Volume 3, Issue 5, May 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Multiple Nodes Based Analysis of Parallel Application's Performance and its Dependency on RAM using PVM

Gopalakrishna C[1], Gagana B G[2], Fathima Ruqsar T S[3], Deepesh R[4], Nanjesh B R[5]
Department of Computer Science and Engineering
Adichunchanagiri Institute of Technology
Chikmagalur, Karnataka, India

Abstract— Parallel computing operates on the principle that large problems can often be divided into smaller ones, which are then solved concurrently which results in saving time to solve larger problems and to provide concurrency using desktop PC's. The main aim is to form a cluster based parallel computing architecture for PVM based applications that demonstrates the performance gain and losses achieved through parallel processing using PVM 3.4.6 under PVM. The architecture for demonstrating PVM based parallel applications works on the Master-Slave computing paradigm. The master will monitor the progress and be able to report the time taken to solve the problem, taking into account the time spent in breaking the problems into sub-tasks and combining the results along with the communication delay. The slaves are capable of accepting sub problems from the master and finding the solution and sending back to the master. We aim to evaluate these statistics of parallel execution for solving matrix multiplication problem and do comparison with the time taken to solve the same problem in serial execution to show the communication overhead involved in parallel execution. The results which are obtained from different number of nodes are compared to evaluate the efficiency of PVM based parallel applications. We also aim to show the performance dependency of multiple nodes based parallel execution, single node parallel execution and serial computation on different sizes of RAM.

Keywords— Parallel Execution, Cluster Computing, Symmetric Multi-Processor (SMP), MPI (Message Passing Interface), PVM (Parallel Virtual Machine), RAM (Random Access Memory).

I. 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. This paper deals how to handle Matrix Multiplication problem that can be split into sub-problems and each sub-problem can be solved simultaneously. With computers being networked today, it has become possible to share resources like files, printers, scanners, fax machines, email servers, etc. One such resource that can be shared but is generally not, is the CPU. Today's processors are highly advanced and very fast, capable of thousands of operations per second. If this computing power is used collaboratively to solve bigger problems, the time taken to solve the problem can reduce drastically.

A. Existing Frameworks

1) PVM 1.0: PVM that have been released from the first one in February 1991. PVM 1.0 cleaned up the specification and implementation to improve robustness and portability [7].

2) PVM 2.X Versions: PVM 2.1 provided with process-process messages switched to XDR to improve portability of source in heterogeneous environments and simple console interpreter added to master pvmd. Later versions belonging to PVM 2.x provided with more and more useful functionalities such as pvmd-pvmd message format switched to XDR, get and put functions vectorized to improve performance, broadcast function deprecated ,improved password-less startup via rsh/rcmdetc [7].

3) PVM 3.X Versions: These allow scalability to hundreds of hosts, allow portability to multiprocessors / operating systems other than Unix, allows dynamic reconfiguration of the virtual machine, allows fault tolerance, includes dynamic process groups, provide the option to send data using a single call [7].

4) PVM 3.4.6: Includes both Windows and UNIX versions and improved use on Beowulf clusters. Also includes the latest patches for working with the latest versions of Linux (like fedora 14), Sun, and SGI systems New features in PVM 3.4.x include communication contexts, message handlers, persistent messages. In our project, we are using PVM 3.4.6 for providing parallel environment using PVM [8].

B. Frameworks used in Proposed System

This paper deals with the implementation of parallel application, matrix multiplication under PVM using PVM3.4.6 for communication between the processes and for the computation. Because it is very much suitable to implement in LINUX systems.

II. RELATED WORKS

Traditionally, multiple processors were provided within a specially designed "parallel computer"; along these lines, Linux now supports SMP Pentium systems in which multiple processors share a single memory and bus interface within a single computer. It is also possible for a group of computers (for example, a group of PCs each running Linux) to be interconnected by a network to form a parallel processing cluster [1]. V.S Sunderam, G.A Geist, J Dongarra, R Manchek (1994) [4] describe the architecture of PVM system, and discuss its computing model, the programming interface it supports, auxiliary facilities for process groups and MPP support, and some of the internal implementation techniques employed. Amit Chhabra, Gurvinder Singh (2010) [5] proposed Cluster based parallel computing framework which is based on the Master-Slave computing paradigm and it emulates the parallel computing environment. Muhammad Ali Ismail, Dr. S. H. Mirza, Dr. Talat Altaf (2011) [6] performed the Concurrent Matrix Multiplication on Multi-Process Processors. Kamalrulnlzam Abu Bakar, Zaitul Mlir lizawati Zal nuddln (2006) [7] made the performance comparison of PVM and RPC. The comparison is done by evaluating their performances through two experiments namely one is a broadcast operation and the other are two benchmark applications, which employ prime number calculation and matrix multiplication. We aim to present a architecture over which PVM based parallel application runs and which demonstrates the performance gain and losses achieved through parallel processing. And also demonstrates the performance dependency of parallel applications on RAM.

