Volume 7, Issue 5, May 2017 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Development of Integrated Parallel Computing Platform for Parallel Computing Environment

Surender Jangra
Dept. of Computer Science and Applications, Guru Tegh Bahadur College, Bhawanigarh, Sangrur, Punjab, India

Shant Kaushik
Dept. of Computer Science and Applications, D.A.V. College (Lahore), Ambala City, Haryana, India

Mehzabeen Kaur
M.Tech. (CSE) Student Department of Computer Science and Engineering, BBSBEC, Fatehgarh Sahib, Punjab, India

DOI: 10.23956/ijarcsse/V7I5/0167

Abstract*: *Parallel Virtual Machine and Message Passing Interface are the most successful Message passing libraries to map parallel algorithm onto parallel computing platform. Configuration of MPI and PVM in the nodes present in parallel computing environment (desktop PC's interconnected using Ethernet LAN) is the time consuming task for a user. This configuration procedure requires lot of knowledge about the steps to be followed to make them work properly. Configuration becomes the difficult task when there is more number of nodes in the parallel computing environment. Our work aims on developing a common parallel programming platform which allows a user to get MPI and PVM in the nodes without any time consuming configuration steps. This is done by integrating the recent version of PVM: PVM3.4.6 into the MPI's MPICH2 packages in order to simplify the time consuming task of configuring them separately.

Keywords: MPI; MPICH2; PVM; Parallel; GUI, Linux; PVM3.4.6

I. INTRODUCTION

Many image processing applications involves image multiplication where each image represented in the form of matrix, the currently the Matlab is in use whose computation by single computer takes more computation time, this can be parallelized using cluster based parallel computing using PVM which is faster than that of Matlab applications. If parallel merge sorting used instead of parallel matrix multiplication, it can be used It can be used as alternative for parallel cursing in IT sector and it can be used in sorting of voters list, Adhar card holders based on their ID's.

The configuration procedure for MPI and PVM requires lot of knowledge about the steps to be followed to make them work properly. Configuration becomes the difficult task when there is more number of nodes in the parallel computing environment. This motivated to develop a common parallel programming platform which allows a user to get MPI and PVM in the nodes without any time consuming, complex configuration steps.

II. LITERATURE SURVEY

A recent version of PVM such as pvm3.4.6 is updated with more and more functionalities to give competition in performance, with MPI. This motivated us to evaluate and compare the performance of recent versions of MPI and PVM to demonstrate when PVM is faster than MPI and vice versa. The recent development is MPICH2 and it is freely downloadable source code via MPI organization [3]. MPI-1 and MPI-2 both are supported by latest version of MPI and which opened the door of further research in the field of high performance and scalability. PVM3.4.6 is freely downloadable source code [4]. PVM3.4.6 [5] is the recent version of PVM which is work on GUI (Window) and CUI (Linux). It is very much compatible with all the advanced version of Linux.

This work introduces the contribution of Open standards such as Open Multi Processing (OpenMP), Message Passing Interface (MPI), Open Computing Language (OpenCL), Open Computer Visio (OpenCV) and OpenACC in parallel computing. This survey is accomplished with study of different programming languages according to parallel models. Nisha Dhankher et al. 2014 [6] investigated the performance of parallel implementation of BLAST algorithm on HPC platform using Infiniband. They described the optimized and extended version of mpiBLAST called mpiBLAST-PIO. Due to high nonsearch overhead, parallel-writing the results by the slaves evolved as the efficient solution to the problem. Rajesh Kumar et al. [7]- [10] proposed automated fault tolerance systems for controlling computation power in desktop grid through GUI based environment using Alchemi desktop grid middleware. Sampat S. et al. [11] compared the MPI and PVM on different perspectives. S. Jangra et al. proposed a fault tolereance architecture for mobile distributed systems [12]-[13]

III. INTEGRATED PARALLEL COMPUTING PLATFORM

For the development of Integrated Parallel Computing Platform following configuration steps are used as shown in Figure 1. Initially configuration of MPI and PVM is done over the LINUX platform separately.

(a) Configuration of MPICH2 [1]

(b) Configuration of PVM3.4.6 [2]

(c) Integration of PVM3.4.6 into MPICH2 packages

(i) After configuration of MPI and PVM we need to do the following procedure: Copy the architecture named directory present in pvm3.4.6 directory into mpich2 1.3.2p1.

