Project #1

Multicore Computing

Problem 2

Date	May 10, 2020
Instructor	Bongsoo Sohn
Std. Name	Junhyuck Woo
Std. ID	20145337

INDEX

ENVIRONMENT

Hardware

- MacBook Pro (15-inch, 2017)
- Processor: 2.8 GHz Quad-Core Intel Core i7
- Memory: 16GB 2133 MHz LPDDR3

Operating System

- macOS Catalina, ver: 10.15.4

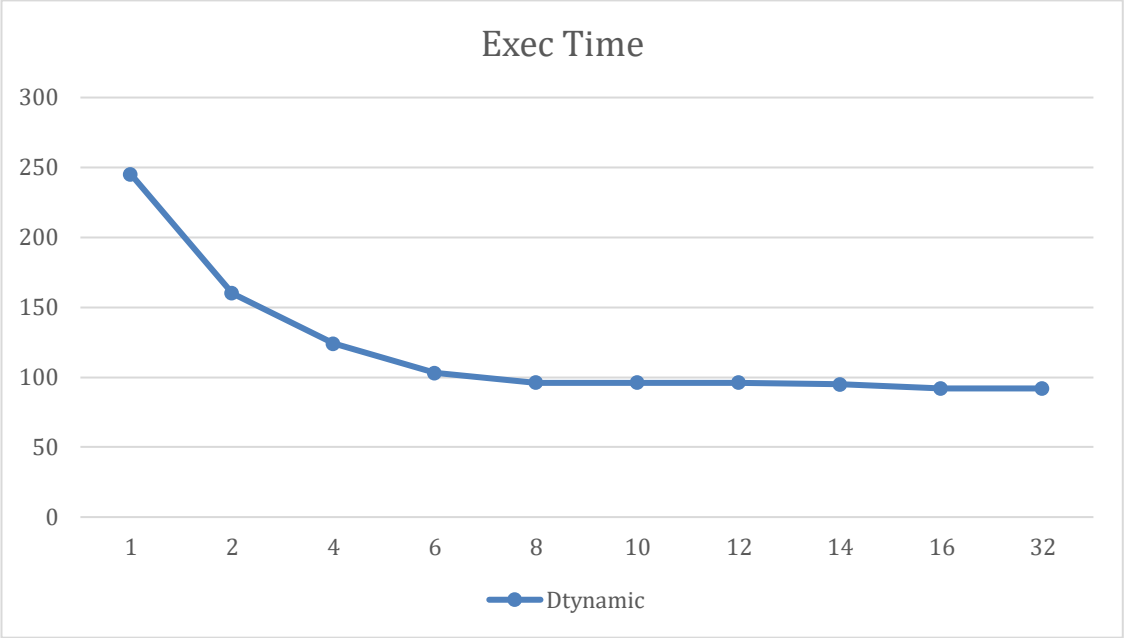
IDE (Integrated Development Environment)

- IntelliJ IDEA 2019.3.4 (Ultimate Edition)
- Java version “11.0.6”

Testing Environment

- iTerm2
- Build 3.3.9
- openjdk 14.0.1 2020-04-14
  OpenJDK Runtime Environment (build 14.0.1+7)
  OpenJDK 64-Bit Server VM (build 14.0.1+7, mixed mode, sharing)

