Parallelization: Sieve of Eratosthenes

By Aaron Weeden, Shodor Education Foundation, Inc.

Exercise 2 – Scaling the Sieve of Eratosthenes on a Cluster

Part I – strong scaling

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

The program is being scaled through strong scaling, so we are always trying to find the primes under the constant value 6,000,000.

Walltimes for Strong Scaling, Finding Primes Under 6,000,000

# of nodes used	Total # of cores	Serial	OpenMP	MPI	Hybrid
1	4				
2	8				
3	12				
4	16				
5	20				
6	24				
7	28				
8	32				
9	36				
10	40				
11	44				
12	48				

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

2. Log into al-salam using a secure shell (ssh):
```
$ ssh yourusername@cluster.earlham.edu
$ ssh as0
```

3. Change directories into the `sieve` code directory:
```
$ cd sieve
```

```
$
```

4. Compile all versions of the code:
   ```
   $ gcc –o sieve.serial sieve.serial.c –lm
   $ gcc –fopenmp –o sieve.openmp sieve.openmp.c –lm
   $ mpicc –o sieve.mpi sieve.mpi.c –lm
   $ mpicc –fopenmp –o sieve.hybrid sieve.hybrid.c –lm
   $
   ```

5. Open a new file called `sieve.serial.qsub` in **vi** and add the following lines:

```
#PBS -q ec
#PBS -o sieve.serial.out
#PBS -e sieve.serial.err

cd $PBS_O_WORKDIR

time ./sieve.serial -n 6000000
```

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

6. Submit a job to the scheduler using `sieve.serial.qsub`:
   ```
   $ qsub sieve.serial.qsub
   19723.as0.al-salam.loc
   $
   ```

7. Monitor the job with `qstat` until the job ID is unknown (i.e. until the job finishes):
   ```
   $ qstat 19723
   Job id                    Name             User            Time Use S 
   Queue
   ------------------------- ---------------- --------------- -------- - ---
   --
   19723.as0                 ...e.serial.qsub amweeden06             0 R ec

   $ qstat 19723
   qstat: Unknown Job Id 19723.as0.al-salam.loc
   $
   ```

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

   real 0m19.011s
   user 0m18.980s
   sys  0m0.026s
   ```

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 the table under the “Serial” column in the row with 1 node used.

10. Open a new file called `sieve.openmp.qsub` and enter the following text:

```
#PBS -q ec
#PBS -o sieve.openmp.out
#PBS -e sieve.openmp.err

export OMP_NUM_THREADS=8

cd $PBS_O_WORKDIR

time ./sieve.openmp -n 6000000
```

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

```
$ qsub sieve.openmp.qsub
19724.as0.al-salam.loc
$ qstat 19724
qstat: Unknown Job Id 19724.as0.al-salam.loc
$ cat sieve.openmp.err

real 0m2.844s
user 0m22.628s
sys  0m0.034s
$
```

12. Enter your result for `real` time in the table under the “OpenMP” column in the row with 1 node used.

13. Open a new file called `sieve.mpi.qsub` and enter the following text:

```
#PBS -q ec
#PBS -o sieve.mpi.out
#PBS -e sieve.mpi.err
#PBS -l nodes=1:ppn=4
```

```
cd $PBS_O_WORKDIR

time mpirun -np 4 -machinefile $PBS_NODEFILE \
./sieve.mpi -n 6000000
```

14. Submit a job to the scheduler with `sieve.mpi.qsub`, output the result (`sieve.mpi.err`), and add the `real` time to the table under the "MPI" column for 1 node used.

15. Change the following bold sections of `sieve.mpi.qsub`:

```
#PBS -q ec
#PBS -o sieve.mpi.out
#PBS -e sieve.mpi.err
#PBS -l nodes=2:ppn=4

cd $PBS_O_WORKDIR

time mpirun -np 8 -machinefile $PBS_NODEFILE \
./sieve.mpi -n 6000000
```

16. Repeat step 14 but enter the result in the row with 2 nodes used.

17. Repeat steps 15 and 16, but change `nodes=2` to `nodes=3` and `-np 8` to `-np 12`. Enter the result in the row with 3 nodes used.

18. Continue to fill out the table under the MPI column. Whenever you change `nodes=X` and `-np Y`, make sure that `Y = X * 4`. For example, when you run 8 nodes, make sure you specify 32 MPI processes with `-np`.

19. Open a new file called `sieve.hybrid.qsub` and enter the following text:

```
#PBS -q ec
#PBS -o sieve.hybrid.out
#PBS -e sieve.hybrid.err
#PBS -l nodes=1:ppn=4

export OMP_NUM_THREADS=2

cd $PBS_O_WORKDIR

time mpirun -np 4 -machinefile $PBS_NODEFILE \
./sieve.hybrid -n 10000
```

20. Scale the hybrid jobs just as you did the MPI jobs in steps 14 – 19.

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 finding primes under a constant number of 6,000,000, we increase the number as we increase the number of cores. You can choose the factor by which we increase the number, but we will use a factor of 1,000,000 * number of cores as the example in this case.

Walltimes for Weak Scaling, Finding Primes Under 1,000,000 * Number of Cores

# of nodes used	Total # of cores	Finding primes under	Serial	OpenMP	MPI	Hybrid
1	4	4,000,000				
2	8	8,000,000				
3	12	12,000,000				
4	16	16,000,000				
5	20	20,000,000				
6	24	24,000,000				
7	28	28,000,000				
8	32	32,000,000				
9	36	36,000,000				
10	40	4,000,000				
11	44	44,000,000				
12	48	48,000,000				

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

```
#PBS -q ec
#PBS -o sieve.serial.out
#PBS -e sieve.serial.err

cd $PBS_O_WORKDIR

time ./sieve.serial -n 4000000
```

2. Fill in the first rows of the table under the Serial column as you did in Part I.

3. Fill out the table for OpenMP, MPI, and Hybrid, but make sure that for every job you submit the value for `-n` is equal to the value of `-np` times 1,000,000. The PBS files should look like the following to start:

   a. **`sieve.openmp.qsub`:**

```
#PBS -q ec
#PBS -o sieve.openmp.out
#PBS -e sieve.openmp.err

export OMP_NUM_THREADS=8

cd $PBS_O_WORKDIR

time ./sieve.openmp -n 4000000
```

   b. **`sieve.mpi.qsub`:**

```
#PBS -q ec
#PBS -o sieve.mpi.out
#PBS -e sieve.mpi.err
#PBS -l nodes=1:ppn=4

cd $PBS_O_WORKDIR

time mpirun -np 4 -machinefile $PBS_NODEFILE \
./sieve.mpi -n 4000000
```

   c. **`sieve.hybrid.qsub`:**

```
#PBS -q ec
#PBS -o sieve.hybrid.out
#PBS -e sieve.hybrid.err
#PBS -l nodes=1:ppn=4

export OMP_NUM_THREADS=2

cd $PBS_O_WORKDIR

time mpirun -np 4 -machinefile $PBS_NODEFILE \
./sieve.hybrid -n 4000000
```