(ii) Merge the files present in conf, lib directory of pvm3.4.6 to conf, lib directory respectively of mpich2-1.3.2p1

(iii) MPIC++, mpirun and mpiexec are added to mpich2-install file /bin.

(iv) Copy the libfpvm, include, xdr, pvmgs, tracer, shmd, src directory to (new features of pvm3.4.6) to mpich2-1.3.2p1.

(v) Merge the makefile and makefile.aimk, makefile.in.

(vi) In mpich2-install directory add (few are copied or few are merged) the lib, bin, include files of pvm3.4.6 to the respectively named directories.

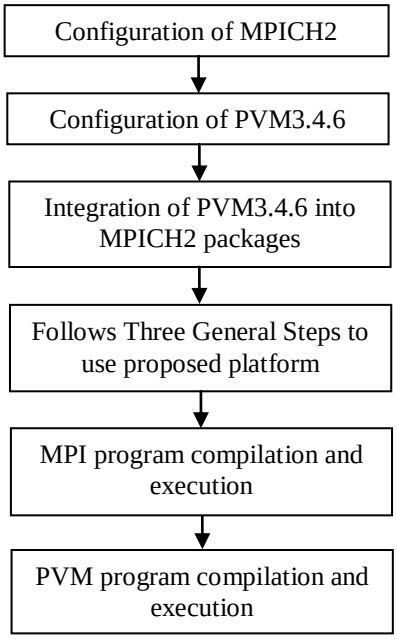

Figure 1: Flowchart for proposed integrated Parallel Computing Platform

The above steps give us the directories ***mpich2-1.3.2p1*** and ***mpich2-1nstall*** containing the packages of both MPICH2 and PVM3.4.6. These two are now portable and can be used in any node where PVM and MPI programming environment need to be established.

(d) Steps for using of Proposed Platform

A user must do the following steps to make the new programming model to work properly:

(i) Paste the two integrated files ***mpich2-1.3.2p1*** and ***mpich2-install*** in the node. We use two supporting files along with the integrated directories namely supporting_file1 and supporting_file2 shown in Figure 2 and Figure 3 respectively.

(ii) ***vi $HOME/.bashrc:*** Remove all contents of this file and Paste the contents of supporting file1 to it and save. Instead of specifying PVM_RSH, PVM_ROOT, PVM_ARCH, user should specify only his/her user name of the LINUX platform in the first line of the contents shown in supporting file1.

(iii) ***vi $HOME/.bash_profile:*** Remove all the contents of this file and Paste the contents of supporting file2 to it and save. User must specify the his/her user name of the LINUX platform and architecture of the LINUX based node in the fourth and sixth line of the contents shown in supporting file2.

```
PVM_RSH=/usr/bin/ssh
Export PVM_RSH
PVM_ROOT=/home/username/mpich2-1.3.2pl
export PVM_ROOT
PVM_ARCH=architecuter _name
#(example:LINUX64)
export PVM_ARCH
if[-f ~/.bashrc ]; then
. ~/.bashrc
fi
PATH=$PATH: $HOME/bin
export PATH
```

Figure 2 Contents of supporting_file1

(e) Program Compilation and Executions

```
PVM_ROOT=$HOME/pvm3
export PVM_ROOT
PVM_RSH=/usr/bin/ssh
export PVM_RSH
if [ -f /etc/bashrc ]; then. /etc/bashrc
fi
if [ -z $PVM_ROOT ]; then
if [ -d ~/pvm3 ]; then
export PVM_ROOT=~/pvm3
else echo "Warning -PVM_ROOT not defined"
fi
fi
if [ -n $PVM_ROOT ]; thena
    export PVM_ARCH=`$PVM_ROOT/lib/pvmgetarch`
    export PATH=$PATH:$PVM_ROOT/lib/$PVM_ARCH
    export PATH=$PATH:$PVM_ROOT/bin/$PVM_ARCH
fi
```

Figure 3: Contents of supporting_file2

Program compilation and execution of proposed developed programming model is same as we do in the existing programming environment. User can compile and execute parallel code using MPI or PVM depending on his/her need.

