# 2022.1 Multicore Computing, Project #1

## Problem 2

## Document

소프트웨어학부

20176342 송민준

### (i) Result

## (a) Execution environment

CPU : AMD Ryzen 5 2600X Six-Core Processor (12 CPUs), ~3.6GHz

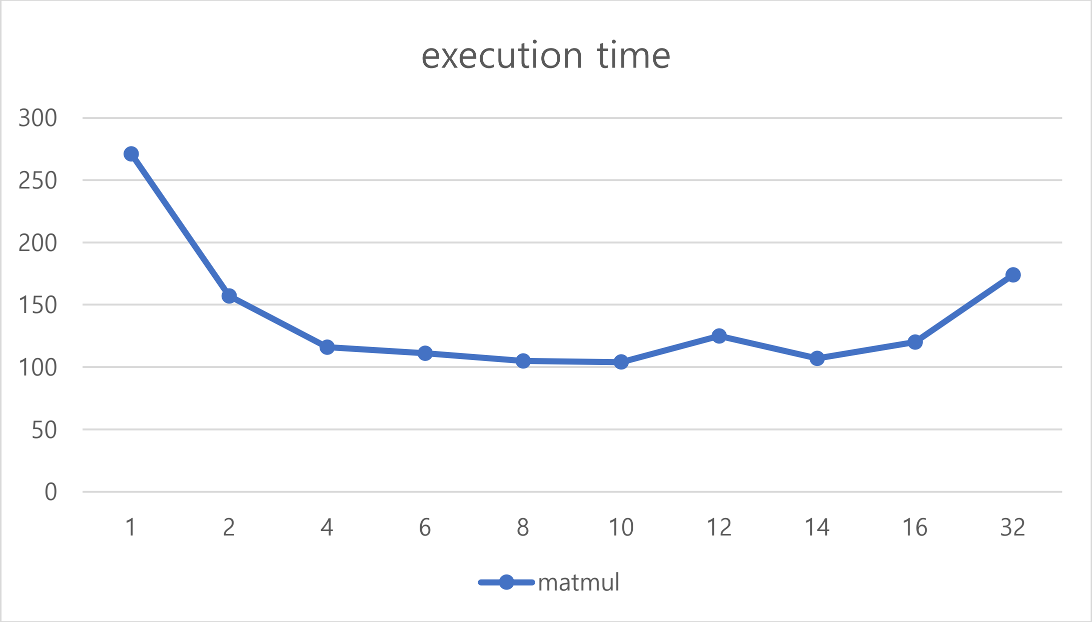
Memory : DDR4 16384MB RAM

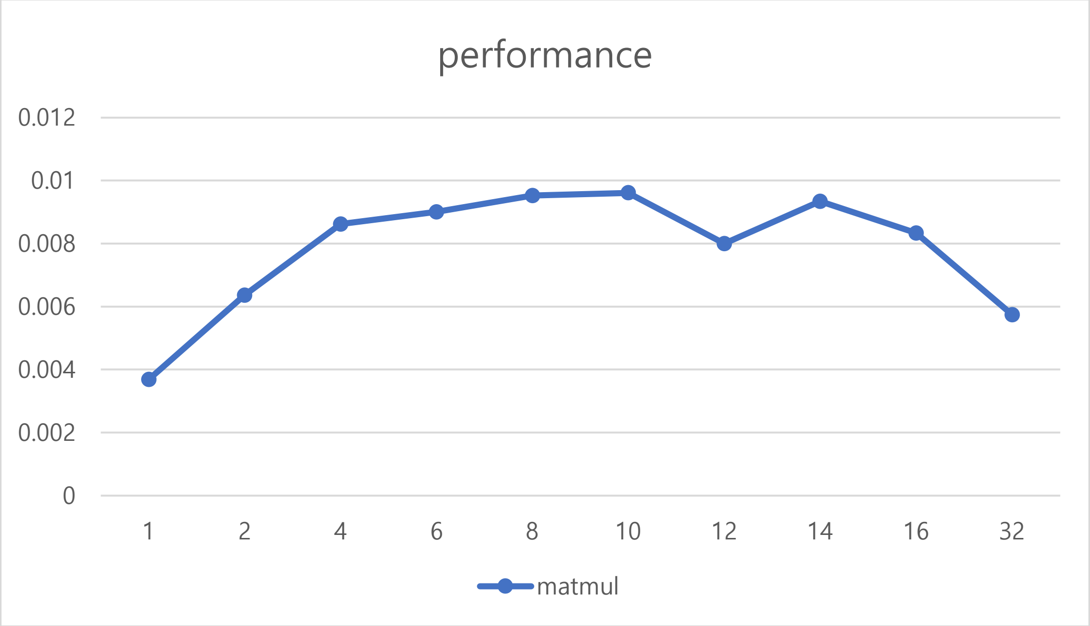
OS : Windows 10

## (b) Tables and graphs

| Thread# | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|
| Exec time | 271 | 157 | 116 | 111 | 105 | 104 | 125 | 107 | 120 | 174 |