III. SYSTEM REQUIREMENT

A. Hardware Requirements

- Processor: Pentium D (3 G Hz)
- Two RAM: 1GB and 2GB
- Hard Disk Free Space: 5 GB

B. Software Requirements

- Operating System: Linux
- Version: Fedora Process 14
- Compiler: GCC
- Communication protocol: PVM

IV. System Design

A. System Analysis

The system is to be designed such that it demonstrates the performance dependency of parallel and serial execution on RAM and also it demonstrates the following:

- How a client can submit the entire problem to a master and collects the solution back from it without bothering about how it has been solved.
- How the master detects the available slaves on the network, and how it detects the system load on that machine to determine whether it is worth sending a task to that particular client.
- How a problem can be submitted to the slaves.
- How the solutions of the given problem can be retrieved from the slave.
- How the slaves solve the given problem.

The design was made modular i.e. the software is logically partitioned into components that perform specific functions and sub-functions.

1) *Master:* It is designed such that it has functionality to manage connection and communication with the slave, It scans and identifies all the processes or slaves available on the node here it is only one slave to be identified. It then assigns the processor ranks to identify the processes. The master assigns the problem to slave. It also has to accept the results sent back by the slave after they finish the computation of the sub-tasks assigned to them. Then the received result has to be assembled in the right order to obtain the solution for the main problem.

2) *Slave*: It is designed to have the functionality to read the problem (in case of single slave)/sub-problem sent by the master, evaluate the problem (in case of single slave)/sub-problem and send the result back to the master.

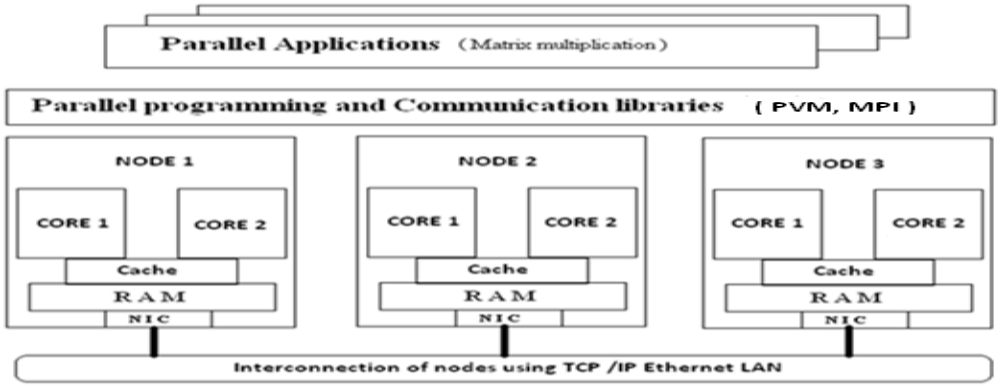

Fig 1. Cluster based parallel computing architecture

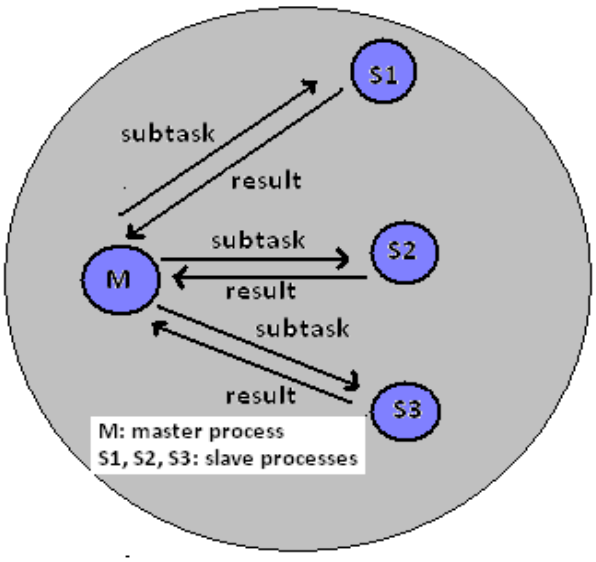

Fig 2. Operations involved in cluster based parallel computing architecture

B. Cluster Based Parallel Computing architecture

The main problem is taken by the master process and assigns the task into slave processes. Each slave process send back the solutions of the assigned sub problem. The working principle involved in this architecture is shown in Fig.2 and Fig.1 shows the cluster based parallel computing architecture.