It is shown in Figure 4 and Figure 5 for MPI and PVM respectively. This program is made to run over single and multiple nodes for the different sizes of the matrices and checked for correctness. For the execution over the multiple nodes, all the nodes are configured with ssh for password less login from one node to the other during the computation. Sample results of MPI and PVM is shown in Appendix-snapshots. Multiple nodes testing is limited to three nodes.

Paste program files into the directory mpich21.3.2p1→examples, Type the following:

1) cd /home/username/mpich2/1.3.2p1/examples
2) PATH=/home/username/mpich2-install/bin:$PATH; export PATH
3) mpicc –c programname.c –o programname.o mpiexec-f machine –n x /home/username/mpich2-1.3.2p1/examples/programname
4) mpicc programname.o –o programname mpiexec –f machine –n x /home/username/mpich2-1.3.2p1/examples/programname

/* in the above lines x is the number of slaves needed for computation and machine is machine file containing list of hosts (in the form host1:2 host2:2 and so on), this file is created in cd /home/username/mpich2-1.3.2p1/examples).

Figure 4: MPI Program Compilation and Execution

Paste master and slave program files into the directory mpich2-1.3.2p1 →examples
Type the following:

1) cd /home/username/mpich2-1.3.2p1/examples
2) gcc –g –ggdb –I/home/username/mpich2-install/include – L/home/username/mpich2-3.2p1/lib /home/username/mpich21.3.2.p1/examples/masterprogramname.c –o /home/mpich2-1.3.2p1/examples/masterprogramname /home/username/mpich21.3.2p1/lib/architecturename/libpvm3.a /home/username/mpich21.3.2p1/lib/architecturename/libgpvm3.a
3) gcc –g –ggdb –I/home/username/mpich2-install/include – L/home/username/mpich2-3.2p1/lib /home/username/mpich21.3.2.p1/examples/slaveprogramname.c –o /home/mpich2-1.3.2p1/examples/slaveprogramname /home/username/mpich21.3.2p1/lib/architecturename/libpvm3.a /home/username/mpich21.3.2p1/lib/architecturename/libgpvm3.a
4) cp masterprog $PVM_ROOT/bin/architecturename
5) cp slaveprog $PVM_ROOT/bin/architecturename

/* Proposed platform is called with name is LINUX64 here */

Figure 5: PVM program compilation and execution

IV. RESULT AND ANALYSIS

Table 1 shows an approximate comparison of time required for the configuration of MPI and PVM separately and time required for installing MPI and PVM using integrated programming model.

Table 1: Time comparison between Separately configuration Vs integrated configuration

Time Required minutes per node	MPI	PVM	Total time in minutes per node (Approx.)
Individual Configuration (Old Method)	20	10	30
Proposed Integrated Platform	-	-	05

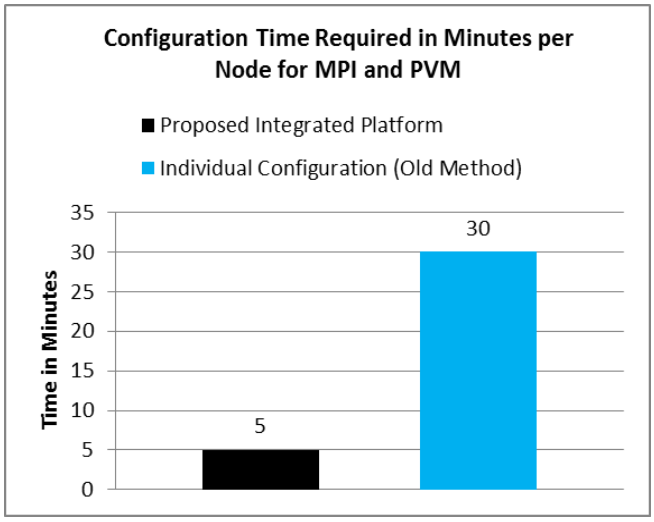

Figure 6: Configuration Time Comparisons

The common platform designed for MPI and PVM require comparatively very less configuration time. This will be very useful where the parallel computing environment involving more number of nodes has to be established.

V. CONCLUSION

Thus we discussed on the development of common parallel programming platform for both MPI and PVM using recent versions of MPI and PVM: MPICH2 and PVM3.4.6 respectively. The parallel programming model developed by integrating PVM into MPI provides a common platform for both MPI and PVM programming where separate configuration is not required for both. Proposed model is highly portable as software (zipped into one file containing two supporting files and two integrated directories) and takes much less time needed for getting MPI and PVM installed in the nodes.

