ISSN 1392 – 1215 ELECTRONICS AND ELECTRICAL ENGINEERING
ELEKTRONIKA IR ELEKTROTECHNIKA
2009. No. 5(93)

T 120 SYSTEM ENGINEERING, COMPUTER TECHNOLOGY
SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

Performance Evaluation of an Experimental Grid Computer using MPI Applications

I. Ungurean, S. G. Pentiuc, V. Gaitan
Department of Computers, Faculty of Electrical Engineering and Computer Science,
Stefan cel Mare University of Suceava, Romania, phone: +40-230-520277; e-mails: ioanu32@yahoo.com, pentiuc@eed.usv.ro, gaitan@eed.usv.ro

Introduction

In this paper we will present a discussion about parallel programs performance [1][2]. These programs were developed using the OpenMPI[3][4] implementation of the Message Passing Interface (MPI)[5] standard and were run on an experimental grid computer made of 7 desktop computers. These test applications measure execution times for applications implementing the Jacobi[6] approximation for a linear system of equations. The output of the test application is the processing speed, measured in MFlops and obtained using more methods in order to test more aspects of inter-process communication. The final purpose of this paper is to see increasing performance in a Grid [7] by increasing the number of computing nodes.

A GRID is a hardware and software infrastructure that provides consistent access to computing capabilities. In fact the main idea of the GRID is developing a system in which access to computing resources is supposed to be as easy as accessing electrical power resources. One of the remarkable things in the electrical infrastructure is that it does not require knowing the electricity generator location and details of network infrastructure systems. Unfortunately this goal is not yet fully achieved. A GRID architecture is structured on several levels, each level having a specific function. In general the highest level targets the user. In this work tests were performed for performance testing of parallel programs on an experimental grid. The experimental grid is made of computers connected using Ethernet networks of 100Mbit and 1Gbit.

The experimental grid

The experimental grid is composed of 7 computers, one representing the server and other 6 nodes the computing nodes. On each computer Scientific Linux 5.0 is installed as the operating system, Globus Toolkit [8] and OpenMPI packages (representing an open-source implementation of the MPI standard).

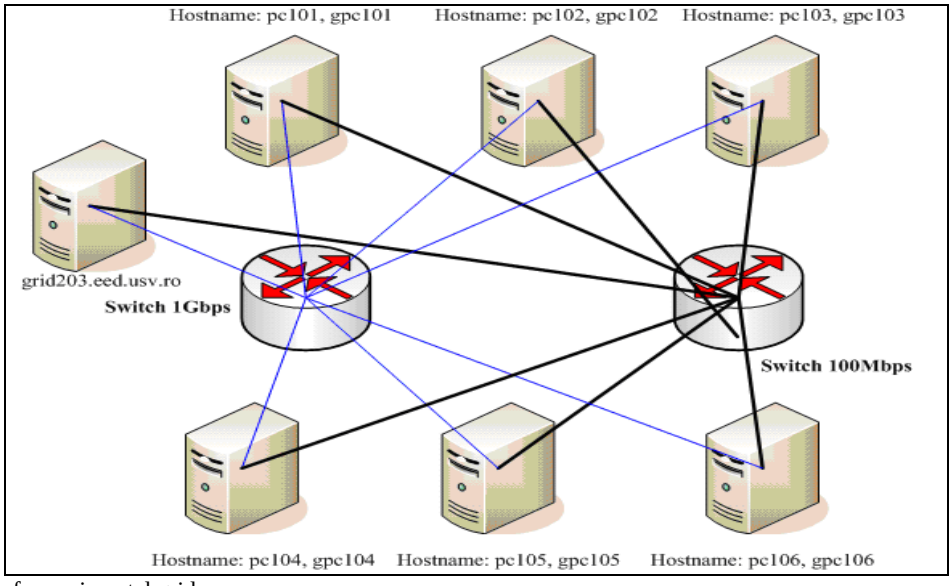

Fig. 1. Architecture of experimental grid

The architecture of the experimental GRID is presented in Fig. 1. The GRID contains two swhiches (one of 100Mbps and one of 1Gpbs). These two swhiches are connected to all nodes of the GRID. In this way two networks are formatted, one can be used for administration and one can be used for implementation and execution of MPI programs. Each node has associated two hosts (eg pc101 and gpc101) which correspond to each network. Nodes are using private classes of IP addresses and the server has the "grid203.eed.usv.ro" hostname and public IP Address.

