Assignment 3 - Parallel Strassen's Matrix Multiplication

Ghazal Rafiei

April 4, 2022

1 Introduction

The objective in this assignment is to parallelize Strassen's matrix multiplication. In the rest of this document, I will discuss the platform specifications, implementation approach, experiment method and the results.

2 Platform

The program is written in C++ language using OpenMP library. Furthermore, here is the specification of the system.

```
Linux zenbookux434flcux433flc 5.15.25-1-MANJARO 1 SMP PREEMPT x86_64 GNU/Linux
CPU(s):  8
Vendor ID: GenuineIntel
Model name:  Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz
MemTotal:  16179008 kB
```

3 Implementation

As you know, Strassen algorithm divides matrices into four sub-matrices and uses some arbitrary operations to simplify the matrices multiplications. Those arbitrary operations consist of addition, subtraction and multiplication which is calculated recursively by Strassen's algorithm or by ordinary matrix multiplication. In this paper, I will discuss to which depth should we run the algorithm recursively and when we should use the ordinary multiplication as the ad-hoc state. It is worth nothing that the program is implemented only for square matrices with the dimension of powers of 2.

3.1 Parallelization

To parallelize the algorithm, I used `omp task` to run those arbitrary operations at the same time. Furthermore, addition, subtraction and the ordinary multiplication are executed parallel using `omp parallel for`.

4 Experiment

Multiplication of two random generated 1024 * 1024 matrices are calculated using the serial and the parallel algorithms. In each run, algorithms are executed 10 times and the average is reported. Sizes of 32, 64, 128 and 256 are tested for the ad-hoc. Moreover, all the 16 threads are utilized in this experiment.

4.1 Output sample

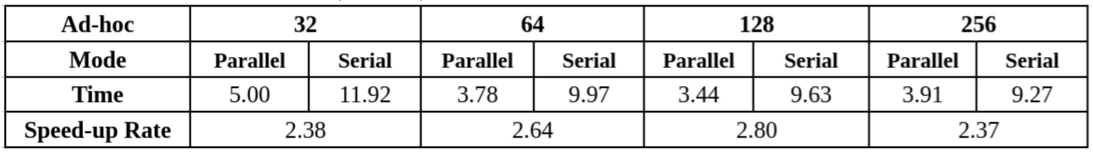
```
mode:  parallel
average time:  3.78667
threads:  16
```

```
matrix size:  1024 * 1024
ad-hoc size:  64
times of run:  10
```

5 Results

In the following table, we can see the results of explained experiment in the last section.

Table 5.1 - Times(second) for random 1024*1024 matrix on 16 threads.



Ad-hoc	**32**		**64**		**128**		**256**	
Mode	**Parallel**	**Serial**	**Parallel**	**Serial**	**Parallel**	**Serial**	**Parallel**	**Serial**
Time	5.00	11.92	3.78	9.97	3.44	9.63	3.91	9.27
Speed-up Rate	2.38		2.64		2.80		2.37	

As you can see in the table 5.1, the best ad-hoc is 128 which speeds-up the algorithm 2.80 times.