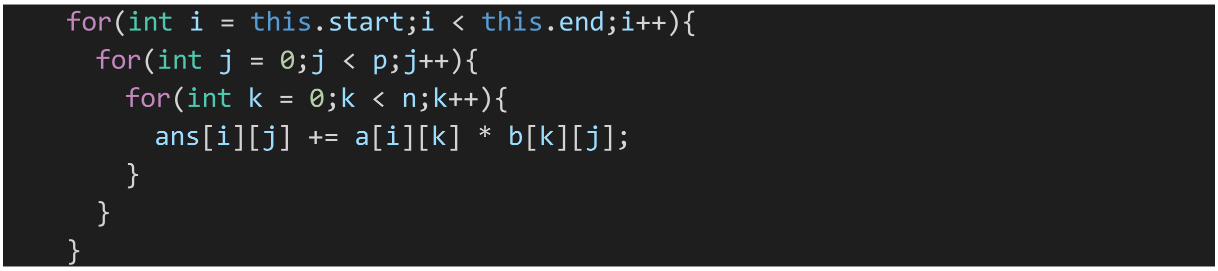
| Thread# | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|
| Performance (1/exec time) | 0.00369 | 0.00636 | 0.00862 | 0.00900 | 0.00952 | 0.00961 | 0.00800 | 0.00934 | 0.00833 | 0.00574 |


execution time
300
250
200
150
100
50
0
1
2
4
6
8
10
12
14
16
32
matmul


performance
0.012
0.01
0.008
0.006
0.004
0.002
0
1
2
4
6
8
10
12
14
16
32
matmul

## (c) Explanation of results

I implemented this matrix multiplication project by dividing 3-nested for iteration.



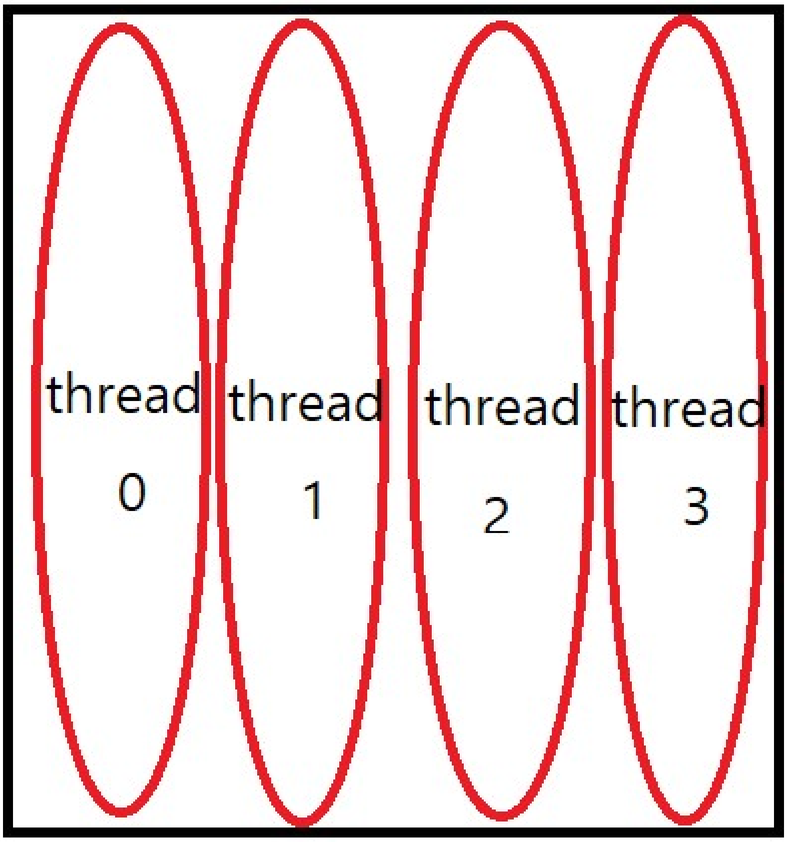
```
    for(int i = this.start;i < this.end;i++){
      for(int j = 0;j < p;j++){
        for(int k = 0;k < n;k++){
          ans[i][j] += a[i][k] * b[k][j];
        }
      }
    }
```

This iteration is the basic of multiplication of two matrices.