Hardware configuration of computers used for the experimental GRID is:

- Processor: AMD Athlon (tm) XP 2000 +, 1.66 Ghz;
- RAM: 64MB.

Determination of processing speed

The Jacobi method is an algorithm in linear algebra for determining the solutions of a system of linear equations with largest absolute values in each row and column dominated by the diagonal element.

To determine the processing speeds of the experimental GRID the Jacobi iterations [6] for approximation of the solution for a system of linear equations were implemented. The applications will solve Laplace equation in two dimensions with finite difference.

Nxn system of equations can be represented as a grid (as seen in

Fig. 2). Any numerical analysis will show that the iteration in which a point from the grid is replaced with average of neighbors will achieve an approximation for the solution of the Laplace equation. There is one last detail: the replacement of the grid with the average values of around only applies to the inside, the values for boundary point is unchanged. Since the values are replaced with the average values from around, this method is called relaxed. You can also define a convergence condition, but for the determination of processing speed the application will execute a fixed number of iteration.

For simplicity we consider an example with a grid of 8x8 on 3 processors.

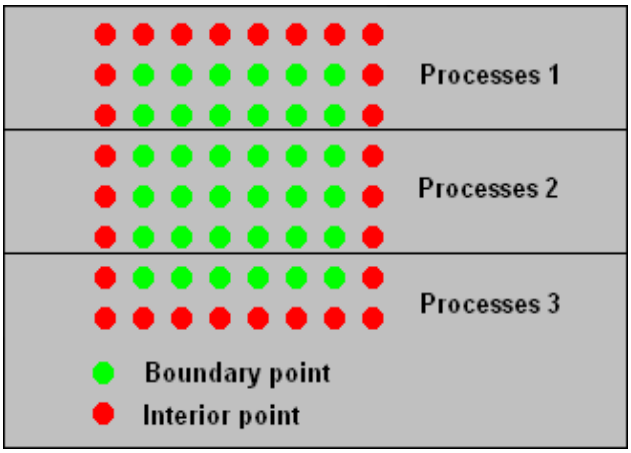

Fig. 2. Grid example for a 8x8 system of equations on 3 processors

This method is very weak but is used because of its simplicity. We develop a few applications that implement this method using different techniques for the communication between processes (which are technically defined in the MPI standard).

The MPI_Wtime function is used in order to measure execution time in an MPI application. This function returns a double value, which represents the time passed from an arbitrary point in the past. If the value for MPI_WTIME_IS_GLOBAL attribute is set to true then the value returned by this function is synchronized to all processes from MPI_COMM_WORLD.

Jacobi method using MPI_Send / MPI_Recv operations

The Jacobi method for a problem size of 8x8 was implemented in this example. For communication were used MPI_Send and MPI_Recv functions. Also the solution is given for exactly 250 iterations.

The results are given (in MFlops) for execution of 2.3 ....6 processes on the same node and 6 different nodes. It can be noted that if the number of nodes increases the performance does not increase significantly. This can be explained by the fact that many communication operations are performed, operations which run much slower than the computing operations performed by each task.

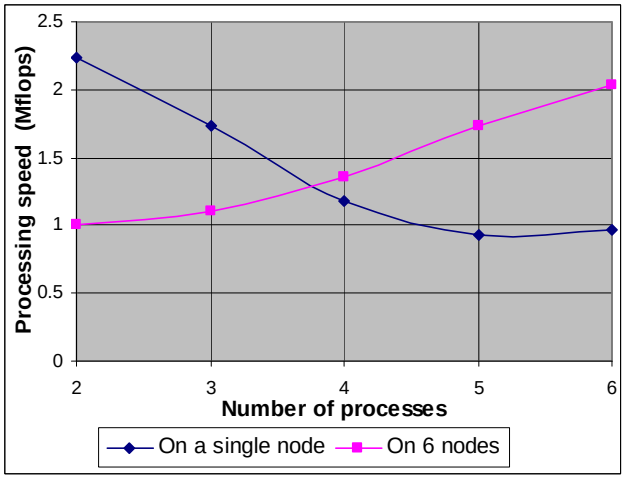

Fig. 3. Results for implementing Jacobi method using MPI_Send / MPI_Recv operations

Jacobi method using MPI_Sendrecv operations

In this section are presented results of an implementation of the Jacobi method which use the MPI_Sendrecv function. This function carries out a data exchange between two processes. A process calling this function remains blocked until the message is transmitted and another message is received. The results are given for application execution on the same node or different 6 nodes.

