HPC Course - January 2020

OPENMP Hands on Exercises

- Hello World [folder:OMPHelloWorld]
- Openmp Schedules [folder:OMPSchedules]
- Data Scoping (private firstprivate..) [folder:OMPDataScoping]
- #### Compute PI [folder: OMPComputePI]
- Fibonacci series computation [folder:OMPFibonacci]
- Matrix Multiplication [folder:OMPMatrixMult]
- Mandelbrot set computation [folder:OMPmandelbrot]

1. Copy Sample Code for OPEMP Lab Session

Before starting the course, the example programs and jobscripts used in this tutorial must be copied to your home directory, so that you can work with your personal copy. All examples are present in the "/home/proj/16/secpraba/2020JanOPENMP" directory. Copy folder and change permissions to read write and execute all files in the folder that you created.

a. Create a folder/directory with your name. Try to make it as unique as possible.

```
$mkdir <yourname>[Number]
```

b. Copy all code for openmp lab sessions into your folder that you just created

```
$scp -r /home/proj/16/secpraba/2020JanOPENMP /<workingdirectory>/<yourname>[Number]
```

#

Exercise 3 - Compute PI

1. First write a serial program that computes PI - based on the methodology discussed in the lecture Alternatively, copy the program below and run the Serial function to compute PI
2. Time the program and record the time for various (STEPS [100K, 500K, 1000K etc.]) use double start = omp_get_wtime(); when the computation starts use double end = omp_get_wtime(); when computation ends double delta = end -start; print compute time in seconds using printf("PI = %.16g computed in %.4g seconds\n", pi, delta);
3. Use Work Sharing Construct #pragma omp for to parallelize the for loop (Write a new function for parallel PI) using 4 threads a) Do NOT add a reduction clause. Try to see race conditions - check the value of PI returned. b) Use a syncronization construct #pragma omp atomic to ensure that there are no race conditions. Compile and Run program and time the computation. Is there any performance improvement in the parallel code? If NO, why? c) Use a reduction clause to reduce "sum", now check the

accuracy and performance of the program. Report findings.

4. Time the parallel program with reduction for 1,2,3,4,5,6,7,8,9,10 ..24 Threads for STEPS {100K, 500K, 1000K}
5. Record the time and accuracy of PI for various threads and step_size.