C. PVM Configuration

Download the software package from http://www.netlib.org/pvg3

Unpacking:

Command: $tar zxvf pvm3.4.6.tgz

Opened the .bhrc file through terminal using vi editor and set the following lines in the file [3] and closed the file.

The .bhrc is a hidden file of course and can be done as:

$home

$ls –a

$vi .bhrc

Going to the insert mode add the following lines as:

PVM_ROOT=$HOME/pvm3

PVM_DPATH= PVM_ROOT/lib/pvmd

Export PVM_ROOT PVM_DPATH

Going to pvm3 directory ($cd pvm3) type make ($make). This would make pvm(the PVm console), pvmd3(the pvm daemon), libpvm3.a(PVM C/C++ library), libfpvm3.a (PVM Fortran library) and libgpvm3.a (PVM group library). All these files would be placed in the $/pvm3/lib/LINUX and pvmgs (PVM group server) would be placed in $/pvm3/bin/LINUX. Open .rhosts file in the home directory ($ vi .rhosts) and added the name of the node (computer) in the cluster of four computers.

Set the following environment variables in .bashrc file as:

export PVM_ARCH='$PVM_ROOT/lib/pvmgetarch'

export PVM_ROOT= $HOME/pvm3/xpvm

export PATH =$PATH:$PVM_ROOT/lib

export PATH==$PATH:$PVM_ROOT/lib/$PVM_ARCH

Going to the pvm3 directory and again executed make command. If a prompt pvm> is got means pvm is successfully installed.

V. IMPLEMENTATION

Implementation is the most crucial stage in achieving a successful parallel system. The problem to be solved has to be parallelized so that computation time is reduced. The framework consists of a client, a master process, capable of handling requests from the client, and slave, capable of accepting problems from the master and sending the solution back. The master and the slave communicate with each other using PVM3.4.6 under PVM. The problem has to be divided such that the communication between the master and the slave is minimum. The total computational time to solve the problem completely is effected by the communication time between the nodes.

A. Parallel Matrix Multiplication Design

In the algorithm which we have implemented is for solving matrix multiplication problem on several nodes it may be for only one or more slaves. It divides the matrix into set of rows and sends it to the slaves rather than sending one row at a time [6].The slaves compute the entire set of rows that they have received and send it back to the server in one send operation. Hence, we need to implement parallel systems consisting of set of independent desktop PCs interconnected by fast LAN cooperatively working together as a single integrated computing resource so as to provide higher availability,

reliability and scalability. But to show the performance dependency on RAM we are considering single node analysis and multiple node analysis both. Consider two matrix, matrix A and B. The operations involved in getting the resultant matrix are shown in Fig. 3 with simple example.

VI. TESTING

There is a need of testing whether the resultant matrix is correct or not. Hence testing is one of the important steps that must be performed, which is explained in the following section. Separate testing is made for smaller order matrices and larger order matrices.

A. Testing of Small Order Matrices

We have tested the output of our system by verifying the output using a online matrix calculator. We have repeated this test for different smaller sizes of matrix. The tool used by us is Blue Bit online matrix multiplication calculator.

B. Testing of higher order matrices

Blue Bit online matrix multiplication calculator is limited to matrix of order 32 as the testing tool does not support the matrix multiplication of higher order. To verify our result for higher order matrix we have assigned 1 to all the elements of the matrix. The elements of the output matrix will be equal to the order of the matrix. One of experimental results is shown in Fig. 4.

