CS 301
High-Performance Computing

Lab 3

Member 1 (Krunal Lukhi - 201901449)
Member 2 (Dhwanil Shah - 201901450)

April 5, 2022

Contents

1 Introduction

In this lab we would be analyze various parallelization technique of matrix multipilcation problem by paralllelizing the three loops separately using $pragma omp for$ and $pragma omp schdule(dynamic)$. Then, we looked at at load balancing and finally used vtune memory access command for profiling.

Load balancing: It refers to the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient. Load balancing can optimize the response time and avoid unevenly overloading some compute nodes while other compute nodes are left idle.

2 Hardware Details

- CPU - 4
- Socket - 1
- Cores per Socket - 4
- Size of L1d cache - 32K
- Size of L1i cache - 32K
- Size of L2 cache - 256K
- Size of L3 cache - 6144K

3 Part 2

3.1 Naive Parallelization

3.1.1 Outermost loop parallelization

- **Run-time analysis:**

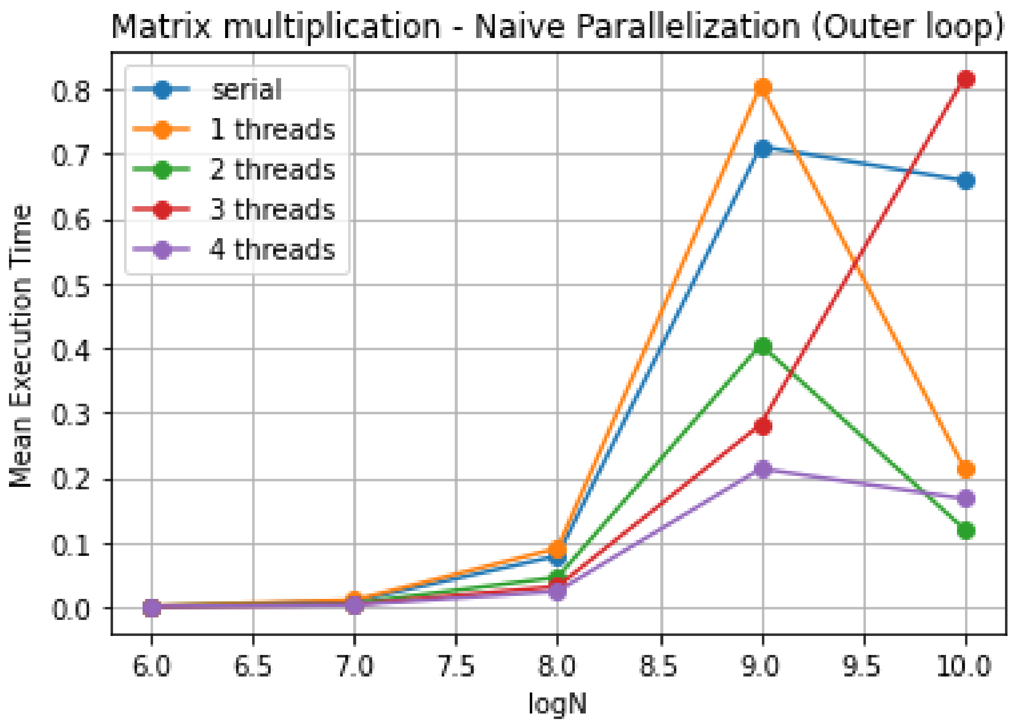

Figure 1: Problem size vs Mean execution time

- **Memory-access analysis:**

  Memory Bound: 36.0% pipeline slots
  L1 Bound: 13.7% of Clockticks
  L2 Bound: 0 % of Clokticks
  L3 Bound: 6.6 % of Clockticks
  DRAM bound: 1.3% of Clockticks
  Store Bound: 27.1% of Clockticks
  LLC miss Count: 1400042
  Average latency: 9 cycles

