MPI Labs

Yvon Kermarrec

Télécom Bretagne
September 2012

Aims of these exercises :

- To develop a data-processing solution for a problem which requires an important computing time and to use the computing resources available on a network.
- To evaluate and understand the various modes of distribution and their benefit/respective disadvantages.
- To use a parallel programming (Open MPI) and to exploit the traces to understand what occurs in a distributed system and to contribute to the development of the applications.

Exercice 0 :

Generate, run the executable and read the code: understand the MPI calls from experience and by reading the MPI documentations.

Exercice 1 :

This is an initial exercise in order to discover the MPI environment and its tools.

- generate the executable file with the Unix make command
- start the MPI program with the mpirun command and indicate the name of the executable mpi-example and give in the argument a string of characters
- Activate the various traces and understand their meanings and how they can contribute to the debugging phase of an application.

Exercise 2:

- You can generate an executable file from the Makefile with the Unix command make : this program "mpi-ring" requires an integer argument. By setting different values and running it, understand how the program works and what it does.
- This programs does not stop correctly (it does not terminate well and there are pending messages).
- Modify the code so that the termination is correct.

Exercise 3:

- Execute make and start the execution on the cluster (a set of PCs) and provide a directory path as the argument (e.g.; /etc).
- This program is very similar to the Unix du (disk usage) command. This Unix command computes the size of a file tree. Here we split the load of the computation by splitting the tree and assigning to each of the processor one sub tree (in an iterative way).
- From the traces, what can you determine on the load balancing features? Explain how the master and the slaves work. Explain why and how the slowest slaves constraints the others.
- Propose an alternate solution where the slave (a nice and willing slave !) asks for work to perform, code it and analyze the outcomes and benefits.

Exercise 4 :

The aim of this exercise is to computer Mandelbrot set and produce interesting visual results. Mandelbrot set requires intensive computations and therefore are good candidate to be run on a cluster. You can find an interesting survey on the topic at :

http://www.math.utah.edu/~alfeld/math/mandelbrot/mandelbrot.html

for this exercise, we should monitor the complexity and the computation time of the Mandelbrot set. Use the various MPI functions as defined in Exo0 directory.

- in the source, you can generate mandel-basic-mpi which provides a slow and complex solution. You can execute it and display the result with the Unix command 'xv mandel.ppm'.
- explain why (from an analyse of the trace) why it takes so long to compute the image. Indicate in particular what is the complexity of the solution implemented (in terms of messages that are exchanged).
- Propose a modification of the program: instead that a point is computed, the slave computes a complete line (instead sending one single point, we send a complete line of points).
- Check that the program still works and produces a similar image after your alteration.
- In a second stage, you can send a block of lines instead of a single line. Measure the impact on the response time.

- Modify then the various parameters of the Mandelbrot set to generate different images!

Exercise 5 :

The aim of this exercise is to investigate a classic example of intensive math with matrix multiplication. The code has been completed and works fine. From the traces, what can you determine on the use of the CPUs ? explain how the program works and how the data flows between the nodes (use graphical notation to describe).