At the outermost loop of for, i is divided by start and end point.

If matrix is 100x100 and If I have 4 threads, each threads has start and end value like

0~25, 26~50, 51~75, 76~100 respectively.



If the matrix above is the multiplication result of the two matrices, Different threads take charge of the results of each column of the matrix.

So more threads reduce execution time, but too many threads slow execution time.

The more threads, the higher the overhead cost of context switching between threads, the slower the execution time.

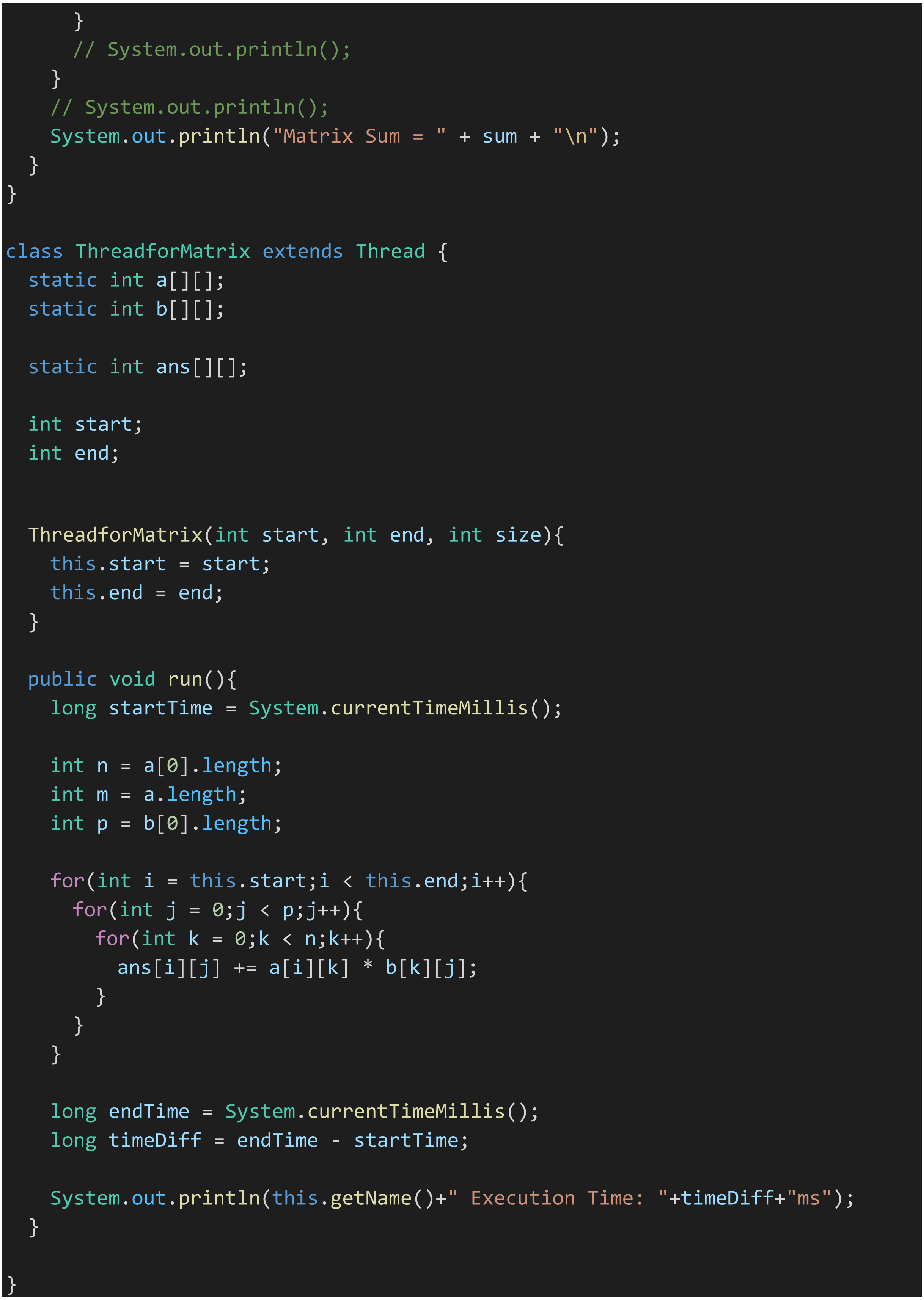
In this matrix multiplication example, 8 to 10 threads appear to be the most efficient.

I think the results will be similar even if dynamic load balancing is implemented in 'for iteration'. Because the calculation time for multiplication is similar no matter what, so the time required for each thread is already similar.