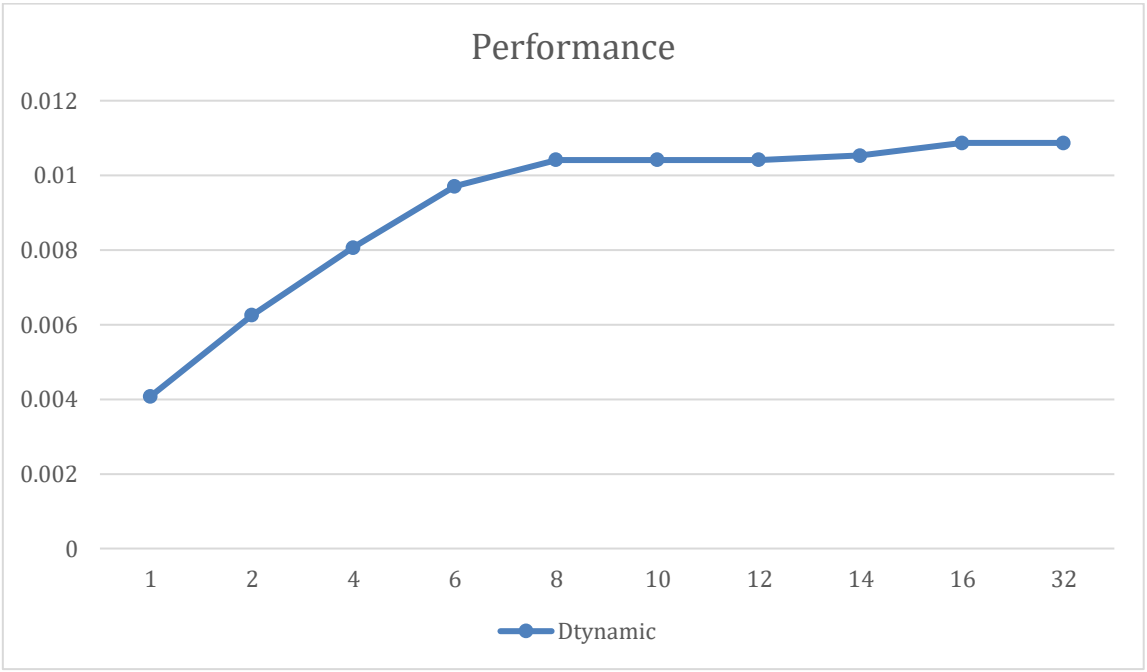
TABLE & GRAPH



▲ Fig. 1. Exec Time of multi-thread programming with dynamic load balancing approach

	1	2	4	6	8	10	12	14	16	32
Exec Time	245	160	124	103	96	96	96	95	92	92

▲ TABLE 1



▲ Fig. 2. Performance of multi-thread programming with dynamic load balancing approach

	1	2	4	6	8	10	12	14	16	32
Performance (1/ Exec Time)	0.004 08163	0.006 25	0.008 06452	0.009 70874	0.010 41667	0.010 41667	0.010 41667	0.010 52632	0.010 86957	0.010 86957

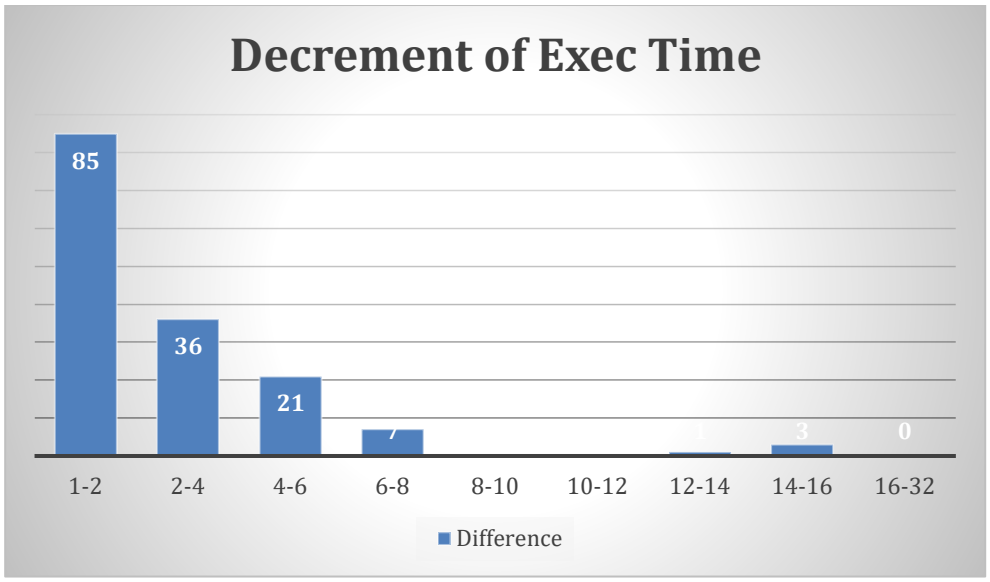
▲ TABLE 2

EXPLANATION ON RESULT

In this problem, the given work is calculating matrix multiplication.

