Homework 1 - The Beowulf cluster and MPI

Course : Operating Systems
Deadline :Friday , 2019/3/15 18:00

Overview

In this project, we are going to practice how to build the Beowulf cluster, using it to execute a program in parallel, and we will write our own parallel program with MPICH.

Tutorials

The Beowulf cluster

First, we need to build the Beowulf cluster which use mpi as the communication channel.

1. Create 3 nodes(VMs) on AWS and rename your VMs to **master**, **slave1**, and **slave2**.
2. Install **Network File System** in each nodes. Install nfs-server in the master, and install nfs-client in slaves.
3. Create a directory in master and share the directory to slaves.
4. Setup ssh configurations in each node to make sure that master can access slaves without password-checking.
5. Install **MPICH** in each node.
6. Validation
    1. Download the attached source code - mpi_hello.c.
    2. Compile the code with the MPI C compiler.

```
$ mpicc mpi_hello.c -o mpi_hello
```

    3. Run the program by mpiexec.

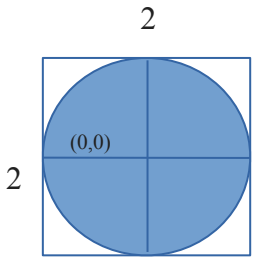
```
$ mpiexec -n <total cpu number> -host master,slave1,slave2 ./mpi_hello
// -n means the number of cores to ask for.
// This should not be more than the sum of cpu cores in all nodes.
```

    4. You would see some message like this:

```
// If you have 6 cpus in total.
Hello from processor 0 of 6
Hello from processor 1 of 6
Hello from processor 2 of 6
Hello from processor 3 of 6
Hello from processor 4 of 6
Hello from processor 5 of 6
// That will be probably unordered.
```

Pi calculation

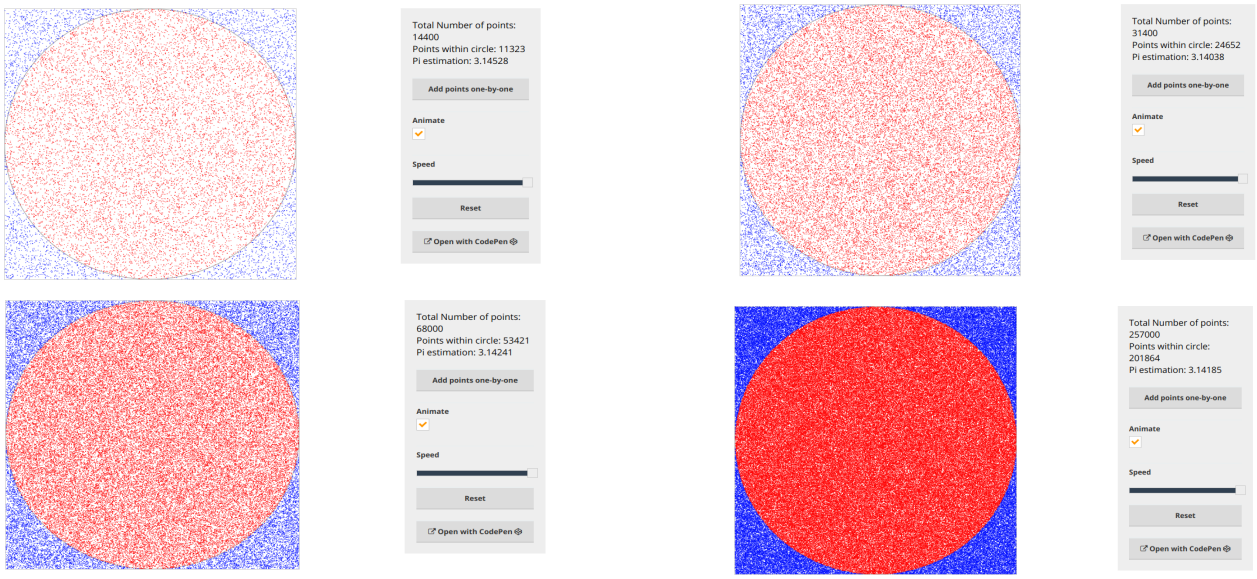
There are many ways to calculate Pi. In this part, we are going to parallel Pi calculation using the Monte Carlo method. The main idea of the Monte Carlo method is to solve the problem by using random numbers and the formula of the area.
Suppose there is a square with a side length of two, and a circle insides the square with a radius of one, as shown below:



We can find that the area of the circle is Pi. Now we toss a certain number of darts into the square and all darts are uniformly distributed on the square. Suppose the number of the darts we toss is N(Darts will always hit the square), and the number of darts inside the circle is C. So we can have the following equation:

```
C:N = Pi:4 => Pi = 4 * C / N
```

Using this method, we can estimate the value of Pi. If we want more accurate Pi, we can just make the value of N very large.



Benchmark running

We have provided the code using the Monte Carlo method which runs sequentially.
1. Download attached source codes – test.c.
2. Compile the code with the MPI C compiler.

```
mpicc test.c -o test
```

3. Run the program by mpirun.

```
mpirun -n 1 -host master ./test
```

HINT: Because our seed is a constant, so your answer should always be `Pi = 3.141704` .

MPI programming

In this part, we are going to write our own parallel program to calculate Pi with Monte Carlo algorithm. You will need to implement your program in two versions:(1) Using MPI Send & Receive function (2) Using MPI Reduce function. Your program should be scalable which means it can be executed on two or three CPU cores and it should allow us to input a different number of tosses. The seed of srand() for each process should be a constant: rank*<constant> to ensure the consistency of each execution's result.

We will execute your program by the following instruction:
`mpirun -n <total cpu number> ./<program name> <number of tosses>`
Note:If your instruction's parameter number is wrong, such as you missed the <number of tosses> field. You should output the following error message:
`Usage: mpirun -n <total cpu number> ./<program name> <number of tosses>`

Your output should have the Pi value and the time duration of the Pi calculation:

```
ubuntu@master:/mpishare$ mpirun -n 3 -host master,slave1,slave2 ./pi_reduce 10000000
pi = 3.142062 duration = 0.102316
ubuntu@master:/mpishare$ mpirun -n 2 -host master,slave1 ./pi_reduce 10000000
pi = 3.141672 duration = 0.153336
ubuntu@master:/mpishare$ mpirun -n 1 -host master ./pi_reduce 10000000
pi = 3.142256 duration = 0.303782
ubuntu@master:/mpishare$ mpirun -n 1 -host master ./pi_reduce
Usage: mpirun -n <total cpu number> ./<program name> <number of tosses>
```

File submission

Create a directory named <student_id>. The structure should be :
<student_id>
    |—<student_id>_pi_comm.xx
    |—<student_id>_pi_reduce.xx

Zip the directory into <student_id>.zip and upload it to the new E3 platform.
TAs would validate your source codes by cheating detection. Please finish the assignment by yourself.

Grades

In each part, we have prepared several oral questions to make sure that you finished the homework by yourself.

If you meet the requirements and pass oral defense, you can get the following grades:

- The Beowulf cluster **(40%)**
- Benchmark running**(20%)**
- MPI programming**(40%)**

References

MPI

https://en.wikipedia.org/wiki/MessagePassingInterface#Example_program
http://mpitutorial.com/tutorials/mpi-send-and-receive/
https://www.mpich.org/static/docs/v3.2/www3/

MPI for python

https://mpi4py.readthedocs.io/en/stable/

Calculate PI

https://en.wikipedia.org/wiki/Monte_Carlo_method
https://academo.org/demos/estimating-pi-monte-carlo/