OpenMP Exercises

These exercises will introduce you to using OpenMP for parallel programming. There are four exercises:

To begin, log onto an interactive node of Ranger using your account:

```
ssh -X <user-name>@ranger.tacc.utexas.edu
```

Untar the openmp_lab.tar file (in ~train100) into your directory:

```
tar xvf ~train100/labs/lab_openmp.tar
cd lab_openmp
```

The makefile that comes with these exercises is set up to use the Intel compilers, but the PGI compilers are the default when you log on, so switch compilers:

```
module swap pgi intel
```

Also, some of the exercises require the Intel Math Kernel Library, so add that to your list of modules:

```
module add mkl
```

OMP Hello world

The *Hello world* example is very short, so for convenience we will run it on the interactive node where you're logged in. The other examples will run longer and will involve measuring performance, so they will be done on dedicated nodes through the batch system.

Look at the code in *hello.c* and/or *hello.f90*. This code simply reports OpenMP thread IDs in a parallel region. Compile hello.c or hello.f using the makefiles provided and execute first with 3 threads:

make hello_c *or* **make hello_f90**
setenv OMP_NUM_THREADS 3
./hello_c *or* **./hello_f90**

and then with 2 to 16 threads:

make run_hello_c *or* **make run_hello_f90**

Parallel Region

Look at the code in work.f90. Threads perform *some* work in a subroutine called pwork. The timer returns wall-clock time. Compile work.f90 and run it with one set of threads to verify that it built properly. Running with other numbers of threads will be done in a batch job after all of the executables have been built.

```
make work
setenv OMP_NUM_THREADS 3
./work
```

Dynamic Scheduling

Look at the code in file saxpy.f90. The nested loop performs a simple DAXPY type operation. Parameter *N* determines the size of the problem (N=1024*4 is the default). Compare the performance with static and dynamic scheduling. A more detailed comparison will be done in the batch job.

```
make saxpy
setenv OMP_SCHEDULE static
./saxpy
setenv OMP_SCHEDULE dynamic
./saxpy
```

Hand coding *vs.* libraries

Look at the code in file saxpy2.f90. The nested loop performs a DAXPY operation for each outer loop. The DAXPY routine comes from the Intel MKL library. You should have loaded its module at the beginning of these exercises. Compare the performance with what you saw in the previous exercise with the hand-coded version of DAXPY. Then run a batch job that makes more detailed comparisons on a dedicated node.

```
make saxpy2
setenv OMP_SCHEDULE static
./saxpy2
setenv OMP_SCHEDULE dynamic
./saxpy2
```

Edit the job file to put your account number after the -A flag

```
qsub job
```