1. Exec Time

The Fig. 1 shows the decrement tendency. To see the detail, I make an additional figure.

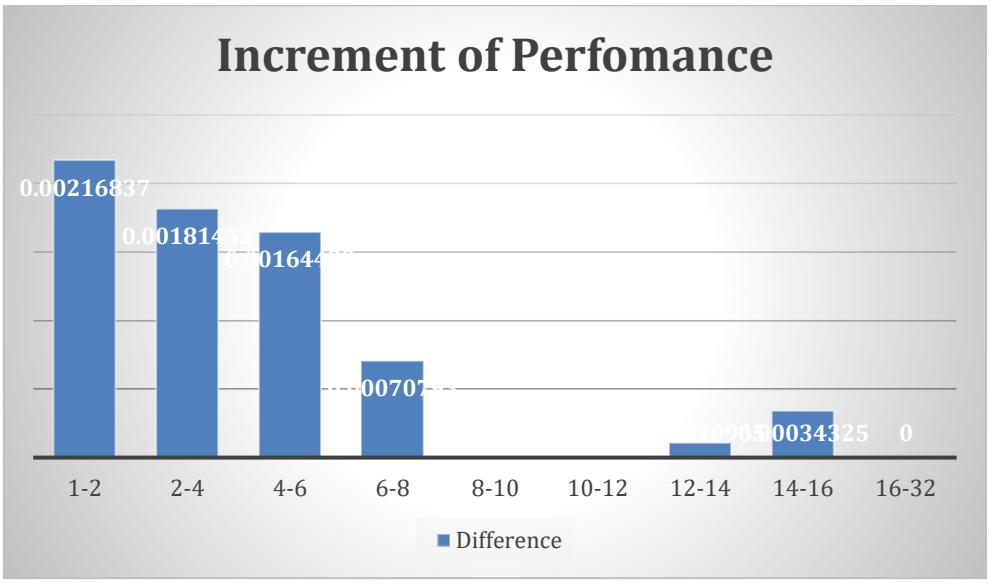


▲ Fig. 3. Decrement of execution time in dynamic load balancing approach

The Fig. 3 shows how the execution time has decreased as the thread increases at the dynamic load balancing approach. The significant results appear to be when increasing from 1 to 6. For 6–32, the execution time is almost same, so there is no significant achievement. I think it's because each thread is assigned too little work.

2. Performance

The Fig. 2 shows the increment tendency. To see the detail, I make an additional figure.



▲ Fig. 4. Increment of performance in dynamic load balancing approach

The performance tends to be similar to execution time. Because performance evaluation criterion is 1 / (exec time). It means that short execution time is the high performance.

