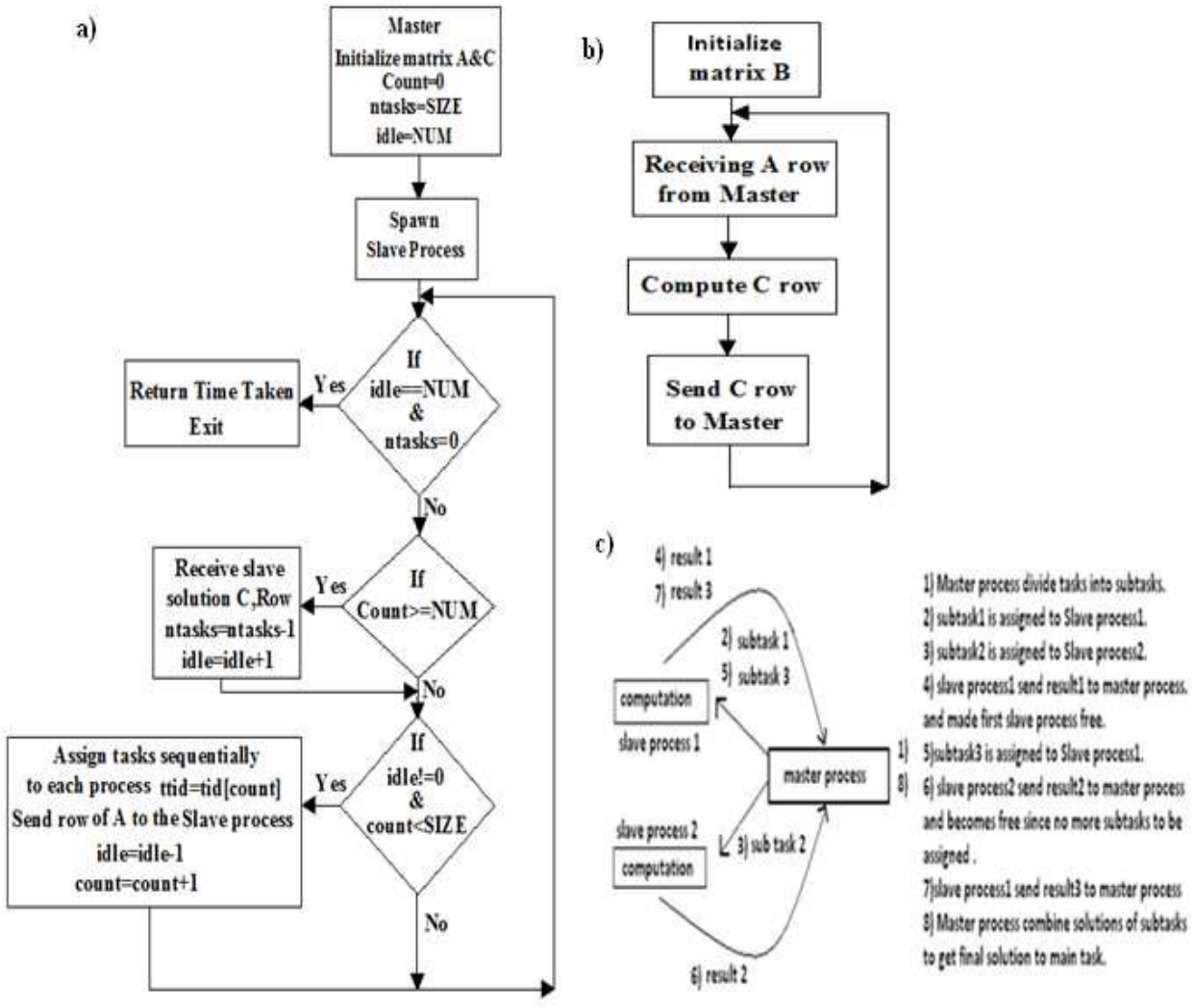

Fig.4.a) Flow diagram for operations involved at the Master side b). Flow diagram for operations involved at the Slave side c). Example (1 task into 3 subtasks computed using 2 slaves) to show Operations involved

Table 1: performance comparison of rows set per slave and row per slave method using pvm(all time in seconds)

Number of Nodes	Type of matrix multiplication	1000*1000	1500*1500	2000*2000	2500*2500	3000*3000
Single Node	Rows set per slave	10.1783	36.2335	86.3251	171.3234	291.5617
	Row per slave	5.4749	28.2361	43.0896	133.6212	157.8411
Two Nodes	Rows set per slave	7.5521	18.4315	45.8543	75.1528	136.2641
	Row per slave	2.8996	12.0456	22.0475	58.4992	80.4918
Three nodes	Rows set per slave	7.6645	19.1214	36.2816	66.9552	108.2531
	Row per slave	1.9545	7.4321	14.6072	31.4274	48.4312

