Distributed Systems – TD1 : Introduction to MPI

{ Bogdan.Pasca } @ens-lyon.fr

18 septembre 2009

What is MPI ?

Go to http://perso.ens-lyon.fr/bogdan.pasca/teaching_distribsys.html and chose one of the two presentations on MPI (one in French and one in English) for more details.

How to ?

Compile C/C++ program which uses MPI

```
[mpicc|mpicxx] -o your_executable program_name.[c|cpp]
```

Run the obtained executable

```
mpirun -np 3 your_executable
```

where `-np 3` means that 3 processes are executed.

Optional step

Add your SSH public key to your `authorized_keys` file. This is usually found in `.ssh` from your home folder.

Exercises

1. Write an MPI program that prints for each process :

   `Hello, I'm process X out of Y`

   where X is the current process ID and Y is the total number of processes.
2. How do I assign different work to each processor, if all processors run the same program ?
3. Modify the hello wold program such that odd processes display a different type of message.
4. Write a program that given a common value for x, computes y such that :

```
for process 0, y=x^2
            1, y=x^3
...
            k, y=x^k+2
```

and displays the results.

5. Starting from the previous program, send all the computed values of y to process 0. Process 0 will display the values of y, and the process that computed this value of y. Then it will perform an average on these values and display the result.
6. Write a program in which a token moves in a ring of processes. The token starts-off in process 0, and then it passes from process to process (with modification) to finally return to process 0.
7. Write a program in which two processes want to exchange messages in the same time (send then receive). Test this program with classical blocking communications. Correct this version with non-blocking communications.
8. Write a program in which a process sends a table of integers to all the other processes.
9. Write a program which computes the sum of all process IDs and which sends this result to all processes. Write two versions, one with reduction and broadcast, and another one with global reduction (`MPI_Allreduce`).
10. Write a program in which the processes synchronize using a synchronization barrier. Modify the latency of each process using loops or `sleep`.