Parallel and Distributed Computing: Homework 4

Objectives: Parallelization of nested loops can be achieved by "loop-unfolding," namely replicating the body of a loop in all computational resources labeled with the index of the loop. Matrix multiplication can be obtained using a three level nested loop program. This assignment illustrates the loop-unfolding technique by characterizing the progression of the computation when unfolding each one of the loops in all possible orderings of the loops.

Consider the following three-level nested loop algorithm 1 to multiply two square matrices A and B of size n (meaning they have $n \times n$ elements), on a single processor system:

Algorithm 1: Matrix multiplication

```
   Input: A;                    // Input Matrix
   Input: B;                    // Input Matrix
   Input: C;                   // Result Matrix
1  Function Matrix_Mult(A, B, C):
2  |  for 0 ≤ i ≤ n − 1 do
3  |  |  for 0 ≤ j ≤ n − 1 do
4  |  |  |  for 0 ≤ k ≤ n − 1 do
5  |  |  |  |  C_{i,j} ← C_{i,j} + A_{i,k} · B_{k,j}
6  |  |  |  end
7  |  |  end
8  |  end
9  return
```

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).

Assignment

Computation Progress: 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. Show how the main arithmetic expression progresses, accumulating its value as the indices advance, with the innermost loop being the one that advances fastest.

N-processors Ring: 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.

Multi-thread unfolding: Consider now the nested loop pseudocode as given in Alg. 1 and its parallelzation (unfolding) on the outermost loop with index i.

Utilizing the provided template code, complete the `mat_multiply()` function that uses multiple threads to parallelize the matrix multiplication computation.

```
void mat_multiply(Mat *A, Mat *B, Mat *C, \
                  unsigned int threads);
```

The `multiply` function receives four parameters:

- **The input matrices `A` and `B`:** These are square matrices of size n, stored in row major form. For this exercise, consider n restricted to $\{64, 128, 256, 512, 1024, 2048\}$.

- **The output matrix `C`:** This is the matrix where the results will be stored. Given that `A` and `B` are square of size n, then `C` is also square of size n.

- **The number of threads `threads`:** This is the number of threads the multiplication task will be divided on. For this exercise, consider this value restricted to $\{2, 4, 8, 16\}$. Each thread is to compute a row of the resulting product matrix $C = A \times B$.

To understand the use of libraries in C, it is recommended to read: `https://computer.howstuffworks.com/c15.htm`. It is strongly recommended to use a 4-core system as a minimum.

As for debugging purposes, you may change the content of `main.c`, `util.h`, and `util.c`. However, your implemented solution must not depend on those changes, and on submission of this homework, your implementation must strictly use the function signatures provided in `util.h`. Inside of `multiply.c` you can create as many static functions and data structures as needed.

Report

You shall write a report of a maximum of 4 pages consisting in three parts:

1. The characterization of the computation progress in a single processor computer, for each of the six mentioned permutations. Add figures as necessary to support your explanation.

2. Discussion of the mapping of the problem to a n-processors Ring. Add figures as necessary to support your explanation.

3. Discussion of the gain in performance of the multi-thread implementation as a function of the number of threads concurrently used. Add plots and figures as necessary to explain your results.

Submission

Submit **one** zip file named `hw4.zip`, containing **exactly 3 files**. Make sure the zip file does **not** include sub-directories (as Mac's default compression tool does) or extra files:

1. `multiply.c`: The well commented C source code of your multi-threaded implementation of the matrix multiplication. **Do not submit any other source code file**.

2. `report.pdf`: The digitally produced report of a maximum of 4 pages.

3. `team.txt`: Text file with two lines containing the UCINetID and name of each student in your team. Each line will have this format:
`<UCINetID>, <firstName>, <lastName>`