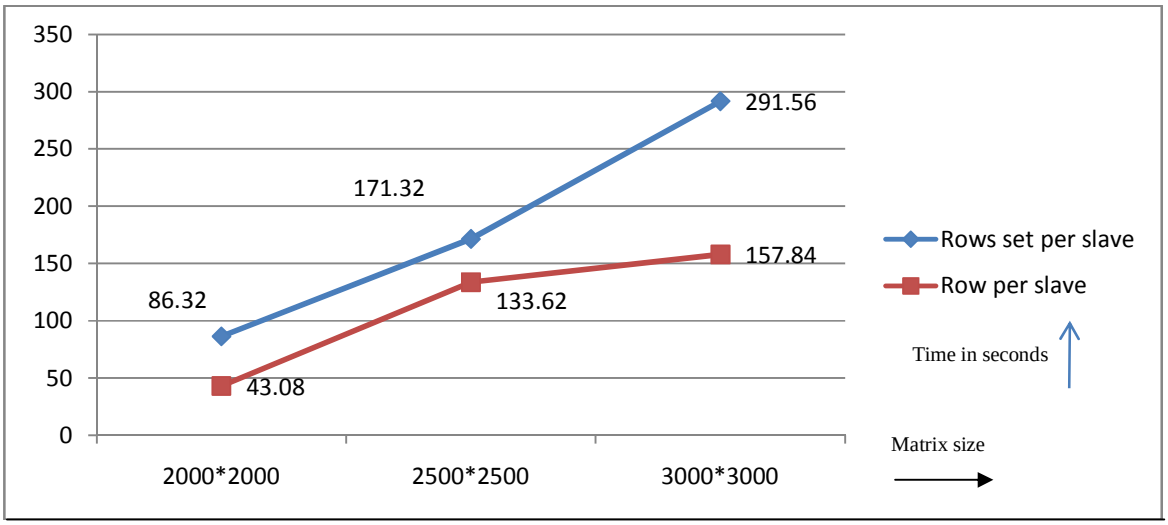

Fig 5: Comparison over single node using PVM

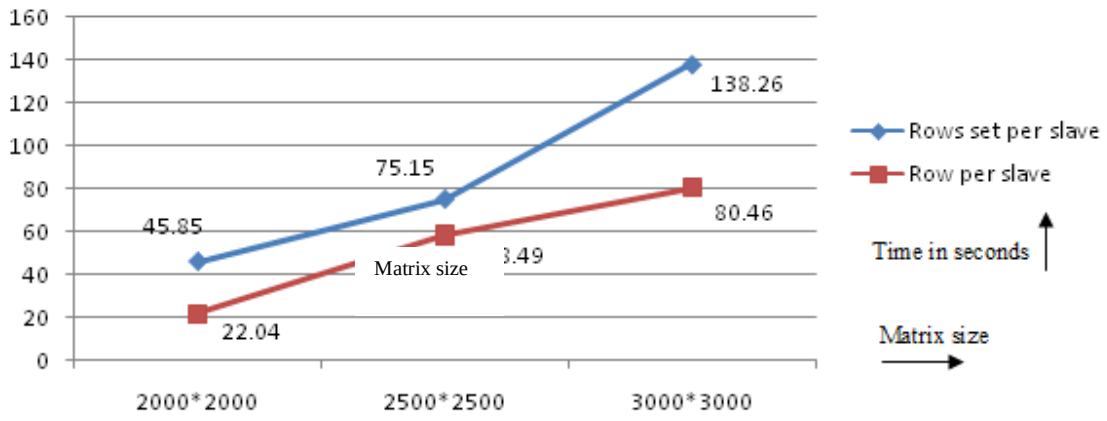

Fig 6: Comparison over two nodes using PVM

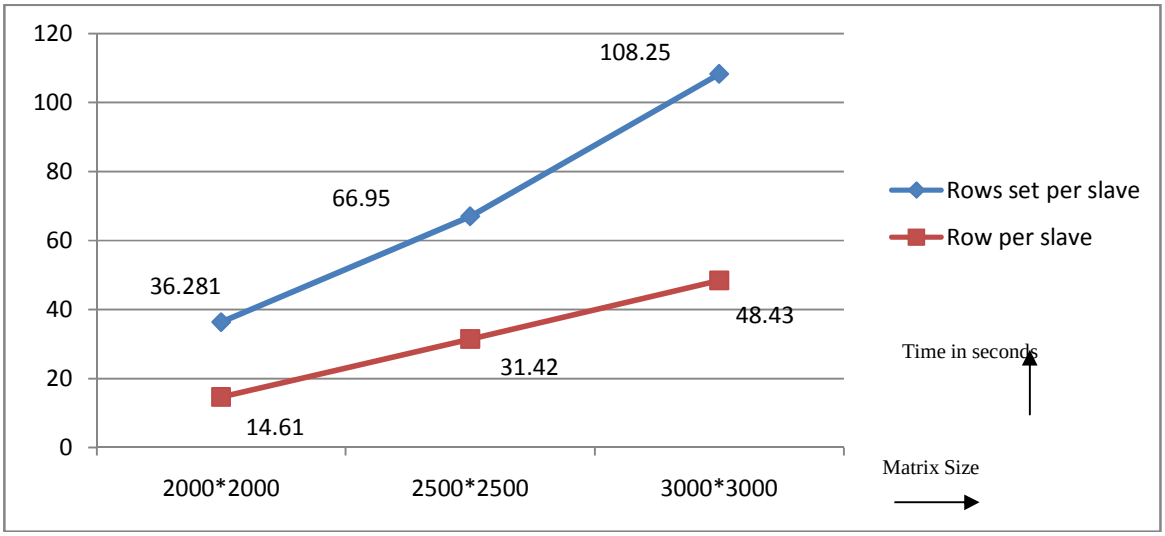

Fig 7: Comparison over three nodes using PVM

Even though the slaves finished computation they must be waiting until their turn to send the solution, comes. But in case of row per slave method the slaves do the computation of only one row at a time and sends back the solution and receives another row for computation.

CONCLUSIONS

The row per slave method is giving more optimal computation time than the rows set per slave method in PVM based parallel matrix multiplication. An average reduction in computation time with row per slave method when compared to rows set per slave method of matrix multiplication is around 50%.

ACKNOWLEDGEMENTS

We express our humble pranams to his holiness SRI SRI SRI Dr|| BALAGANGADHARANATHA MAHA SWAMIJI and seeking his blessings.First and foremost we would like to thank Dr. C.K. Subbaraya, Principal, Adichunchangiri Institute of Technology, Chikmagalur, for his moral support towards completing our work. And also we would like to thank Dr. Mallikarjuna Bennur, for his valuable suggestions given for us throughout our work.

REFERENCES

[1] AmitChhabra, Gurvinder Singh "A Cluster Based Parallel Computing Framework (CBPCF) for Performance Evaluation of Parallel Applications", International Journal of Computer Theory and Engineering, Vol. 2, No. 2 April, 2010.

[2] RafiqulZaman Khan, MdFiroj Ali, "A Comparative Study on Parallel Programming Tools in Parallel Distributed Computing System: MPI and PVM", Proceedings of the 5th National Conference; INDIACom-2011.