SOURCE CODE

```
// Writer: Junhyuck Woo
// Lecture: Multicore Computing
// Organization: Chung-Ang University
// Deadline: May 10, 2020
// Project #1
//  - problem 2

import java.util.*;
import java.lang.*;

// command-line execution example) java MatmultD 6 < mat500.txt
// 6 means the number of threads to use
// < mat500.txt means the file that contains two matrices is given as standard input
//
// In eclipse, set the argument value and file input by using the menu [Run]->[Run Configurations]->{[Arguments], [Common->Input File]}.

// Original JAVA source code: http://stackoverflow.com/questions/21547462/how-to-multiply-2-dimensional-arrays-matrix-multiplication
public class MatmultD_dynamic
{

    public static void main(String [] args)
    {
        int thread_no=0;
        if (args.length==1) thread_no = Integer.valueOf(args[0]);
        else thread_no = 1;

        // Create class for calculation
        matrixOperator mo = new matrixOperator(thread_no);

        // Run the operation
        mo.run();

        // Dealocate the memory
        mo = null;
    }
}

class matrixOperator {
    // Variable
    private int matrix_sum = 0;
    private int thread_num = 0;
    private int[] work = new int[1];
    private int[][] ans;
    private static Scanner sc = new Scanner(System.in);

    // Constructor
    public matrixOperator(int thread_no) {
        thread_num = thread_no;
        work[0] = 0;
    }

    // Read Matrix method
    public int[][] readMatrix() {
        int rows = sc.nextInt();
        int cols = sc.nextInt();
        int[][] result = new int[rows][cols];
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                result[i][j] = sc.nextInt();
            }
        }
        return result;
    }

    // Runner
    public void run() {
        // Variable
        int[][] a=readMatrix();
        int[][] b=readMatrix();
        Matirx[] m = new Matirx[thread_num];
        ans = new int[a.length][b[0].length];

        // Set timer
        long startTime = System.currentTimeMillis();

        // Run
        for (int i=0; i<thread_num; i++) {
            m[i] = new Matirx(work, a, b, ans);
            m[i].start();
        }

        // Wait the work is done
        for (int i=0; i<thread_num; i++) {
            try {
                m[i].join();
                matrix_sum += m[i].getResult();
            }
            catch (InterruptedException e) {}
        }

        // Finish timer
        long endTime = System.currentTimeMillis();

        // Visualize the execution time of each thread
        System.out.println("1. Execution time of each thread");
        for (int i=0; i<thread_num; i++) {
            System.out.println(m[i].getName() + " : " + m[i].getRuntime() +"ms");
        }
        // Visualize the total execution time
        System.out.println("\n2. Total Execution Time: " + (endTime-startTime) +"ms\n");

        // Visualize the sum of elements
        System.out.println("3. Matrix Sum = " + matrix_sum + '\n');

        // Dealocate the memory
        a = null;
        b = null;
        m = null;
        ans = null;
    }
}

class Matirx extends Thread {
    // Variable
    private int length;
    int result = 0;
    private int[] work;
    private int[][] mat_a;
    private int[][] mat_b;
    private int[][] ans;
    private long runtime = 0;
    private String ID;

    // Constructor
    public Matirx(int[] work, int[][] a, int[][] b, int[][] ans) {
        this.work = work;
        this.ans = ans;
        mat_a = a;
        mat_b = b;
        ID = getName();
    }

    public int multMatrix(int i, int j) {
        // Variable
        int n = mat_a[0].length;
        int buf = 0;

        for(int k = 0; k < n; k++){
            buf += mat_a[i][k] * mat_b[k][j];
        }
        synchronized (ans){
            ans[i][j] = buf;
        }
        return buf;
    }

    // Getter - Work
    public int getWork() {
        synchronized(this.work) {
            this.work[0]++;
            return this.work[0]-1;
        }
    }

    // Getter - Result
    public int getResult(){
        return result;
    }

    // Getter - Runtime
    public long getRuntime(){
        return runtime;
    }

    // Getter - ID
    public String getID() {
        return ID;
    }

    // Runner
    public void run(){
        // Variable
        length = ans.length;
        int i_max = mat_a.length;
        int j_max = mat_b[0].length;

        // Set timer
        long startTime = System.currentTimeMillis();
        while(true) {
            int work = getWork();
            final int i = work/length;
            final int j = work%length;

            if((i>=i_max) || (j>=j_max)){
                break;
            }
            result += multMatrix(i, j);
        }

        // Finish timer
        long endTime = System.currentTimeMillis();
        runtime = endTime - startTime;

        // Dealocate the memory
        work = null;
        mat_a = null;
        mat_b = null;
        ans = null;
    }
}
```

OUTPUT

- Thread #1

```
 junhyuckwoo  ~/Documents/CAU/test  java MatmultD_dynamic 1 < mat500.txt
1. Execution time of each thread
Thread-0 : 244ms

2. Total Execution Time: 245ms

3. Matrix Sum = 125231132
```

