An MPI-Based System for Testing Multiprocessor and Cluster Communications

Alexey Salnikov and Dmitry Andreev
({salnikov,andreev}@angel.cs.msu.su)

Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University

Introduction

It is difficult to predict duration of message transfer between two processors in multiprocessor or cluster communications. The difficulty is caused mainly by a large number of communicating processes and/or components of a cluster.

Thus, there is a crucial need to develop intellectual testing applications and visualization tools for it.

We have developed such application on the base of MPI library.

Methodology of testing

The user specifies several parameters for testing application: type of test, parameters of the background communications and parameters of goal communications (an interval of message length, a step of message length and a number of repeats for each message length).

The "network_test" application begins its work with the lowest message length in the given interval, upon each step it increases the message length, it stops after reaching the maximal possible message length.

For each message length the application performs data transmissions through communications. For each pair of processors i and j the average duration of a transmission and other standard statistical parameters are computed. The duration is estimated by the MPI_Wtime function.

The obtained matrices form the result of a test, they are stored in four text files.

Results file format (all parameters are duplicated in command line)

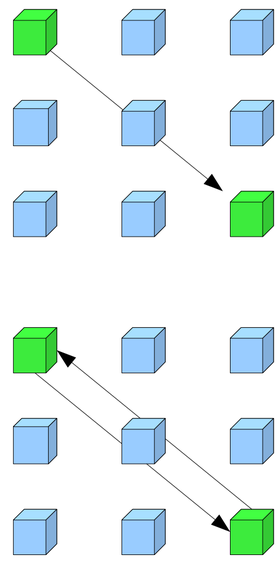
```
processors 100                  - number of MPI-processes
test type "all_to_all"          - one of the testing modes
data type "minimum"             - type of data stored in this  file
begin message length 0          - message length that will be used on
                                  application start
end message length 100000       - upper border of messages length
                                  interval
step length 1000                - step value
noise message length 0          - message length for noise procs for
                                  test_noise and test_noise_blocking
number of noise messages 1      - number of noise messages emmited
                                  by earch noise process
number of noise processes 0     - noise processes
number of repeates 300          - number of testing iteration for one
                                  message length. Used to counting
                                  median, deviation, and so one.

hosts:
<list of hosts>
...

Message length 0
<matrix for this message length>
...
```

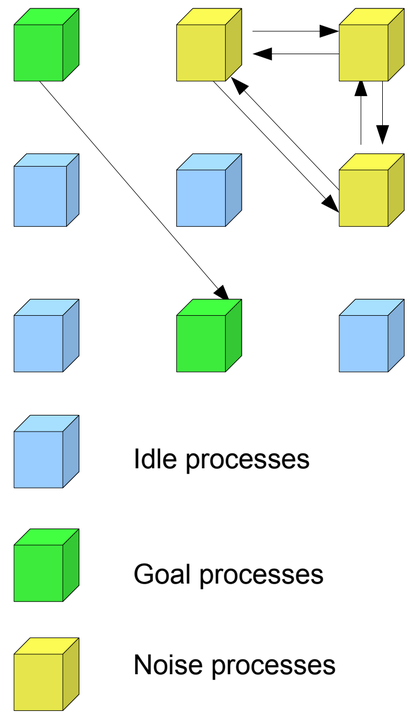
Testing modes

one_to_one MPI-process with rank 0 has a cycle where it go over all possible combinations of processes pairs. Then it sends an information to both processes in pair with their roles and partner's rank. Then one MPI-process from the pair sends messages to the other which receives them. The duration of MPI_Recv call is measured. After all messages are transmitted each process in pair sends a confirmation to the process with rank 0.



async_one_to_one mode is analogous to one_to_one mode but messages are transmitted in asynchronous manner. Processes in the pair use calls MPI_Isend and MPI_Irecv simultaneously in contradict directions. Duration of MPI_Wait are measured only for one process in the pair.

test_noise and *test_noise_blocking* modes are combinations of all_to_all and one_to_one. The process with rank 0 divides all processes into three non-overlapping sets: “goal processes”, “idle processes” and “noise processes”. It uses MPI_Bcast call to broadcast information on the role of each process. Then by means of MPI_Reduce it collects confirmations. The “goal processes” form a pair (similar to one_to_one or async_one_to_one modes) where the durations of MPI_Recv will be measured. The “noise processes” are chosen randomly. They provide background in the cluster communications by sending a number of “noise” messages using the all_to_all-like interaction scheme.



