Parallelization: Area Under a Curve

By Aaron Weeden, Shodor Education Foundation, Inc.

Exercise 3

Part I – strong scaling

This exercise will take you through filling out Table 1, which will indicate how many seconds it takes to execute each program for the given numbers of nodes and cores per node used.

This exercise requires the same constraints for a cluster as specified in Exercise 2.

The program is being scaled through strong scaling, so the number of rectangles stays constant at 100,000,000.

# of nodes used	# of cores per node used	Total # of cores	Total # of Rectangles	Serial	OpenMP	MPI	Hybrid
1	1	1	100,000,000				
1	2	2	100,000,000				
1	4	4	100,000,000				
2	1	2	100,000,000				
2	2	4	100,000,000				
2	4	8	100,000,000				
4	1	4	100,000,000				
4	2	8	100,000,000				
4	4	16	100,000,000				
8	1	8	100,000,000				
8	2	16	100,000,000				
8	4	32	100,000,000				

Table 1 – Strong Scaling, 100,000,000 Rectangles

1. Copy the area directory to your account on al-salam using Secure Shell Copy (`scp`):
   ```
   $ scp –r area yourusername@cluster.earlham.edu:
   ```

2. Log into the cluster using a secure shell (ssh):
   ```
   $ ssh yourusername@cluster.earlham.edu
   ```

```
$ ssh as0
$
```

3. Change directories into the area code directory of your choice (C or Fortran90):
```
$ cd area/C
$
```
OR
```
$ cd area/Fortran90
$
```

4. Compile the serial version of the code:
```
$ make serial
make area.serial
make[1]: Entering directory
`/nfs/cluster/home/amweeden06/area/C'
gcc  area.c -o area.serial
make[1]: Leaving directory
`/nfs/cluster/home/amweeden06/area/C'
$
```

5. Open a new file called `area.serial.qsub` and add the following lines:
```
$ cat area.serial.qsub
#PBS -q ec
#PBS -o area.serial.out
#PBS -e area.serial.err

cd $PBS_O_WORKDIR

time ./area.serial -l 0.0 -r 10.0 -n 100000000
$
```

The `time` command will show us how much time it takes the program to run.

6. Submit a job to the scheduler using `area.serial.qsub`:
```
$ qsub area.serial.qsub
19214.as0.al-salam.loc
```

7. Monitor the job with `qstat` until the job ID is unknown (i.e. until the job finishes):
```
$ qstat 19214
qstat: Unknown Job Id 19214.as0.al-salam.loc
$
```

8. Check the contents of `area.serial.err` to see how long it took to run the program:

   ```
   $ cat area.serial.err

   real 0m3.021s
   user 0m2.765s
   sys  0m0.257s
   $
   ```

   This shows three times. The `real` time is the time spent executing the program from start to finish, including times during which the program is interrupted by other programs using the operating system or is blocking waiting for I/O. `User` time is the time spent by the program in user space, and `sys` time is the time spent by the program in the kernel. We are interested in how much time it took the program to finish once it was started, so we will consider `real` time in this exercise.

9. Enter your result for `real` time in Table 1 under the "Serial" column in the row with 1 node and 1 core per node used.

10. Open `area.serial.qsub` and change the following **bold** line:

```
#PBS -q ec
#PBS -o area.serial.out
#PBS -e area.serial.err
#PBS -l nodes=1:ppn=2

cd $PBS_O_WORKDIR

time ./area.serial -l 0.0 -r 10.0 -n 100000000
```

11. Submit a job to the scheduler, wait for it to finish, and then show the result:

   ```
   $ qsub area.serial.qsub
   19215.as0.al-salam.loc
   $ qstat 19215
   qstat: Unknown Job Id 19215.as0.al-salam.loc
   $ cat area.serial.err

   real 0m3.010s
   user 0m2.766s
   sys  0m0.243s
   $
   ```

12. Enter your result for `real` time in Table 1 under the "Serial" column in the row with 1 node and 2 cores per node used.

13. Repeat steps 11 – 14, but this time change **ppn=2** to **ppn=4** in `area.serial.qsub` and enter the result in the row with 1 node and 4 cores per node used.

14. Open a new file called `area.openmp.qsub` and enter the following text:

```
#PBS -q ec
#PBS -o area.openmp.out
#PBS -e area.openmp.err
#PBS -l nodes=1:ppn=1

export OMP_NUM_THREADS=8

cd $PBS_O_WORKDIR

time ./area.openmp -l 0.0 -r 10.0 -n 100000000
```

15. Submit a job to the scheduler, wait for it to finish, and show the results:

```
$ qsub area.openmp.qsub
19217.as0.al-salam.loc
$ qstat 19217
qstat: Unknown Job Id 19217.as0.al-salam.loc
$ cat area.openmp.err

real 0m0.742s
user 0m3.043s
sys  0m0.287s
$
```

16. Enter your result for `real` time in Table 1 under the "OpenMP" column in the row with 1 node and 1 core per node used.

17. Repeat steps 11 – 15, making sure to work with `area.openmp.qsub` and `area.openmp.err` rather than `area.serial.qsub` and `area.serial.err`. When you finish, Table 1 should have values in the "Serial" and "OpenMP" columns for the first three rows.