- Thread #2

```
 junhyuckwoo  ~/Documents/CAU/test  java MatmultD_dynamic 2 < mat500.txt
1. Execution time of each thread
Thread-0 : 160ms
Thread-1 : 160ms

2. Total Execution Time: 160ms

3. Matrix Sum = 125231132
```

- Thread #4

```
 junhyuckwoo  ~/Documents/CAU/test  java MatmultD_dynamic 4 < mat500.txt
1. Execution time of each thread
Thread-0 : 123ms
Thread-1 : 123ms
Thread-2 : 123ms
Thread-3 : 123ms

2. Total Execution Time: 124ms

3. Matrix Sum = 125231132
```

- Thread #6

```
 junhyuckwoo  ~/Documents/CAU/test  java MatmultD_dynamic 6 < mat500.txt
1. Execution time of each thread
Thread-0 : 103ms
Thread-1 : 102ms
Thread-2 : 102ms
Thread-3 : 102ms
Thread-4 : 102ms
Thread-5 : 102ms

2. Total Execution Time: 103ms

3. Matrix Sum = 125231132
```

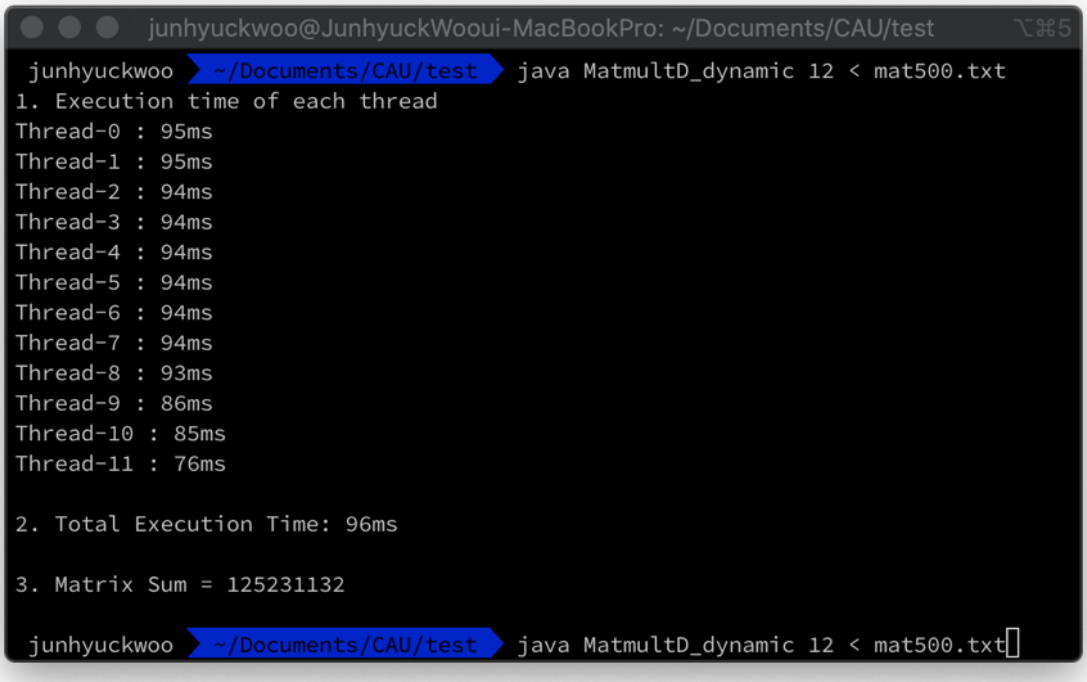
- Thread #8

```
 junhyuckwoo  ~/Documents/CAU/test  java MatmultD_dynamic 8 < mat500.txt
1. Execution time of each thread
Thread-0 : 94ms
Thread-1 : 94ms
Thread-2 : 94ms
Thread-3 : 94ms
Thread-4 : 94ms
Thread-5 : 94ms
Thread-6 : 94ms
Thread-7 : 94ms

2. Total Execution Time: 96ms

3. Matrix Sum = 125231132
```