It can be seen that the performance differences are not very high between this case and the case where used MPI_Send and MPI_Recv functions for communication between processes.

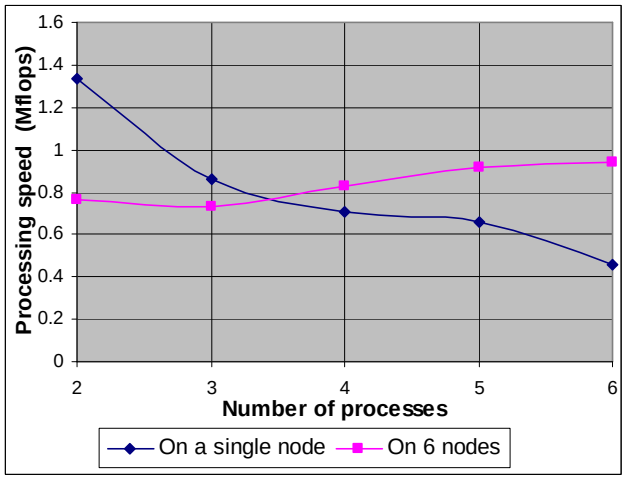

Fig. 4. Results for implementing Jacobi method using MPI_Sendrecv operations

Jacobi method using non-blocking operation

This version uses the non-blocking communication operations for sending and receiving messages. In this version, the transmission is announced first allowing the use of rendezvous protocols for receiving data the by the recipient process and decreasing the synchronization cost (this can not be guaranteed). The results are given for application execution on the same node or different 6 nodes.

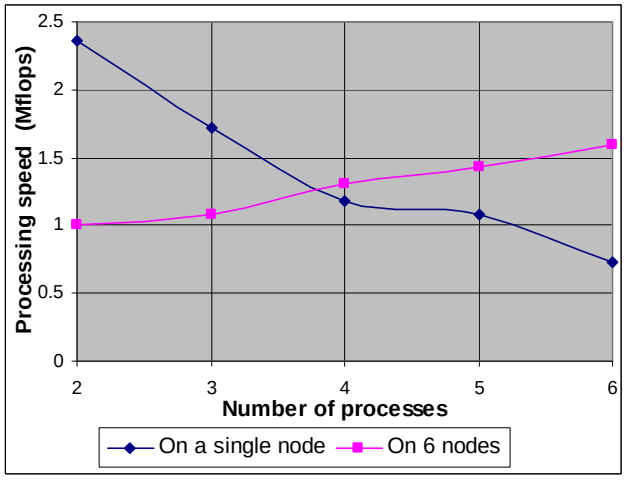

Fig. 5. Results for implementing Jacobi method using non-blocking operation

Jacobi method using MPI_Rsend function

In a simple Jacobi loop at the end of each iteration the MPI_Allreduce function can be used to test the solution convergence. This provides a synchronization point in the program. The current example uses this synchronization to allow the use of routine MPI_Rsend. In this case, the reception (MPI_Isend) is posted before the point of synchronization and transmission (MPI_Rsend) is posted after the synchronization. The results are given for application execution on the same node or different 6 nodes.

