Day 1 lab instructions

Compiling and running an OpenMP job on Scholar

In the folder *OpenMP* there are two OpenMP programs, omp_hello.c and omp_hello.cpp. You only need to run one of them — they are the C and C++ versions of the same program.

In the following, *italics* indicate things you should enter. **bold $** indicates prompts from the computer. [regular text in brackets] indicates a comment

1. Transfer the omp_hello.c or omp_hello.cpp program to Scholar:

sftp scholar.rcac.purdue.edu
enter your password
$ *put omp_hello.c [or omp_hello.cpp]*
$ *quit*

2. Log in to Scholar and set up the environment to compile OpenMP programs.

$ *ssh scholar.rcac.purdue.edu*
enter your password
$ *module load intel*
$ *icc —qopenmp omp_-std=c99 hello.c -o omp_hello* [or icc -openmp omp_hello.cpp -o omp_hello *for C++]*

3. Use the script omp.sub, that has the following lines, to run your job.

```
#!/bin/sh -l
# FILENAME:  omp.sub

cd $PBS_O_WORKDIR
export OMP_NUM_THREADS=20
./omp_hello
```

4. Submit your OpenMP job.

$ *qsub -l nodes=1:ppn=20,walltime=00:01:00 omp.sub*

5. Examine your output. Your output will be in two files, ending in .onnnnn and .ennnnn , where *nnnnn* is a unique number.

Compiling and running an MPI job on Scholar

In the folder *MPI* there are two MPI programs, mpi_hello.c and mpi_hello.cpp. You only need to run one of them — they are the C and C++ versions of the same program.

In the following, *italics* indicate things you should enter. **bold $** indicates prompts from the computer. [regular text in brackets] indicates a comment

1. Transfer the mpi_hello.c or mpi_hello.cpp program to Scholar:

sftp scholar.rcac.purdue.edu
enter your password
$ *put mpi_hello.c [or mpi_hello.cpp]*
$ *quit*

2. Log in to Scholar and set up the environment to compile MPI programs.

$ *ssh scholar.rcac.purdue.edu*
enter your password
$ *module load intel*
$ *mpiicc -std=c99 mpi_hello.c -lm -o program* [or icc -openmp mpi_hello.cpp lm -o mpi_hello *for C++]*

3. Create a script to run your job. Create a file called mpi.sub that has the following lines:

```
#!/bin/sh -l
# FILENAME:  mpi_hello.sub

cd $PBS_O_WORKDIR

mpiexec -n 40 -machinefile ./nodefile ./mpi_hello
```

Note that 40 is the number of processes that will run your job, i.e., how much possible parallelism there is.

4. Submit your MPI job. Your output will be in two files, ending in .o*nnnnn* and .e*nnnnn* , where *nnnnn* is a unique number.

$ *qsub -l nodes=1:ppn=20,walltime=00:01:00 ./sub*

where *nodes=1* says how many nodes will run your job (there are 20 cores on each node), *ppn=20* says how many processes will run on each node, *walltime=00:01:00* says the job will run for approximately one minute before being killed, and *sub* is the name of the script you wrote in Part 3.