LAB : OpenMP

Lab Outline

- OpenMP
  - Hello World
  - Parallel Work Loop
  - Pi (π): convert serial version to multi-threaded
- We will use the login nodes to make short, quick runs (normally all computations must be submitted to the queueing system)
- For this lab, the compile scripts are setup for the Intel compiler; use modules to switch from PGI to Intel

```
> module unload mvapich2
> module swap pgi intel
> module load mvapich2
```

- Log on to Ranger using your train## account:

```
ssh -X train##@ranger.tacc.utexas.edu
```

You will be assigned this number.

- Untar the file openmp_lab.tar file (in ~train00) into your directory:

Ranger : `tar -xvf ~train00/openmp_lab.tar`

- cd down into the openmp_lab directory:

```
cd openmp_lab
```

- Please refer to the next set of slides for instructions on how to build and run the examples on either system. As an example, the plots of library performance are only illustrated for the IBM ESSL library. Similar plots can be obtained for the Intel MKL library.

OMP Hello World

Look at the code in the source files hello.c and hello.f
This code simply reports thread IDs in a parallel region.
Compile hello.c and hello.f for multitasking and execute with
1 to 5 threads. (See also do_c_hello and do_f90_hello scripts.)

- C program: Compile and run

```
make hello_c
setenv OMP_NUM_THREADS 3
./hello_c
```

- F90 program: Compile and run

```
make hello_f90
setenv OMP_NUM_THREADS 3
./hello_f90
```

Do a "**make run_hello_c**" or "**make run_hello_f90**" for an automated execution of 1 to 16 threads.

Parallel Region Example

Look at the code in file work.f90
Threads perform some work in a subroutine called pwork.
The timer returns wall-clock time
Compile work.f for multitasking and execute with
3 and 4 threads.

There can be a wide variation in the runs when the system is busy. Use the **top** command *Ranger* to see the load on the login node.

- Compile and run the work program

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

- Look at the system load

`top` **{hit the "1" key to see the cpus loads; hit the "q" key to quit }**

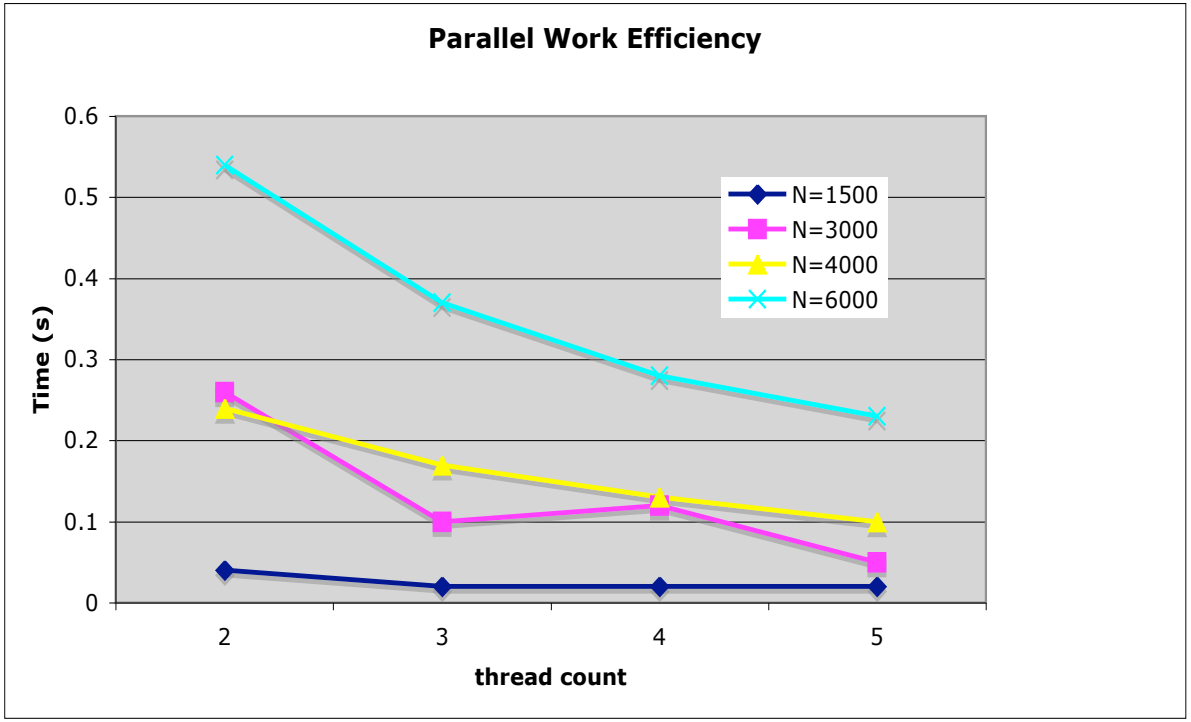
Parallel Region Example

Execute:

```
make run_work
```

to run the executable for 1 through 5 threads.

- Change N in the code and execute "`make run_work`" again.
- Why does the parallel efficiency improve with data size *N*?
- The graph below shows the times on a dedicated (non-busy) system.
  (Example taken from Champion, IBM system)



Calculating π – OpenMP

- In this example, you will parallelize the π code using OMP directives (cd to the pi directory)
- The pi.c/pi.f90 program calculates an integral using the Simpson method to evaluate pi
- Copy the serial version to *pi_new* and parallelize it using OpenMP
  - To run serial version, create a file with the number of iterations to perform (5e7 is a good starting point, so 50,000,000)
  - `./pi < number`
  - Verify that parallel and serial versions get the same answer
  - Compare scaling for various number of threads (by setting OMP_NUM_THREADS)
  - Do you get the same answer with arbitrary thread counts?