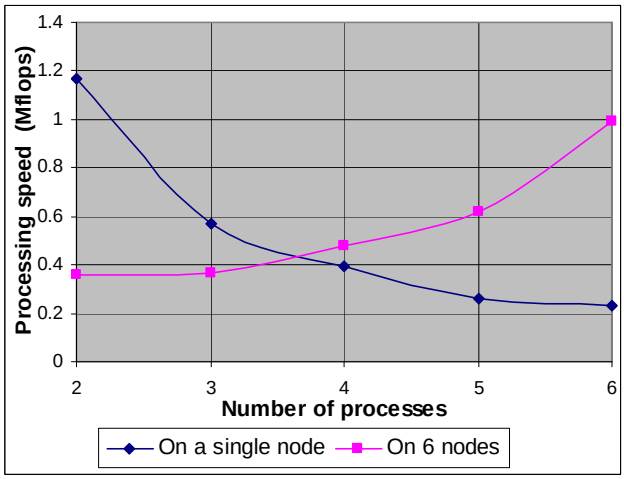

Fig. 6. Results for implementing Jacobi method using MPI_Rsend function

Jacobi method using MPI_Irsend function

This example is similar with the previously one, the only difference being that instead of MPI_Rsend function it is using MPI_Irsend function (this function implementing a non-blocking operation for data transmission). The results are given for application execution on the same node or different 6 nodes.

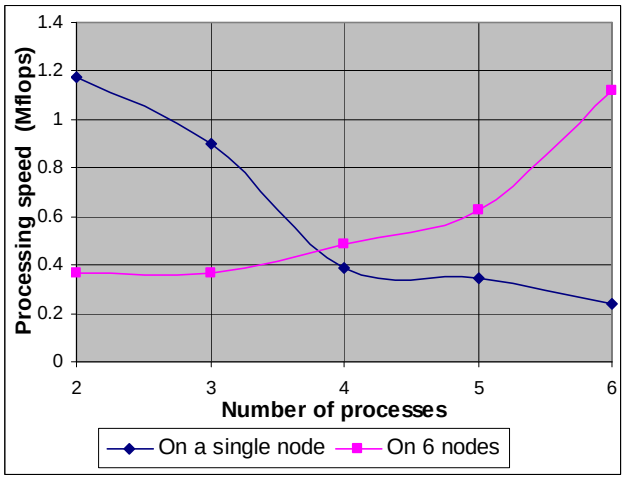

Fig. 7. Results for implementing Jacobi method using MPI_Irsend function

Jacobi method using overlap of calculations and communications

In this example the calculations are divided in two parts: one part that needs data from other processors and the other part that does not need. The part that is independent of other processes is the domain (relative to each process) and the part which needs external data is the border. Communication is switched on, is done for data communication from the inside, ending the communication and made calculations for the border. This will produce an overlap of calculations and communications. The results are given for application execution on the same node or different 6 nodes.

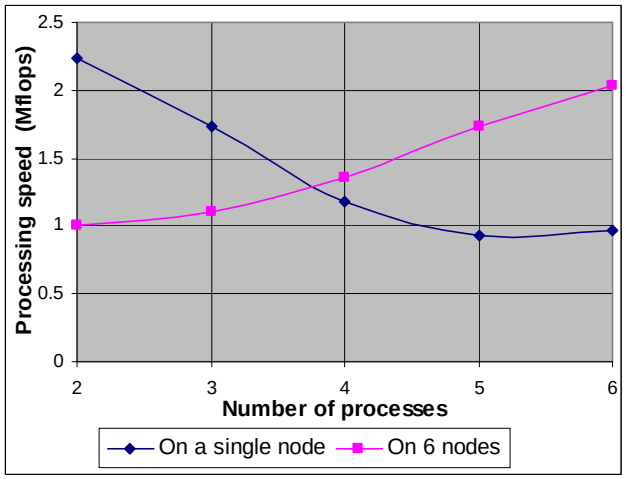

Fig. 8. Results for implementing Jacobi method using overlap of calculations and communications

Conclusions

Following the mentioned tests, some differences between parallel programs performances can be noted given the different communication techniques. Communication between parallel applications is a very important issue, as shown in the tests that were made by the effect it has on global performance. The tests are a real proof of Amdalh's Law [9]: performance does not rise two times if the number of nodes is doubled.