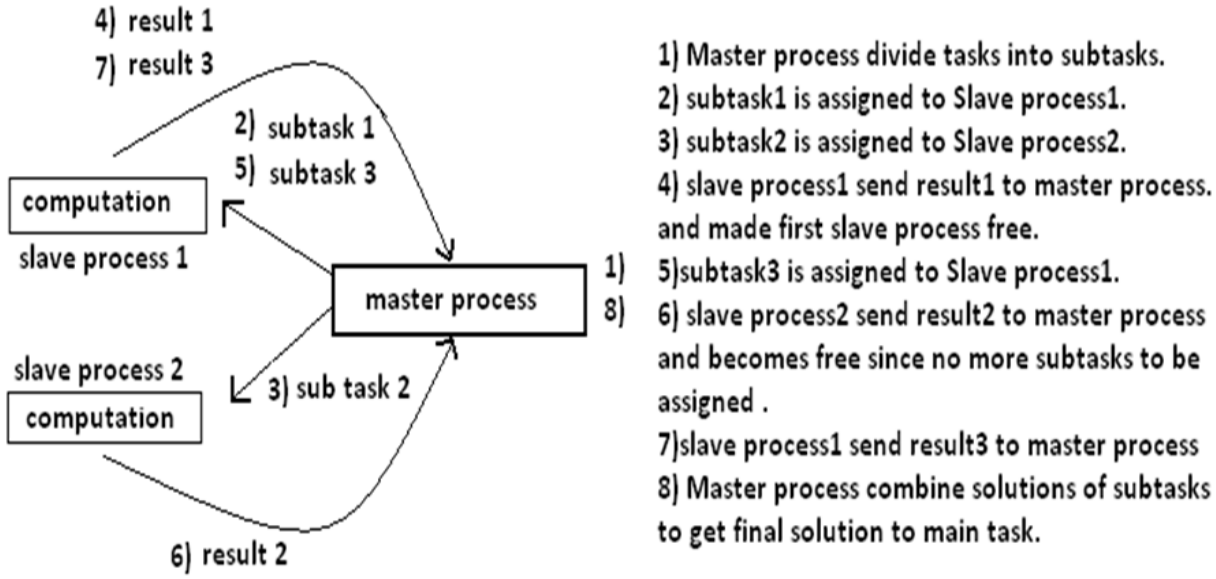

Fig 3: Example for cyclic computation (1 task into 3 subtasks computed using 2 slaves)

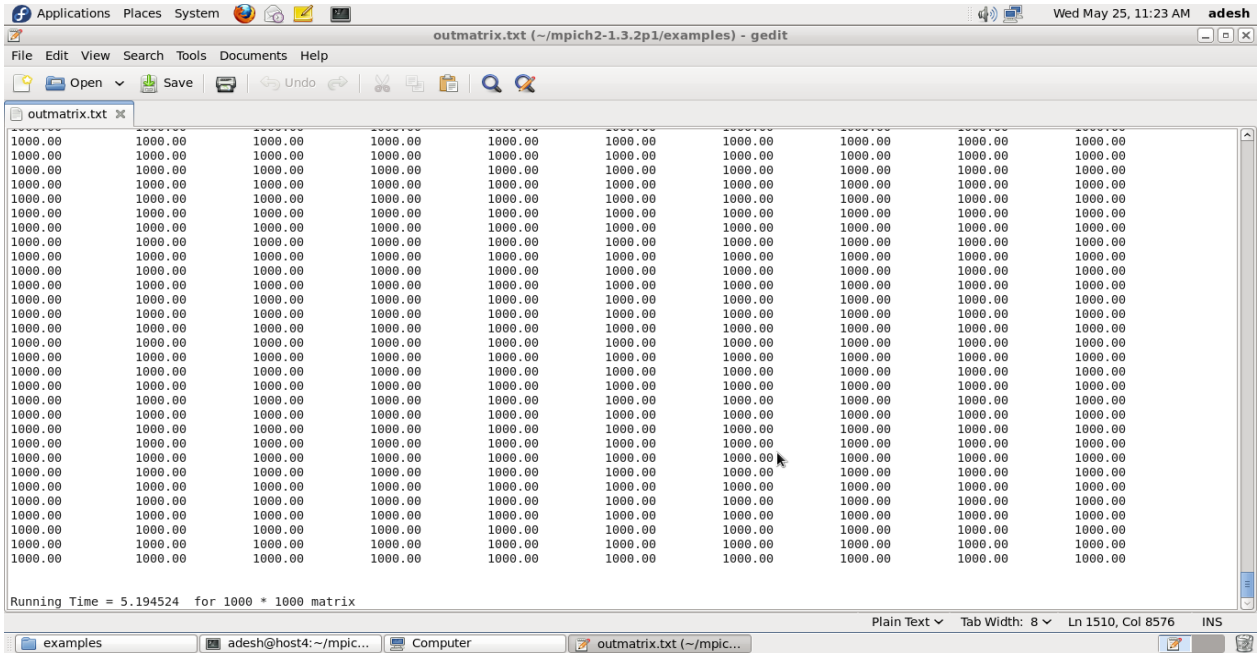

Fig 4. Output of 1000 * 1000 matrix in parallel execution

VII. RESULTS AND ANALYSIS

We have analyzed the performance of parallel method against traditional serial method. The results are tabulated and compared. We calculated the time for solving the matrix multiplication problem using both serial and algorithm using PVM. From the Table I, we can make the following analysis.