3.1.2 Middle loop parallelization

- **Run-time analysis:**

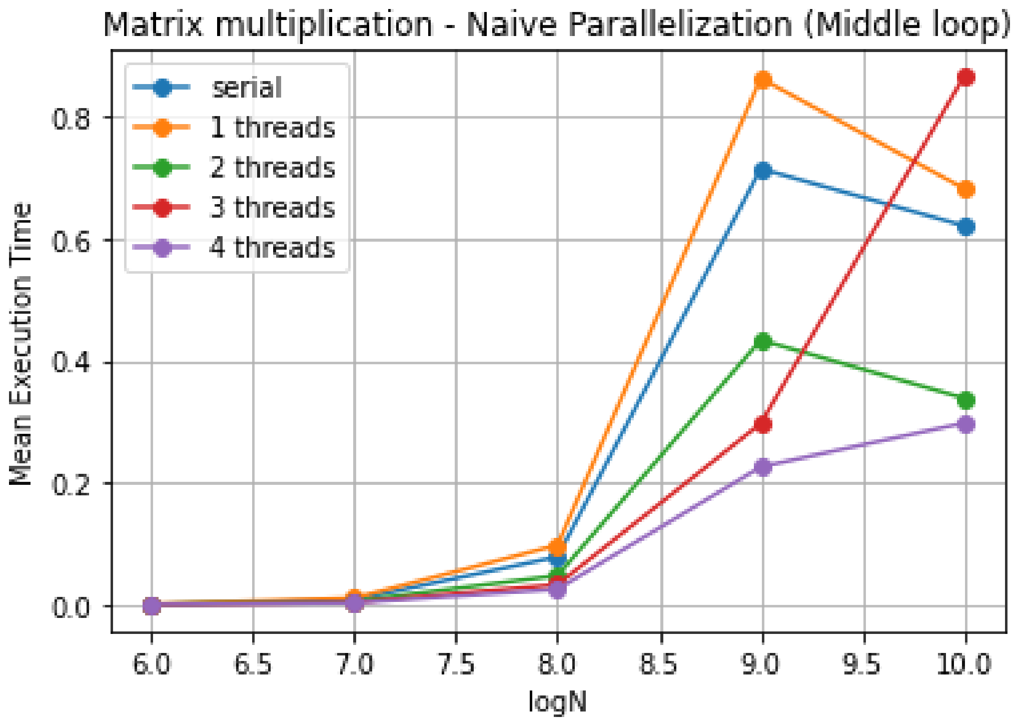

Figure 2: Problem size vs Mean execution time

- **Memory-access analysis:**

  Memory Bound: 37.3% pipeline slots
  L1 Bound: 12.9% of Clockticks
  L2 Bound: 0 % of Clokticks
  L3 Bound: 6.2 % of Clockticks
  DRAM bound: 1.2% of Clockticks
  Store Bound: 32.9% of Clockticks
  LLC miss Count: 1400042
  Average latency: 9 cycles

3.1.3 Inner loop parallelization

- **Run-time analysis:**

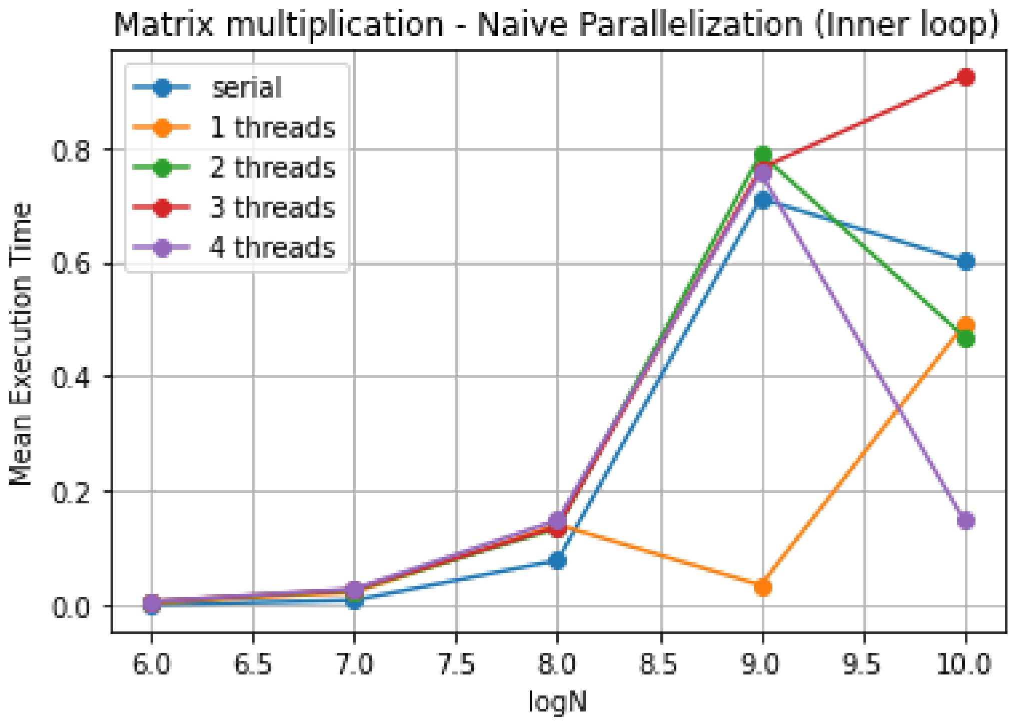

Figure 3: Problem size vs Mean execution time

- **Memory-access analysis:**

  Memory Bound: 35.9% pipeline slots
  L1 Bound: 13.6% of Clockticks
  L2 Bound: 0 % of Cloktick
  L3 Bound: 6.5 % of Clockticks
  DRAM bound: 1.5% of Clockticks
  Store Bound: 27.3% of Clockticks
  LLC miss Count: 2800084
  Average latency: 9 cycles

From the graphs we can see that the parallelization of the inner most loop is not good because of the fact that the outer two loops are running serially.