References

1. **Foster Ian**. Designing and Building Parallel Programs. – Addison-Wesley. – 1995.
2. **Clément F., Martin V., Vodicka A., Di Cosmo R., and Weis P.** Domain decomposition and skeleton programming with OCamlP31 // Parallel Comput.– Sep. 2006. – Vol. 32, No. 7. – P. 539–550.
3. Open MPI: Open Source High Performance Computing [interactive]. Accessed at: www.open-mpi.org.
4. **Rainer Keller, Shiqing Fan, and Michael Resch.** Memory Debugging of MPI-Parallel Applications in Open MPI // ParCo2007. – September 4–7, 2007. – Julich, Germany.
5. The Message Passing Interface (MPI) standard [interactive]. Accessed at: http://www-unix.mcs.anl.gov/mpi/. – 2000.
6. **Londre T. and Rhee N. H.** Numerical Stability of the Parallel Jacobi Method // SIAM J. Matrix Anal. Appl. – Apr. 2005. – Vol. 26, No. 4. – P. 985–1000.
7. **Ian Foster, Carl Kesselman.** The Grid – Blueprint for a new computing infrastructure. – Morgan Kaufmann. – 1999.
8. **Park J. H. and Dai H. K.** Reconfigurable hardware solution to parallel prefix computation // J. Supercomput. – Jan. 2008. – Vol. 43, No. 1. – P. 43–58.
9. **Hill M. D. and Marty M. R**. Amdahl's Law in the Multicore Era // Computer. – Jul. 2008. – Vol. 41, No. 7. – P. 33–38.
10. **Montella R., Agrillo G., Mastrangelo D., and Menna M.** A globus toolkit 4 based instrument service for environmental data acquisition and distribution // Proceedings of the Third international Workshop on Use of P2p, Grid and Agents For the Development of Content Networks. – Boston, MA, USA. – June 23, 2008. – P. 21–28.

Received 2009 02 15

I. Ungurean, S. G. Pentiuc, V. Gaitan. Performance Evaluation of an Experimental Grid Computer using MPI Applications // Electronics and Electrical Engineering. – Kaunas: Technologija, 2009. –No. 5(93). – P. 55–58.

A discussion about parallel programs performance is presented. These programs were developed using the OpenMPI implementation of the Message Passing Interface (MPI) standard and were run on an experimental grid computer made of 7 desktop computers. These test applications measure execution times for applications implementing the Jacobi approximation for a linear system of equations. The output of the test application is the processing speed, measured in MFlops and obtained using more methods in order to test more aspects of inter-process communication. The final purpose is to see increasing performance in a Grid by increasing the number of computing nodes. Ill. 8, bibl. 10 (in English; summaries in English, Russian and Lithuanian).

И. Унгуреан, С. Г. Пентюс, В. Гайтан. Оценка производительности экспериментальной сети компьютеров, используя MPI программы // Электроника и электротехника. – Каунас: Технология, 2009. – № 5(93). – С. 55–58.

Приведен анализ производительности параллельно работающих программ. Программы созданы используя «Open MPI» – версию «Message Passing Interface» (MPI) открытого кода. Они работали в экспериментальной сети из 7-и компьютеров. Программы измеряли продолжительность Jacobi аппроксимации системы линейных уравнений. Скорость обработки информации измерялась по числу операций подвижной запятой. Стремясь протестировать как можно больше аспектов процессных коммуникаций, для измерений использовалось несколько рвзличных методов. Целью было связать производительность сети с количеством ее узлов. Ил. 8, библ. 10 (на английском языке; рефераты на английском, русском и литовском яз.).

I. Ungurean, S. G. Pentiuc, V. Gaitan. Eksperimentinio kompiuterių tinklo našumo įvertinimas naudojant MPI programas // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2009. – Nr. 5(93). – P. 55–58.

Pateikta lygiagrečiai veikiančių programų našumo analizė. Programos buvo sukurtos naudojant „OpenMPI“ – atvirojo kodo „Message Passing Interface“ (MPI) standarto versiją. Jos buvo vykdomos naudojant eksperimentinį septynių kompiuterių tinklą. Programos matavo tiesinės lygčių sistemos Jakobio aproksimacijos trukmę. Informacijos apdorojimo sparta matuota pagal slankiojo kablelio operacijų skaičių. Siekiant testuoti kuo daugiau procesinių komunikacijų aspektų, matavimui taikyti keli skirtingi metodai. Tikslas – susieti tinklo našumą su jo mazgų skaičiumi. Il. 8, bibl. 10 (anglų kalba; santraukos anglų, rusų ir lietuvių k.).

DOI: 10.5755/j02.eie.10181