TABLE I
RESULTS OBTAINED FOR SERIAL AND PARALLEL EXECUTION AT 1000MB AND 2000MB RAM

TYPE OF MATRIX	SIZE OF MATRIX ↓	SERIAL		PARALLEL ONE NODE		PARALLEL TWO NODES		PARALLEL THREE NODES	
	RAM SIZE ►	1OOOMB	2000MB	1OOOMB	2000MB	1OOOMB	2000MB	1OOOMB	2000MB
SMALLER ORDER MATRICES	100*100	0.009879 seconds	0.009868 seconds	0.010389 seconds	0.008761 seconds	0.016303 seconds	0.005357 seconds	0.021133 seconds	0.0073775 seconds
	200*200	0.056072 seconds	0.056135 seconds	0.067870 seconds	0.027632 seconds	0.059804 seconds	0.021436 seconds	0.069564 seconds	0.024377 seconds
	300*300	0.133191 seconds	0.131281 seconds	0.188082 seconds	0.121471 seconds	0.150387 seconds	0.098763 seconds	0.192606 seconds	0.100231 seconds
	400*400	0.456641 seconds	0.457432 seconds	0.592474 seconds	0.217563 seconds	0.512551 seconds	0.189231 seconds	0.611410 seconds	0.220101 seconds
	500*500	0.617780 seconds	0.615478 seconds	0.799223 seconds	0.352782 seconds	0.709623 seconds	0.308741 seconds	2.228170 seconds	0.331722 seconds
HIGHER ORDER MATIRICES	1000*1000	4.880135 seconds	4.885176 seconds	5.445081 seconds	1.274521 seconds	2.870787 seconds	1.088341 seconds	1.831946 seconds	0.899451 seconds
	2000*2000	39.076844 seconds	39.122671 seconds	42.779060 seconds	21.473342 seconds	22.047581 seconds	18.713992 seconds	13.607614 seconds	13.467201 seconds
	3000*3000	131.986616 seconds	131.991312 seconds	157.531261 seconds	58.448233 seconds	80.499557 seconds	42.721765 seconds	47.275596 seconds	33.891427 seconds
	4000*4000	311.428811 seconds	311.406789 seconds	445.617023 seconds	198.289276 seconds	252.446168 seconds	172.321761 seconds	125.896913 seconds	123.091176 seconds
	5000*5000	607.981106 seconds	607.979623 seconds	955.095157 seconds	524.162254 seconds	493.288030 seconds	475.207114 seconds	312.011965 seconds	298.562781 seconds

Case 1: Performance dependency on RAM. Table I shows that performance of serial execution remains same even after the increase in RAM size. There are negligible computation time variations for increase in RAM size. This is because the Serial execution is performed by the processes itself with negligible RAM usage and also due to the no communication involved between processes. Hence it is independent of RAM. We can also conclude that performance of parallel execution increases when there is increase in RAM size. It shows drastic decrease in computation time with the increase in RAM. Because parallel execution often uses RAM for the communication between processes and also it involves lot of send and receive operations and temporarily storing the result of problem assigned to processes. We can analyze that higher the size of matrices the time difference is very high in the table, because higher the matrix size, more will be sends and receives resulting in the need of higher utilization of RAM. So for smaller RAM the computation time will be more and larger the RAM size computation time will be less in parallel execution finally resulting in better performance. However by seeing the time results for higher order matrices such as 1000*1000, 2000*2000, there is a large reduction in computation time for PVM when there is increase in RAM size. Hence we can conclude that PVM based parallel applications is more dependent on RAM size

Case 2: Single node analysis to show communication overhead involved in parallel computations. Generally we can say parallel execution is faster than the serial execution but the results of serial execution with 1000MB RAM and parallel execution using PVM with 1000MB RAM shown in the Table I depicts that serial execution is faster than parallel execution in a single node having two processes, for different sizes of matrices. This is due to the communication overhead involved in the parallel execution but this can be overcome by increasing the number of nodes as shown in the TABLE I. Overheads that are considered are the connection time required to connect to slave, time taken to send the problem along with inputs to slave time taken to retrieve the solutions from the client, time taken to assimilate the results obtained.

