**Assignment2 Report (CSE436, Summer 2016)**

**Kazumi Malhan, 06/08/2016**

1. Backgrounds and Motivation

Programmers are always looking into new ways to improve the performance of algorithms. As hardware reaching its limit to improve serial code execution time, parallel programming is increasingly used. However, executing code in parallel causes overhead due to idling, data communication, and more. In this assignment 2, sum and matrix multiplication algorithms are implemented in both serial and parallel. The performance (execution time), speedup, efficiency, overhead, and cost are compared for both algorithms to understand the benefit and side effect of parallel execution, and determine the number of threads and scheduling policy to achieve better performance.

2. Function Implementation Description

**sum.c parallel function**

Before entering into parallel region, total amount of numbers are divided by number of tasks to determine left over. Left over identified which thread needs to perform extra work to sum all numbers. Inside parallel region, each thread obtains its thread id (tid), and determines the start and end of loop by dividing total number with number of threads (tasks). In the first for loop, all numbers are added to temporary local variable to increase memory access. Next, only threads whose id is less than left over add left over numbers. Finally, resulting partial result is copied to shared array to carry result out of parallel region. At the end, all elements of array are added to the result.

**sum.c parallel for function**

This function is mostly same as serial implementation of code except adding parallel for directives to enable penalization. After declaring local variables, parallel region is created with number of tasks. Just before the for loop, for directive is added with “schedule (runtime)”. This enables various scheduling policy implementation of changing OMP\_SCHEDULE without re-compilation. Partially calculated results by each threads are added together by “reduction (+:result)”. Also, “nowait” is added to for directive to remove barrier as there is one at end of parallel region.

**mm.c parallel row function**

**mm.c parallel col function**

**mm.c parallel rowcol function**

**mm.c parallel for row function**

**mm.c parallel for col function**

**mm.c parallel for rowcol function**

3. Performance Report

4. Conclusion

References