The *all_to_all* mode strives for condition where all the processes transmit messages to each other simultaneously. Each process sends messages to all other processes and itself with MPI_Isend and receives messages with MPI_Irecv. There is a loop where it waits until any MPI_Irecv finishes with call MPI_Waitany and measure the duration of this message exchange.

GUI description

GUI is a Sun Java 1.5 application designed to visualize results of communications testing with three modes of data visualization.

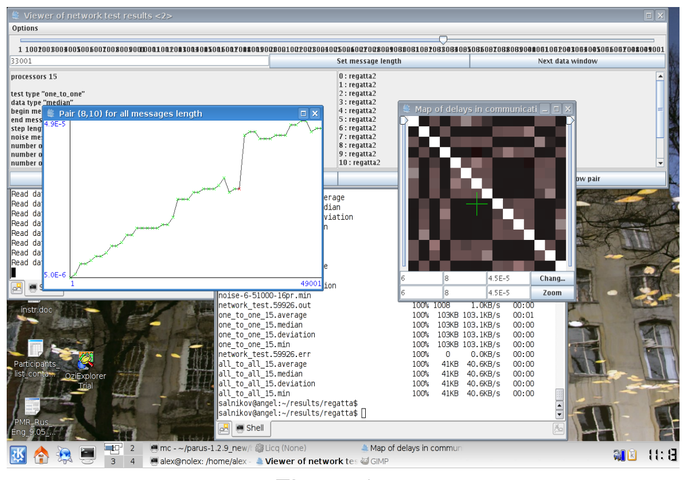


Figure 1

In the first mode a matrix of delays for a fixed message length is drawn. This mode has two internal modes of data normalization: the local mode in one matrix and the global mode in all results. The intensity of black corresponds to the normalized duration of transmitting. The min value is converted to the white color and the max value is converted to the black color. The intensity of red shows the ratio between the deviation and the test result.

In the second mode the user chooses one row or column in matrix and the program draws this for all messages length. This mode highlights delays for one fixed MPI-process. See *Fig. 2.*

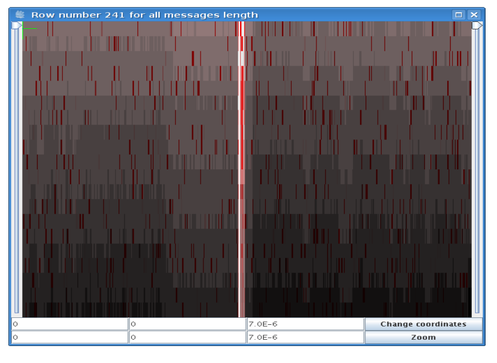


Figure 2

In the third mode a plot for chosen pair of MPI-processes is built. See *Fig. 1.*

Image zooming and black/white levels adjusting are available.

Results

The applications have been tested on mvs100k (cluster of 470 nodes with four Intel Xeon 5160 processors which are connected through Infiniband network) and IBM pSeries 690 (SMP system with 16 processors in our configuration).

We test all modes of network_test application on mvs100k and IBM pSeries 690. Unfortunately testing modes noise and noise_blocking are very slow with many processors so we have not data for mvs100k.

The network_test application allows to highlight multiprocessor or cluster topology. On the *Figure 1* we can see the cache hierarchy for two MCM modules in IBM pSeries 690.

On the *Figure 2* represented result of one_to_one test mode for 500 processors on mvs100k. There is a matrix of delays for all messages length that process with rank 241 receive from other processes. The influence of deviation seems to become neglected with grows of message length.