Case 3: Multiple nodes analysis with smaller order matrices (computation time < communication time). Table I shows the time taken to solve the problem wholly is more when the number of nodes is more for smaller matrix. Because the problem has to be communicated among all the slave processes hence the communication time is larger than the computation time. So the 2 nodes can compute it and assemble it faster than a 3 node or a 4 node system.

Case 4: Multiple nodes analysis with higher order matrices (computation time > communication time). Table I shows that for matrix of higher order the performance of the system increases phenomenally with increase in number of nodes. As the size of the matrix increases the computation time also increases. The computation time is so large that the communication time is negligible compared to it.

VIII. Conclusion

We presented a model that demonstrates the performance gain and losses achieved through parallel processing. Matrix multiplication problem is solved serially and also in parallel under PVM. We demonstrated the evaluation of the performance dependency of PVM based parallel applications and serial execution on RAM under different sizes of RAM. Serial execution is faster for smaller matrix because of the communication and connection overheads in parallel execution. The performance of parallel execution is far greater compared to serial execution when the size of the matrix is large. The total time taken to compute the result decreases drastically when the number of nodes increases.

IX. Future Works

Even though the method that has been used here can be deployed to solve larger order problems, it is cumbersome to give the data input for matrices of larger order. Hence this work can be extended to give input from files for larger order matrices. It can also be extended to solve other similar problems related to matrices, like finding the determinant and other backtracking problems. The analysis is also useful for making a proper recommendation to select the best algorithm related to a particular parallel application. If the nodes are extended, node failure can be a problem that has to be tackled.

Acknowledgement

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. Suraj M.G, Dr. Mallikarjuna Bennur, for their valuable suggestions given for us throughout our work.

References

[1] A. Nazir, H. Liu, and S.-A. Sørensen, "On-demand resource allocation policies for computational steering support in grids, " in International Conference on High Performance Computing, Network and Communication Systems, Orlando, USA, 2007.
[2] History of PVM versions: http://www.netlib.org/pvm3/book/node156.html.
[3] PVM3.4.6:http://www.csm.ornl.gov/pvm/pvm3.4.6
[4] V.S Sunderam, G.A Geist, J Dongarra, R Manchek, "PVM Concurrent Computing System Evolution, Experiences and Trends", Parallel Computing - Special issue: message passing interfaces archive Volume 20 Issue 4 Pages 531-545, April 1994.

[5] Amit Chhabra, 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.

[6] Muhammad Ali Ismail, Dr. S. H. Mirza, Dr. Talat Altaf, "Concurrent Matrix Multiplication on Multi Process Processors", International Journal of Computer Science and Security (IJCSS), Volume (5) : Issue (2) : 2011.

[7] Kamalrulnlzam Abu Bakar, Zaitul Mlir lizawati Zal nuddln, "Parallel Virtual Machine versus Remote Procedure Calls: A Performance Comparison", Jilid IS. Bit. I, Jumal Teknologi Maklumat, Jun 2006.

AUTHORS PROFILE

Gopalakrishna C. is an Associate professor in the Dept of computer Science & Engineering, Adi chunchanagiri Institute of Technology, chikmagalure. He holds Masters of Engineering in Computer science & Engineering (UVCE, Bangalore, 2003) Having Teaching Experience of 14 years. His current research interests include Networking, Web Services, cloud computing, DBMS. He has reported his work in Journals, conference papers and Technical reports and so far has supervised more than 30 projects at the Graduate and Post-Graduate level.

Gagana B.G, Fathima Ruqsar T.S and Deepesh R respectively are doing B.E, Computer Science and Engineering Branch in Adichunchanagiri Institute of Technology, Chikmagalur under Vishveshvaraia Technological University. They are working on analysis of different parallel applications using PVM as their UG final year project.

Nanjesh B.R is doing M.Tech in Computer Science and Engineering branch, Adichunchanagiri Institute of Technology, Chikmagalur, Karnataka, India. He obtained his B.E from A.I.T. college Chikmagalur under Vishveshvaraia Technological University. His fields of interest are image processing and parallel computing. He has undertaken the comparison of MPI and PVM as M.Tech project

