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 5 - Fibonacci Series

1. Write a serial function to compute Fibonacci(N) using recursion.

While computing fibnocacci(N), fibonacci(1), fib(2), fib(3)....will be computed several times in this recursion method. Enumerate how many times Fib(I) (I = 1 through N-1) is computed. Time the function and print the time taken for computing Fib(N) for values of N = 20, 25, 30, 35

```
long naiveSerialFib(int N) <= Serial function that conputes Fibonacci[N]
```

The above is a naive function, because it uses recursion to repeatedly call fib[N-1] and fib[n-2] Most of the time the fib of a number is computed more than once. The code includes a counter that keeps track of how many times the fibonacci of a given integer is computed. This is stored in the global(shared) array

repeatComputation. repeatComputation[i] tells how many times the fibonacci of "i" was computed, in the process of calculating fibonacci(N)

2. Optimize the naive function to reduce the number of times a certain Fib(I) is computed

Time the function and print the time taken for computing Fib(N) for values of N = 20, 25, 30, 35

```
long optimalSerialFib(int N) <= Serial function that computes Fibonacci[N]
```

This is an optimal function (not the best one, but much better than the naive function). A shared array (long* computedVals) is implemented to store the computed value of Fibonacci[i] for 0<= i <= N Before computing Fibonacci[i] , the function checks if it has already been computed and reuses that value. By running the following program. it can be seen that the naive function has a very poor performance when compared to the optimal serial function.

3. Parallelize the first serial function using openmp tasks

```
 long parallelNaiveFibWithTasks(int N)
The naive serial function is converted to a parallel function using openmp tasks.
 ```
Performance is expected to be worse than the serial code because of too many tasks being
Sometimes this even fails to return.(try N > 40)

4. Using OPENMP tasks parallelise the optimised serial Fibonacci function.
Also store the number of fibonacci computations does per thread to understand how well/p
``` sh
long parallelFibWithTasks(int N)
improved version using an array of computed Fibonacci values
```

Much better than function 3 in terms of performance. But has a lot of overhead for creation of tasks. Interestngly, the function also stores the number of Fibonacci computations done per Thread.

Complete the following tasks 1. Fix the number of threads = number of processors (just let it default) 2. Run the optimal parallel and serial functions for various values of N (25 through 50) 3. Document the serial and parallel compute times. 4. Calculate speed up for parallel code for various N 5. Do you see any patterns in speedup? 6. Can the parallel code be further optimised? Hint:based on my experience, optimised serial code works well for N < 30. Parallel code works well for N>= 30 One could add an "if" clause in the #pragma omp task directive to see if they get an improvement in speedup