- Thread #10

```
junhyuckwoo@JunhyuckWooui-MacBookPro: ~/Documents/CAU/test  ⌥⌘5
3. Matrix Sum = 125231132

 junhyuckwoo  ~/Documents/CAU/test  java MatmultD_dynamic 10 < mat500.txt
1. Execution time of each thread
Thread-0 : 95ms
Thread-1 : 95ms
Thread-2 : 95ms
Thread-3 : 95ms
Thread-4 : 95ms
Thread-5 : 95ms
Thread-6 : 95ms
Thread-7 : 95ms
Thread-8 : 94ms
Thread-9 : 82ms

2. Total Execution Time: 96ms

3. Matrix Sum = 125231132

 junhyuckwoo  ~/Documents/CAU/test
```

- Thread #12


junhyuckwoo@JunhyuckWooui-MacBookPro: ~/Documents/CAU/test
junhyuckwoo ~/Documents/CAU/test java MatmultD_dynamic 12 < mat500.txt
1. Execution time of each thread
Thread-0 : 95ms
Thread-1 : 95ms
Thread-2 : 94ms
Thread-3 : 94ms
Thread-4 : 94ms
Thread-5 : 94ms
Thread-6 : 94ms
Thread-7 : 94ms
Thread-8 : 93ms
Thread-9 : 86ms
Thread-10 : 85ms
Thread-11 : 76ms
2. Total Execution Time: 96ms
3. Matrix Sum = 125231132
junhyuckwoo ~/Documents/CAU/test java MatmultD_dynamic 12 < mat500.txt

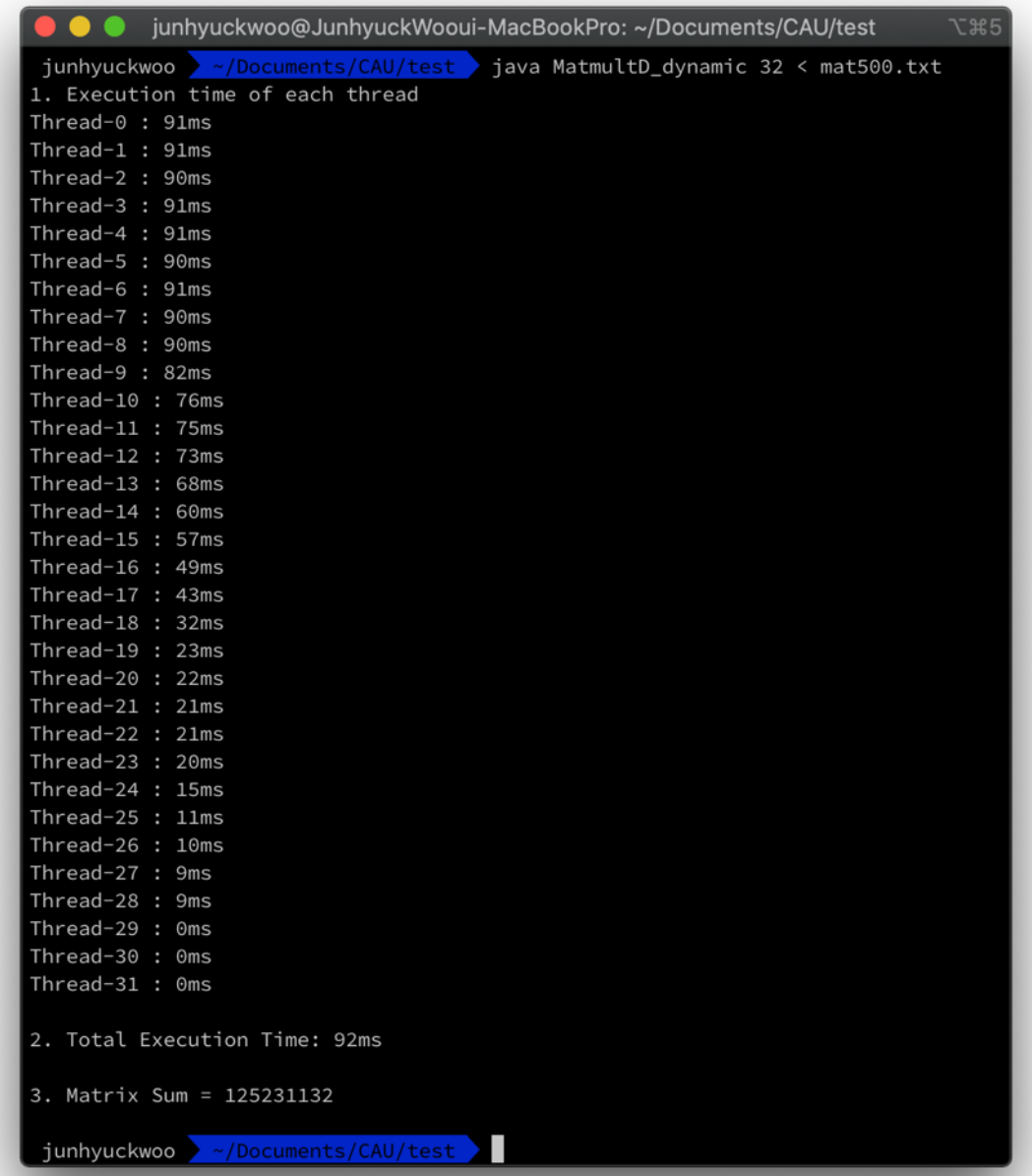
- Thread #14

```
junhyuckwoo ~/Documents/CAU/test java MatmultD_dynamic 14 < mat500.txt
1. Execution time of each thread
Thread-0 : 93ms
Thread-1 : 93ms
Thread-2 : 93ms
Thread-3 : 94ms
Thread-4 : 92ms
Thread-5 : 92ms
Thread-6 : 92ms
Thread-7 : 92ms
Thread-8 : 90ms
Thread-9 : 88ms
Thread-10 : 84ms
Thread-11 : 77ms
Thread-12 : 75ms
Thread-13 : 70ms

2. Total Execution Time: 95ms

3. Matrix Sum = 125231132
```

