Birla Institute of Technology & Science, Pilani
Pilani Campus

II SEMESTER 2019-2020
Assignment-2

Course No.: CS F422 **Course Title:** Parallel Computing
Deadline: 25th Apr 2020 **Maximum Marks:** 52M (13%)

Note:

- Maximum of two students per group.
- Upload code in https://nalanda.bits-pilani.ac.in Name your file idno1_idno2_assignment1.tar .
- Group information to be supplied in group.txt

P1. Write a Pthread-based threaded program for solving a 15-puzzle. The program takes an initial position from a text file "input.txt" and keeps an open list of outstanding positions. This list is sorted on the "goodness" measure of the boards. A simple goodness measure is the Manhattan distance (i.e., the sum of x -displacement and y-displacement of every tile from where it needs to be). This open list is a work queue implemented as a heap. Each thread extracts work (a board) from the work queue, expands it to all possible successors, and inserts the successors into the work queue if it has not already been encountered. Use a hash table to keep track of entries that have been previously encountered.

(a) Plot the speedup of your program with the number of threads.
(b) Modify the above program so that you now have multiple open lists (say k). Now each thread picks a random open list and tries to pick a board from the random list and expands and inserts it back into another, randomly selected list. Plot the speedup of your program with the number of threads. Compare your performance with the previous case. Make sure you use your locks and trylocks carefully to minimize serialization overheads.

Deliverables:

- Design Document (.pdf) containing design steps of parallel algorithm and implementation aspects
- Source code: puzzle_a.c, puzzle_b.c
- Input files
- Plots and data
- Screen shots/ Video recording of the demo
- Group.txt

[32M]

P2. Implement and test OpenMP program for computing a matrix-matrix product. Use the OMP_NUM_THREADS environment variable to control the number of threads and plot the performance with varying numbers of threads separately for the following three cases

(a) only the outermost loop is parallelized;
(b) the outer two loops are parallelized; and
(c) all three loops are parallelized.
(d) What is the observed result from these three cases?

Deliverables:

- Design Document (.pdf) containing design steps of parallel algorithm and implementation aspects
- Source code: matrixmul_case_a.c, matrixmul_case_b.c, matrixmul_case_c.c
- Input files
- Plots and data
- Screen shots/ Video recording of the demo
- Group.txt

[20M]

--&--