**Parallelization: Conway’s Game of Life**

**By Aaron Weeden, Shodor Education Foundation, Inc.**

**Exercise 3 – Scaling 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 running the Game of Life for a constant 1000 rows, 1000 columns, and 100 time steps.

**Walltimes for Strong Scaling, 1000 rows, 1000 columns, and 100 Time Steps**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # 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 life directory to your account on al-salam using Secure Shell Copy (scp):

$ scp –r life yourusername@cluster.earlham.edu:

$

1. Log into al-salam using a secure shell (ssh):

$ ssh yourusername@cluster.earlham.edu

$ ssh as0

$

1. Change directories into the life/C or life/Fortran90 code directory:

$ cd life/C

$

OR

$ cd life/Fortran90

$

1. Compile all versions of the code:

$ make all

make clean

make[1]: Entering directory `/nfs/cluster/home/amweeden06/life/C'

rm -f life.{serial,openmp,mpi,hybrid} \*.o

make[1]: Leaving directory `/nfs/cluster/home/amweeden06/life/C'

make life.{serial,openmp,mpi,hybrid}

make[1]: Entering directory `/nfs/cluster/home/amweeden06/life/C'

gcc life.c -o life.serial

gcc -fopenmp -DOPENMP life.c -o life.openmp

mpicc -DMPI life.c -o life.mpi

mpicc -DMPI -fopenmp -DOPENMP life.c -o life.hybrid

make[1]: Leaving directory `/nfs/cluster/home/amweeden06/life/C'

$

1. Open a new file called life.serial.qsub in **vi** and add the following lines:

#PBS -q ec

#PBS -o life.serial.out

#PBS -e life.serial.err

cd $PBS\_O\_WORKDIR

time ./life.serial -r 1000 -c 1000 -t 100

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

1. Submit a job to the scheduler using qsub :

$ qsub life.serial.qsub

19723.as0.al-salam.loc

$

1. 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 life.serial.qsub amweeden06 0 R ec

$ qstat 19723

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

$

1. Check the contents of life.serial.err to see how long it took to run the program:

$ cat life.serial.err

real 0m17.941s

user 0m17.669s

sys 0m0.254s

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.

1. Enter your result for real time in the table under the “Serial” column in the row with 1 node used.
2. Open a new file called life.openmp.qsub and enter the following text:

#PBS -q ec

#PBS -o life.openmp.out

#PBS -e life.openmp.err

export OMP\_NUM\_THREADS=8

cd $PBS\_O\_WORKDIR

time ./life.openmp -r 1000 -c 1000 -t 100

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

$ qsub life.openmp.qsub

19724.as0.al-salam.loc

$ qstat 19724

qstat: Unknown Job Id 19724.as0.al-salam.loc

$ cat life.openmp.err

real 0m12.63s

user 1m39.099s

sys 0m1.627s

$

1. Enter your result for real time in the table under the “OpenMP” column in the row with 1 node used.
2. Open a new file called life.mpi.qsub and enter the following text:

#PBS -q ec

#PBS -o life.mpi.out

#PBS -e life.mpi.err

#PBS -l nodes=1:ppn=4

cd $PBS\_O\_WORKDIR

time mpirun -np 4 -machinefile $PBS\_NODEFILE \

./life.mpi -r 1000 -c 1000 -t 100

1. Submit a job to the scheduler with life.mpi.qsub, output the result (life.mpi.err), and add the real time to the table under the “MPI” column for 1 node used.
2. Change the following bold sections of life.mpi.qsub:

#PBS -q ec

#PBS -o life.mpi.out

#PBS -e life.mpi.err

#PBS -l **nodes=2**:ppn=4

cd $PBS\_O\_WORKDIR

time mpirun **-np 8** -machinefile $PBS\_NODEFILE \

./life.mpi -r 1000 -c 1000 -t 100

1. Repeat step 14 but enter the result in the row with 2 nodes used.
2. 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.
3. 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.
4. Open a new file called life.hybrid.qsub and enter the following text:

#PBS -q ec

#PBS -o life.hybrid.out

#PBS -e life.hybrid.err

#PBS -l nodes=1:ppn=4

export OMP\_NUM\_THREADS=2

cd $PBS\_O\_WORKDIR

time mpirun -np 4 -machinefile $PBS\_NODEFILE \

./life.hybrid -r 1000 -c 1000 -t 100

1. 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 running with a constant 1000 rows, we increase the number of rows 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 100 \* number of cores as the example in this case.

**Walltimes for Weak Scaling, 100 rows per Core, 1000 columns, and 100 Time Steps**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| # of nodes used | Total # of cores | Total # of rows | Serial | OpenMP | MPI | Hybrid |
| 1 | 4 | 400 |  |  |  |  |
| 2 | 8 | 800 |  |  |  |  |
| 3 | 12 | 1,200 |  |  |  |  |
| 4 | 16 | 1,600 |  |  |  |  |
| 5 | 20 | 2,000 |  |  |  |  |
| 6 | 24 | 2,400 |  |  |  |  |
| 7 | 28 | 2,800 |  |  |  |  |
| 8 | 32 | 3,200 |  |  |  |  |
| 9 | 36 | 3,600 |  |  |  |  |
| 10 | 40 | 4,000 |  |  |  |  |
| 11 | 44 | 4,400 |  |  |  |  |
| 12 | 48 | 4,800 |  |  |  |  |

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

#PBS -q ec

#PBS -o life.serial.out

#PBS -e life.serial.err

cd $PBS\_O\_WORKDIR

time ./life.serial -r 400 -c 1000 -t 100

1. Fill in the first rows of the table under the Serial column as you did in Part I.
2. Fill out the table for OpenMP, MPI, and Hybrid, but make sure that for every job you submit the value for -r is equal to the value of -np times 100. The PBS files should look like the following to start:
   1. **life.openmp.qsub**:

#PBS -q ec

#PBS -o life.openmp.out

#PBS -e life.openmp.err

export OMP\_NUM\_THREADS=8

cd $PBS\_O\_WORKDIR

time ./life.openmp -r 400 -c 1000 -t 100

* 1. **life.mpi.qsub**:

#PBS -q ec

#PBS -o life.mpi.out

#PBS -e life.mpi.err

#PBS -l nodes=1:ppn=4

cd $PBS\_O\_WORKDIR

time mpirun -np 4 -machinefile $PBS\_NODEFILE \

./life.mpi -r 400 -c 1000 -t 100

* 1. **life.hybrid.qsub**:

#PBS -q ec

#PBS -o life.hybrid.out

#PBS -e life.hybrid.err

#PBS -l nodes=1:ppn=4

export OMP\_NUM\_THREADS=2

cd $PBS\_O\_WORKDIR

time mpirun -np 4 -machinefile $PBS\_NODEFILE \

./life.hybrid -r 400 -c 1000 -t 100