Multi-core Programming
Assignment 2

Ali Gholami

Department of Computer Engineering & Information Technology
Amirkabir University of Technology

https://aligholamee.github.io
aligholami7596@gmail.com

Abstract

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, its also widely used in machine learning, which will be the main focus of this article.

Keywords. *Heterogeneous Programming, OpenMP, C Programming, C++ Programming, Parallelization, Multi-thread Programming.*

1 Matrix Multiplication

1.1 What's the goal?

In this assignment, we'll be parallelizing the matrix multiplication using *OpenMP*. The goal is to speed up the matrix multiplication by implementing the parallelization in two axis (*1D* & *2D*). Below the serial code for the matrix multiplication. Sources for this assignment is available in the repository merged with this report.

```
void multiply(DataSet dataSet){
        int i, j, k, sum;
        for(i = 0; i < dataSet.n; i++){
                for(j = 0; j < dataSet.p; j++){
                        sum = 0;
                        for(k = 0; k < dataSet.m; k++){
                                sum += dataSet.A[i * dataSet.m + k] * dataSet.B[k * dataSet.p + j];
                        }
                        dataSet.C[i * dataSet.p + j] = sum;
                }
        }
}
```

1.2 1D Parallelization

The following figures are provided from the problem description by *Dr. Ahmad Siavashi*. Each of the highlighted areas show a job for a thread. Figure 1.1 shows how the multiplication is done by each thread.
Assuming each *integer* as 4 bytes, we'll be filling the table 1.1 using the average time computed after

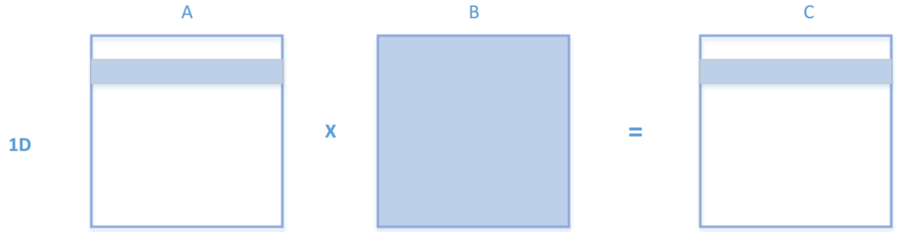

Figure 1.1: Matrix Multiplication Parallelization on Horizontal Axis.

Total Size of Each Matrix					
Num of Threads	1 MB	10 MB	100 MB	1 GB	Speedup
1	AF	AFG	004	004	004
2	AF	AFG	004	004	004
4	AF	AFG	004	004	004
8	AF	AFG	004	004	004

Table 1.1: Results of 1-Dimensional Parallelization.

6 times of running the program.

References

[1] Prashant Gupta, *Cross-Validation in Machine Learning*. Towards Data Science, Jun 5, 2017.