[3] Sampath S, Sudeepa K.B, Nanjesh B R "Performance Analysis and Evaluation of Parallel Applications using a CBPCF", International Journal of Computer Science and Information Technology Research Excellence (*IJCSITRE*), Vol.2,Issue 1,Jan-Feb 2012.

[4] Cirtek P, Racek S, "Performance Comparison of Distributed Simulation using PVM and MPI", The International Conference on "Computer as a Tool". Page(s): 2238 – 2241, EUROCON, 2007.

[5] Rajkumar Sharma, Priyesh Kanungo, Manohar Chandwani, "Performance Evaluation of Parallel Applications using Message Passing Interface in Ntework of Workstations of Different Computing Powers", Indian Journal of Computer Science and Engineering(IJCSE), Vol. 2,No. 2, April-May 2011.

[6] Hai Jin, Rajkumar Buyya, Mark Baker, "Cluster Computing Tools, Applications, and Australian Initiatives for Low Cost Supercomputing", MONITOR Magazine, The Institution of Engineers Australia , Volume 25, No 4, Dec.2000-Feb 2001.

[7] Eyas El-Qawsmeh, Abdel-Elah AL-Ayyoub, Nayef Abu-Ghazaleh, "Quick Matrix Multiplication on Clusters of Workstations", INFORMATICA, Volume 15, Issue.2, pages 203–218, 2004.

[8] Petre Anghelescu, "Parallel Algorithms for Matrix Multiplication", 2012 2nd International Conference on Future Computers in Education, Vols.23-24, pages 65-70, 2012.

[9] Muhammad Ali Ismail, S. H. Mirza, Talat Altaf, "Concurrent Matrix Multiplication on Multi-Core Processors", International Journal of Computer Science

and Security, Volume 5, Issue 2, pages 208-220, Feb 2011.

[10] Ziad A.A. Alqadi, MusbahAqel and Ibrahiem M. M. E l Emary "Performance Analysis and Evaluation of Parallel Matrix Multiplication Algorithms", World Applied Sciences Journals, Volume 5, Issue 2, pages 211-214, 2008.

[11] History of PVM versions: http://www.netlib.org/pvm3/book/node156.html.

[12] PVM3.4.6: http://www.csm.ornl.gov/pvm/pvm3.4.6.