- Thread #16

```
junhyuckwoo@JunhyuckWooui-MacBookPro: ~/Documents/CAU/test ⌥⌘5
Thread-2 : 90ms
Thread-3 : 90ms
Thread-4 : 90ms
Thread-5 : 90ms
Thread-6 : 90ms
Thread-7 : 90ms
Thread-8 : 89ms
Thread-9 : 85ms
Thread-10 : 78ms
Thread-11 : 75ms
Thread-12 : 74ms
Thread-13 : 69ms
Thread-14 : 66ms
Thread-15 : 56ms

2. Total Execution Time: 92ms

3. Matrix Sum = 125231132

junhyuckwoo ~/Documents/CAU/test
```

- Thread #32


junhyuckwoo@JunhyuckWooui-MacBookPro: ~/Documents/CAU/test
⌥⌘5
junhyuckwoo ~/Documents/CAU/test java MatmultD_dynamic 32 < mat500.txt
1. Execution time of each thread
Thread-0 : 91ms
Thread-1 : 91ms
Thread-2 : 90ms
Thread-3 : 91ms
Thread-4 : 91ms
Thread-5 : 90ms
Thread-6 : 91ms
Thread-7 : 90ms
Thread-8 : 90ms
Thread-9 : 82ms
Thread-10 : 76ms
Thread-11 : 75ms
Thread-12 : 73ms
Thread-13 : 68ms
Thread-14 : 60ms
Thread-15 : 57ms
Thread-16 : 49ms
Thread-17 : 43ms
Thread-18 : 32ms
Thread-19 : 23ms
Thread-20 : 22ms
Thread-21 : 21ms
Thread-22 : 21ms
Thread-23 : 20ms
Thread-24 : 15ms
Thread-25 : 11ms
Thread-26 : 10ms
Thread-27 : 9ms
Thread-28 : 9ms
Thread-29 : 0ms
Thread-30 : 0ms
Thread-31 : 0ms
2. Total Execution Time: 92ms
3. Matrix Sum = 125231132
junhyuckwoo ~/Documents/CAU/test