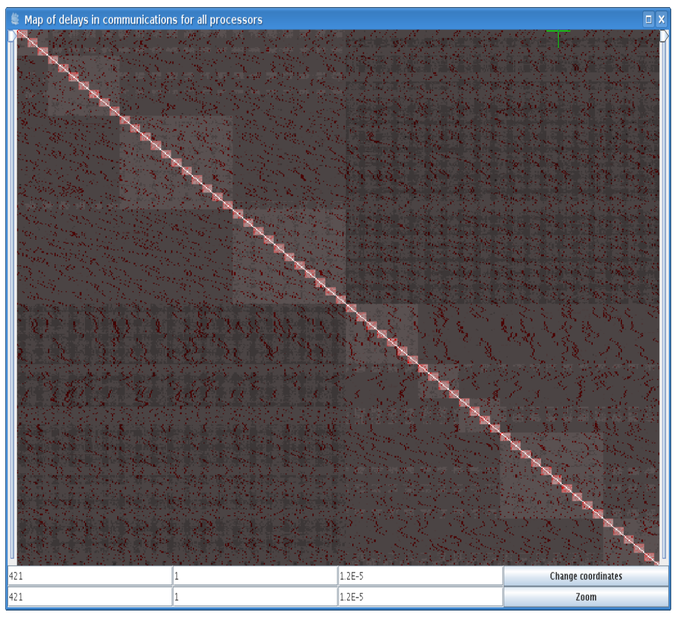


Figure 3

There is a hierarchical cells structure on the Fig. 3 that gives an information on the mvs100k machine topology. The eight pixel of size cells closest to the matrix diagonal correspond shared memory in one cluster node. Each of the surrounding cells corresponds to one of the hardware switches that connect cluster nodes. The next surrounding cells layer corresponds to higher level switch hierarchy. Red moire corresponds to electromagnetic inducings.

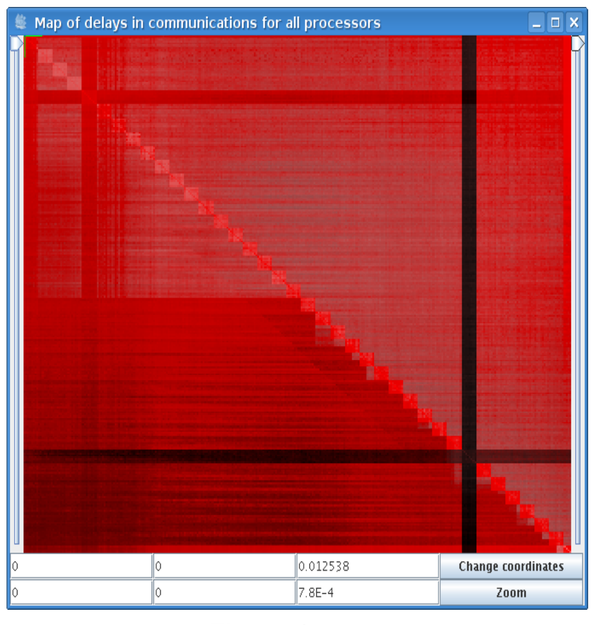


Figure 4

The *Fig. 4* is drawn by results of testing mvs100k in all_to_all mode with 300 processors. We can see high value of deviation. Cells that conform with shared memory are remained in this testing mode similar to the one_to_one mode. We see several cluster nodes that have discriminated behaviour presumably by reason the activity of operating system.

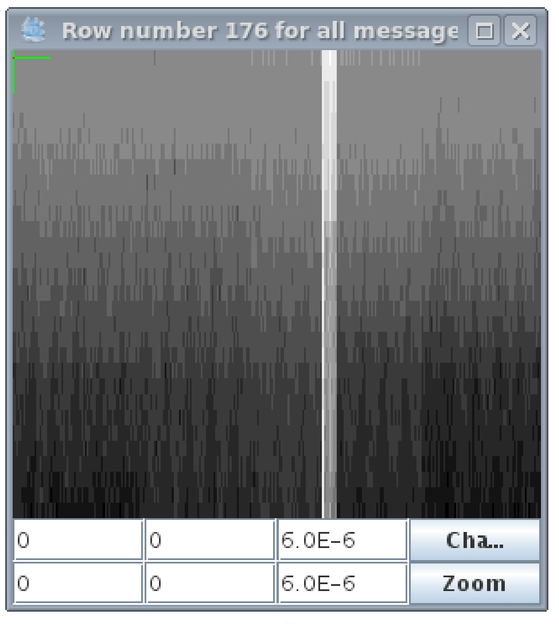


Figure 5

Fig. 5 shows the results of testing mvs100k in async_one_to_one mode with 500 processors. We see increase of the communication heterogeneity with the growth of message length.

The network_test application and GUI are available from download page of PARUS project:

http://parus.sf.net