Show the integrated environment by combining PVM and SVM. This process reduces the configuration time from thirty minutes to five minutes. The first phase of the research focuses on developing integrated computing platform for parallel applications using latest version of PVM & MPI (PVM3.4.6 into MPICH2). In second phase,

REFERENCES

[1] Alaa Ismail El-Nashar, Hussain Abo Surrah (2014), 'Risks Of Using Message Passing Interface (MPI) Paradigm', *International Journal of Advanced Computer Research*, Volume-1, 2014, pp 14-22.

[2] Alfredo Buttari, Julien Langou, Jakub Kurzak, Jack Dongarra (2009), 'A class of parallel tiled linear algebra algorithms for multicore architectures', *International Journal of Parallel Computing*, Volume 35, Issue 1, January 2009, Pp 38-53.

[3] website:http://www.mpich.org.

[4] http://www.netlib.org/pvm3/pvm3.4.6.tgz.

[5] Alfredo Buttari, Julien Langou, Jakub Kurzak, Jack Dongarra (2009), 'A class of parallel tiled linear algebra algorithms for multicore architectures', International Journal of Parallel Computing, Volume 35, Issue 1, January 2009, Pp 38-53.

[6] Nisha Dhankher , O P Gupta ,"Parallel Implementation & Performance Evaluation of Blast Algorithm on Linux Cluster,2014.

[7] R. Kumar and Surender, "Automated Fault Tolerant System for Control Computational Power in Desktop Grid," IEEE Xlpore, Pg. 818-821 (2015).

[8] R. Kumar, Surender Jangra, "Automatic Fault Tolerance Software System for Desktop Grid Middleware," Procedia Computer Science (Elsevier) 85(2016), pp. 987-994, ISSN: 1877-0509.

[9] R. Kumar and S. Jangra, "Automated Fault Tolerance Framework for Alchemi Desktop Grid," International Journal of Advances in Engineering Sciences, vol. 5, issue 4, Oct. 2015, Pg. 11-20.

[10] R. Kumar and S. Jangra, "Memory and CPU Power based Automatic Fault Tolerance Framework in Alchemi Computational Grid," International Journal of Computer Applications(IJCA), vol. 134, No. 6, Jan 2016, Pg. 13-18.

[11] Sampath S, Bharat B. S, Nanjesh B.R, "Performance evaluation and Comparison of Message passing Interface and Parallel Virtual Machine, " International Journal of Advanced Research in Computer Science and Software Engineering, 2013, Volume 3, Isssue 1, pp. 200-206, 2013.

[12] S.Kumar, R.K. Chauhan, P. Kumar, "Minimum Process Error Recovery Algorithms for Mobile Distributed Systems using Global Checkpoint," International Journal of IT and Knowledge Mgmt." vol. 1, No. 1, Pg. 25-33, Jan-June 2008. ISSN- 0973-4414.

[13] A. Sejwal, S. Jangra, Y. Singh Sangwan, "A Decision Support System for Industry based upon Multivariate Spatial Outlier Detection Technique," Published by Science Direct (ELSEVIER), "Procedia Technology", 4(2012), Pg. 401-405.

[14] Naveen Garg, Sanjay Singla and Surender Jangra, "Challenges and Techniques for Testing of Big" published in the Journal, "Procedia Computer Science (Elsevier)", 85 (2016), Pg. 940-948, DOI: 10.1016/j.procs.2016.05.285, ISSN: 1877-0509

[15] Low Overhead Time Coordinated Checkpointing Algorithm for Mobile Distributed Systems" published in book chapter " Computer Networks and Communications (NetCom) Volume 131 of the series Lecture Notes in Electrical Engineering," Edited by Nabendu Chaki, Natarajan Meghanathan Dhinaharan Nagamalai, Total Pg. 865

[16] Surender Kumar, R.K.Chauhan and Parveen Kumar, "A Low Overhead Minimum Process Global Snapshot Collection Algorithm for Mobile Distributed Systems", Published in "International Journal of Multimedia and Its Applications [IJMA] (AIRCC, France)", Vol.2, No. 2, Pg. 12-30, May 2010.

