Parallel and Distributed Computing: Homework 4

Due April 29, 2020, in zip format on Canvas.

Consider the following three-level nested loop program to multiply two matrices A and B on a single processor system:

```
For i = 0, n − 1, do:
    For j = 0, n − 1, do:
        For k = 0, n − 1, do:
            c_{i,j} = c_{i,j} + a_{ik} × b_{k,j}
        endfor
    endfor
endfor
```

This is the exact same nested loop program as discussed in class and available in the class' slides. We discussed three different ways of performing this computation on a n-processor ring when unfolding the outermost loop with the index i.

It is well known that changing the order of the loops is a program transformation that does not alter the final result (an invariant transformation.) The three nested loops can appear in one out of 6 permuted possibilities, namely: (i, j, k), (i, k, j), (j, i, k), (j, k, i), (k, i, j), (k, j, i).

1. Assuming execution of the above program in a single processor computer, characterize the manner in which the computation progresses for each one of the 6 possible nested loop cases: provide a data to memory mapping, and show how the main arithmetic expression progresses, i.e. how the inner products are computed in time, as the indexes advance with the innermost loop being the one that advances fastest.

2. In a manner akin to the one used in class (see slides), for each one of the nested loop cases, discuss the mapping of the computation on a n-processor ring if, for each of the 6 cases, the outermost loop is used to unfold and parallelize the computation.

3. Consider now the nested loop pseudocode as given above and consider parallelizing (unfolding) the outermost loop on index i. Write a C-program that uses multithreading to parallelize the matrix multiplication computation where the number of threads is 2, 4, 8, and 16. For this exercise, the value of n is assumed to be ≥ 64 and it, as well as the number of threads, are to be given as passed parameters with n ranging from 64 to 1024 in powers of two. Each thread is to compute a row of the resulting product matrix $C = A \times B$.

   It is suggested that you write this multithreaded program as if it were to be incorporated in a library and called with passed parameters as follows:

   - A pointer to the first element of matrix A, assuming that matrices are stored in main memory in row major form and that A is stored first, followed by B, followed by $C = A \times B$.

- The number of threads with value $2, 4, 8, 16$.
- The size of the matrix being a power of two ≥ 64.

You may generate matrices A and B with random value entries.

Discuss the gain in performance as a function of the number of threads generated to run concurrently. It is strongly recommended to use a 4-core system as a minimum.

Submit a zip file named `x-y.zip`, where `x` and `y` are your UCInetIDs (the username used to login in Canvas), containing 2 files:

- The well commented C source code called `mat-mult.c`.
- A digitally produced report of a maximum of 4 pages pages called `report.pdf`.