Supercomputers Architecture

Hands on 2

November 24th, 2015

Constantino Gomez

Albert Segura

Cristobal Ortega

Exercise 1: Create a “Hello World” program in MPI and Compile it.

```
#include "mpi.h"
#include <stdio.h>

int main (int argc, char **argv) {
      MPI_Init (&argc, &argv);
      int rank, size;
      MPI_Comm_rank(MPI_COMM_WORLD, &rank);
      MPI_Comm_size(MPI_COMM_WORLD, &size); /* How many are we? */

      printf ( "I am %d of %d\n", rank, size );

      MPI_Finalize();
      return 0;
}
```

Exercise 2: Run the “Hello World” program. How many lines contain the file hello.out? Why?

```
~/handson2> mpirun -np 4 ./a.out
I am 3 of 4
I am 0 of 4
I am 1 of 4
I am 2 of 4
```

Because we spawned 4 processes setting the np parameter to 4

Exercise 3: submit the previous mpi hello world program. Create the LSF file according the following suggestions:

```
~/handson2> cat job.sh
#BSUB -J hello_world
#BSUB -o output_%J.out
#BSUB -e output_%J.err
#BSUB -W 00:15
#BSUB -n 16
#BSUB -R "span[ptile=16]"
#BSUB -x
#BSUB -U sa

#Executing
mpirun ./hello_mpi
```

Exercise 4: Do exactly the same as previous exercise, but now with 4 processes per node. Compare the two executions (exercise 3 vs 4).

```
16 processes per node:
Job was executed on host(s) <16*s03r1b72>, in queue <sequential>,
as user <sam14021> in cluster <mn3>.


4 processes per node:
Job was executed on host(s) <4*s03r1b72>, in queue <training>, as
user <sam14021> in cluster <mn3>.
                          <4*s03r1b74>
                          <4*s03r1b71>
                          <4*s03r1b70>
```

The actual output written in the program is the same, but with 4 processes per node the job is assigned to the queue training instead of the sequential queue.

Exercise 5.a: Write a mpi_trap.c program based on the code given in the theory class that estimates the integral from a to b of f(x) in n intervals.
Compile and run it.

```
~/handson2> mpicc trapezoidal.c
~/handson2> mpirun -np 4 ./a.out
With n = 1024 trapezoids, our estimate
of the area from 0.000000 to 3.000000 = 9.000004291534424
```

Exercise 5.b (OPTIONAL): The same mpi_trap.c program that accept the endpoints of the interval of integration and the number of trapezoids as an input.

In trapezoidal.c, uncomment Get_data call

Exercise 5.c: Create a lsf jobscript and submit it with 8 processors. Include the lsf jobscript in the answer.

LSF:

```
#BSUB -J trapezoidal
#BSUB -o output_%J.out
#BSUB -e output_%J.err
#BSUB -W 00:15
#BSUB -n 8
#BSUB -R "span[ptile=16]"
#BSUB -x
#BSUB -U sa
```

```
#Executing
mpirun ./trapezoidal

Results:
~/handson2> cat output_1291960.out
With n = 1024 trapezoids, our estimate
of the area from 0.000000 to 3.000000 = 9.000004291534424
```

Exercise 6: Take the time for executing the sequential version of exercise 12 of previous Hands-on 1 for different N values. Compare the results. In the answer also include the LSF file used.

1024	**2048**	**4096**	**8192**	**16384**	**32768**
0,01	0,07	0,28	1,13	4,4	17,55

Exercise 7: Answer exercises 3 and 4 again taking the time.

	Exercise 3	**Exercise 4**
Execution time (seconds)	3,15	3,06

We cannot appreciate too much difference in terms of execution time

Exercise 8: Take the time for executing the program version of exercise 5 for different input values and with different number of processors (describe the results).

	1 process	**2 processes**	**4 processes**	**8 processes**
Execution time (seconds)	0,85	1,12	1,12	1,44

It looks like there is more overhead in terms of MPI communication than actual work, that can be the cause why are seeing the time is increasing instead of decreasing. Probably with a larger input we could see speedup