18. Open a new file called `area.mpi.qsub` and enter the following text:

```
#PBS -q ec
#PBS -o area.mpi.out
#PBS -e area.mpi.err
#PBS -l nodes=1:ppn=1
```

```
cd $PBS_O_WORKDIR

time mpirun -np 1 -machinefile $PBS_NODEFILE ./area.mpi \
-l 0.0 -r 10.0 -n 100000000
```

19. Submit a job to the scheduler with `area.mpi.qsub`, output the result (`area.mpi.err`), and add the `real` time to Table 1 under the "MPI" column for 1 node, 1 core per node used.

20. Change the following bold sections of `area.mpi.qsub`:

```
#PBS -q ec
#PBS -o area.mpi.out
#PBS -e area.mpi.err
#PBS -l nodes=1:ppn=2

cd $PBS_O_WORKDIR

time mpirun -np 2 -machinefile $PBS_NODEFILE ./area.mpi -l
0.0 -r 10.0 -n 100000000
```

21. Repeat step 21 but enter the result in the row with 1 node, 2 cores per node used.

22. Repeat steps 22 and 23, but change the 2's to 4's.

23. Change the following bold sections of `area.mpi.qsub`:

```
#PBS -q ec
#PBS -o area.mpi.out
#PBS -e area.mpi.err
#PBS -l nodes=2:ppn=1

cd $PBS_O_WORKDIR

time mpirun -np 2 -machinefile $PBS_NODEFILE ./area.mpi -l
0.0 -r 10.0 -n 100000000
```

24. Enter the result of running this and finding the sum in Table 1 in the row with 2 nodes and 1 core per node used.

25. Repeats steps 25 and 26, but change **ppn=1** to **ppn=2** and **–np 2** to **–np 4**. Enter the result in the row with 2 nodes and 2 cores per node used.

26. Continue to fill out the table under the MPI column. Whenever you change `nodes=X`, `ppn=Y`, and `–np Z`, make sure that `Z = X * Y`. For example,

when you run 8 nodes with 4 cores per node used, make sure you specify 32 MPI processes with `-np`.

27. Open a new file called `area.hybrid.qsub` and enter the following text:

```
#PBS -q ec
#PBS -o area.hybrid.out
#PBS -e area.hybrid.err
#PBS -l nodes=1:ppn=1

export OMP_NUM_THREADS=8

cd $PBS_O_WORKDIR

time mpirun -np 1 -machinefile $PBS_NODEFILE \
./area.hybrid -l 0.0 -r 10.0 -n 100000000
```

28. Scale the hybrid jobs just as you did the MPI jobs in steps 21 – 28.

    When you finish this step, the table should be complete (except for the serial and OpenMP columns for more than 1 node – pure serial and pure OpenMP programs have no meaningful result if they are run using distributed memory, so we leave these cells blank).

Part II – weak scaling

In weak scaling, the problem size varies as the number of cores increases. Thus, instead of using a constant 100,000,000 rectangles, we increase the number of rectangles as we increase the number of cores. You can choose the factor by which we increase the number of rectangles, but we will use 100,000,000 rectangles per core as the example in this case.

# of nodes used	# of cores per node used	Total # of cores	Total # of rectangles	Serial	OpenMP	MPI	Hybrid
1	1	1	100,000,000				
1	2	2	200,000,000				
1	4	4	400,000,000				
2	1	2	200,000,000				
2	2	4	400,000,000				
2	4	8	800,000,000				
4	1	4	400,000,000				
4	2	8	800,000,000				

4	4	16	1,600,000,000				
8	1	8	800,000,000				
8	2	16	1,600,000,000				
8	4	32	3,200,000,000				

Table 2

1. Open `area.serial.qsub` and make sure it looks like this to start:

```
#PBS -q ec
#PBS -o area.serial.out
#PBS -e area.serial.err
#PBS -l nodes=1:ppn=1

cd $PBS_O_WORKDIR

time ./area.serial -l 0.0 -r 10.0 -n 100000000
```

2. Fill in the first 3 rows of the table under the Serial column as you did in Part I, but make sure that for every job you submit the value for `-n` is equal to the value of `nodes=` times the value of `ppn=` times 100,000,000.

3. Fill out the table for OpenMP, MPI, and Hybrid. The PBS files should look like the following to start:

    a. **`area.openmp.qsub`:**

```
#PBS -q ec
#PBS -o area.openmp.out
#PBS -e area.openmp.err
#PBS -l nodes=1:ppn=1

export OMP_NUM_THREADS=8

cd $PBS_O_WORKDIR

time ./area.openmp -l 0.0 -r 10.0 -n 100000000
```

    b. **`area.mpi.qsub`:**

```
#PBS -q ec
#PBS -o area.mpi.out
#PBS -e area.mpi.err
#PBS -l nodes=1:ppn=1

cd $PBS_O_WORKDIR
```

```
   time mpirun -np 1 -machinefile $PBS_NODEFILE ./area.mpi \
   -l 0.0 -r 10.0 -n 100000000
```

c. **area.hybrid.qsub**:

```
#PBS -q ec
#PBS -o area.hybrid.out
#PBS -e area.hybrid.err
#PBS -l nodes=1:ppn=1

export OMP_NUM_THREADS=8

cd $PBS_O_WORKDIR

time mpirun -np 1 -machinefile $PBS_NODEFILE \
./area.hybrid -l 0.0 -r 10.0 -n 100000000
```