HPC: High-Performance Computing
Academic Year: 2022 - 23

Dr. Praveen Kumar Alapati
praveenkumar.alapati@mahindrauniversity.edu.in
Department of Computer Science and Engineering
Ecole Centrale School of Engineering

Assignment 2 (Due Date: February 16, 2023)

Develop a parallel code for the following problems using OpenMP. Report the speedup of your implementations by varying the number of threads from 1 to 16 (i.e., 1, 2, 4, 6, 8, 10, 12, 14, and 16). Use *gettimeofday*() for calculating runtime and consider the average of 5 runs.
Draw appropriate graphs using **gnuplot**.

1. Generate $4x10^7$ numbers using a Uniform Random Number Generator and store the numbers in an array from $A[0]$ to $A[4x10^7 - 1]$. Sort the array using Merge Sort Technique by considering bottom-up approach. Solve the problem for different data types (i.e., int, float, and double).

2. Place N-Queens on the N $\times$ N chessboard in non-attackable positions, $N \in \{10, 12, 14, 16\}$. Record all the solutions.

Note: Top 9 teams will be rewarded with 1 bonus mark.

Submission Guide Lines:

Submission Guide Lines:

- Mail-ID: hpc.mu.2023@gmail.com
- Sub:ROLLNUM_ASSIGN_NUM
- Attach.Name and Type: (ROLLNUM_ASSIGN_NUM).zip
- Write a readme file to understand your solutions.
- Submit source files only.

Learn the art of multi-core and many-core programming

Assignment 1 (Due Date: February 7, 2023)

Develop a parallel code for the following problem using OpenMP. Report the speedup of your implementations by varying the number of threads from 1 to 16 (i.e., 1, 2, 4, 6, 8, 10, 12, 14, and 16). Use *gettimeofday*() for calculating runtime and consider the average of 5 runs. Finally, draw appropriate plots using the GNU plot. For example

- Runtime vs. Matrix Sizes by fixing number of threads
- Runtime vs. Threads by fixing the Matrix Size.

1. **n^{th} Power of a Square Matrix:** Consider a square matrix A and fill the matrix A (vary the order of matrix from 512x512 to 2048x2048, in powers of 2) with random entries ranging from 0 to 1. Assume that the matrix is given in row-major order. If you perform any transformation, that also has to be accounted for in the runtime as well. Consider the following implementations to find the n^{th} Power, vary the value of n from 2 to 16.
   - Ordinary Matrix Multiplication (OMM).
   - Block Matrix Multiplication (BMM) using block sizes: 4,8,16,32,64.
   - Consider the tranpose of A to find n^{th} Power using OMM.
   - Consider the tranpose of A to find n^{th} Power using BMM.

HelpDoc

1. For all subproblems, you have calculate A^2, A^3, A^4, A^5, ... A^{16} by varying the number of threads: 1, 2, 4, 6, 8, 10, 12, 14, and 16.

2. You have to repeat the **step1** for different Matrix sizes (i.e., for 512, 1024, 2048).

3. For BMM, You have to repeat the **step1** and **step2** for different block sizes (i.e., 4, 8, 16, 32, and 64).