The memory access time for the innermost loop is the most, because in this approach the parallelization occurs of the columns and in columns the cells are not consecutive and hence the cache miss would be the highest. For the outermost loop, it would be the least because of least cache miss, since the entire row would be present in the cache line. For the middle loop, it lies in between the time of inner and outer loop.

3.2 Parallelization with scheduling

3.2.1 Outermost loop parallelization

- **Run-time analysis:**

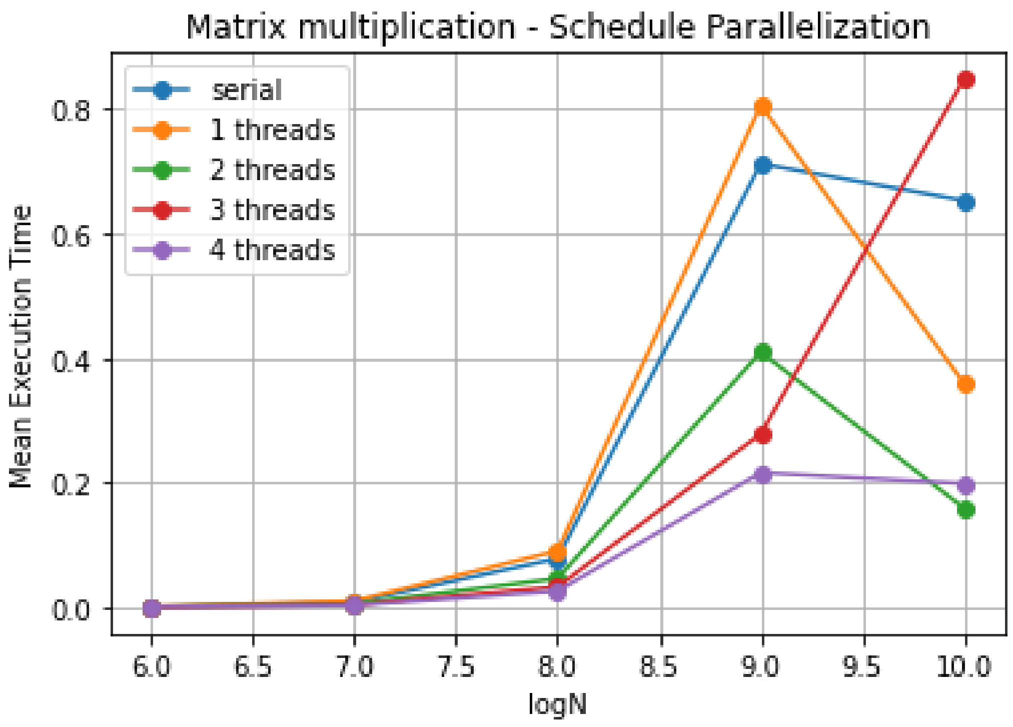

Figure 4: Problem size vs Mean execution time

- **Load-balancing analysis:**

Thread ID	Execution time
0	2.18
1	2.15
2	2.17
3	2.16

Table 1: Table to test captions and labels.

3.2.2 Middle loop parallelization

- **Run-time analysis:**

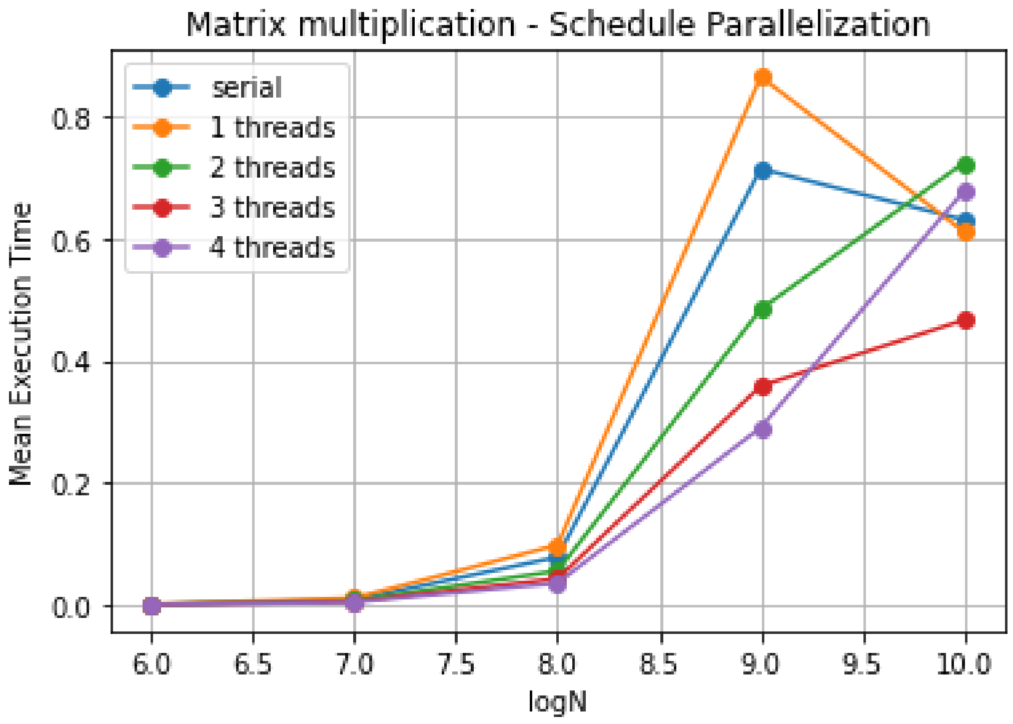

Figure 5: Problem size vs Mean execution time

- **Memory-access analysis:**
- **Load-balancing analysis:**

Thread ID	Execution time
0	2.20
1	2.17
2	2.12
3	2.14

Table 2: Table to test captions and labels.

3.2.3 Inner loop parallelization

- **Run-time analysis:**

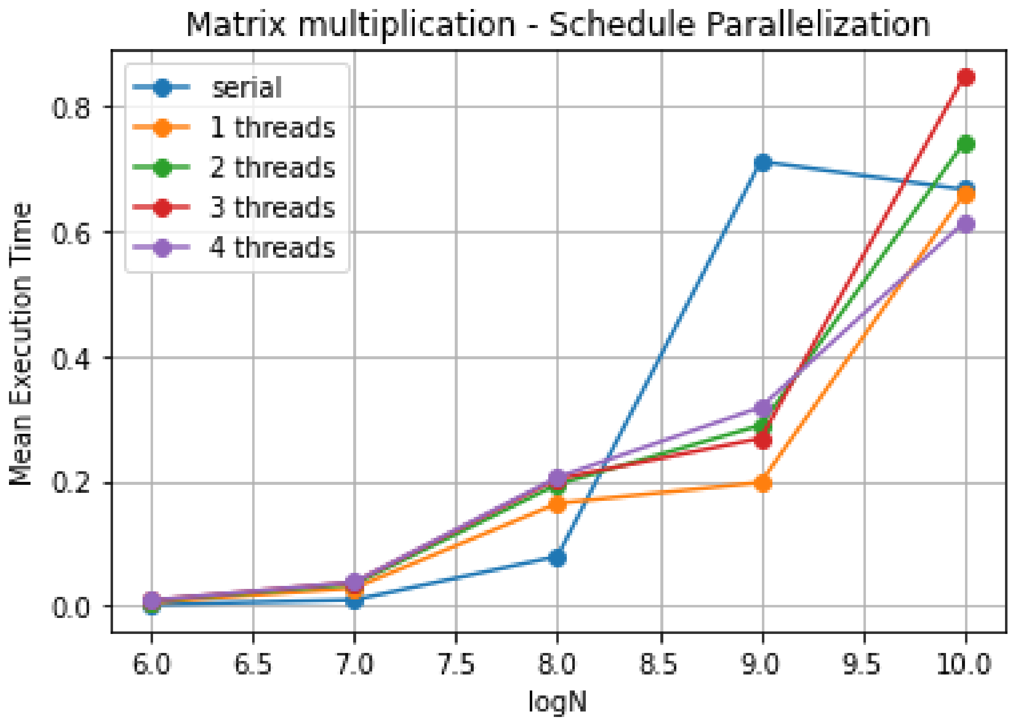

Figure 6: Problem size vs Mean execution time

- **Load-balancing analysis:**

Thread ID	Execution time
0	2.19
1	2.25
2	2.13
3	2.16

Table 3: Table to test captions and labels.

After performing dynamic scheduling, we can see that the execution time for all threads fall in similar range, hence leads to load balancing and better efficiency.