### (d) entire JAVA source code and screen capture image of program execution and output

MatmultD.java

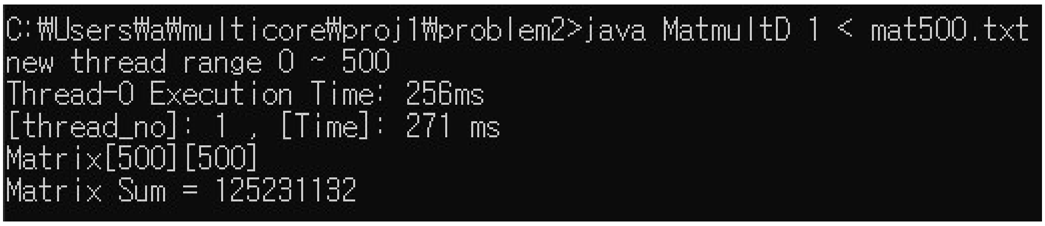
```java
import java.util.*;
import java.lang.*;

// 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
{
  private static Scanner sc = new Scanner(System.in);
  public static void main(String [] args)
  {
    int thread_no=0;
    if (args.length==1) thread_no = Integer.valueOf(args[0]);
    else thread_no = 2;

    int a[][]=readMatrix();
    int b[][]=readMatrix();

    long startTime = System.currentTimeMillis();

    ThreadforMatrix.a = a;
    ThreadforMatrix.b = b;
    ThreadforMatrix.ans = new int[a.length][a.length];

    ArrayList<ThreadforMatrix> thread_arr = new ArrayList<ThreadforMatrix>();

    for(int i = 0; i<thread_no;i++){
      int start = i*(a.length/thread_no);
      int end = i == thread_no-1 ? a.length : (i+1)*(a.length/thread_no);
      System.out.println("new thread range "+start+ " ~ "+end);
      ThreadforMatrix thread = new ThreadforMatrix(start,end, a.length);
      thread_arr.add(thread);
      thread.start();

    }

    for(int i = 0;i<thread_arr.size();i++){
      try {
        thread_arr.get(i).join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }

    long endTime = System.currentTimeMillis();

    System.out.printf("[thread_no]:%2d , [Time]:%4d ms\n", thread_no, endTime-
startTime);

    printMatrix(ThreadforMatrix.ans);


  }

   public static 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;
   }

  public static void printMatrix(int[][] mat) {
  System.out.println("Matrix["+mat.length+"]["+mat[0].length+"]");
    int rows = mat.length;
    int columns = mat[0].length;
    int sum = 0;
    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < columns; j++) {
        // System.out.printf("%4d " , mat[i][j]);
        sum+=mat[i][j];
```



```java
      }
      // System.out.println();
    }
    // System.out.println();
    System.out.println("Matrix Sum = " + sum + "\n");
  }
}

class ThreadforMatrix extends Thread {
  static int a[][];
  static int b[][];

  static int ans[][];

  int start;
  int end;


  ThreadforMatrix(int start, int end, int size){
    this.start = start;
    this.end = end;
  }

  public void run(){
    long startTime = System.currentTimeMillis();

    int n = a[0].length;
    int m = a.length;
    int p = b[0].length;

    for(int i = this.start;i < this.end;i++){
      for(int j = 0;j < p;j++){
        for(int k = 0;k < n;k++){
          ans[i][j] += a[i][k] * b[k][j];
        }
      }
    }

    long endTime = System.currentTimeMillis();
    long timeDiff = endTime - startTime;

    System.out.println(this.getName()+" Execution Time: "+timeDiff+"ms");
  }

}
```

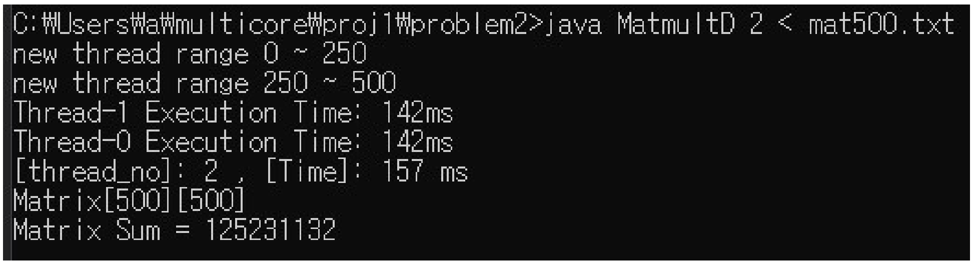
## MatmultD.java

MatmultD thread #1



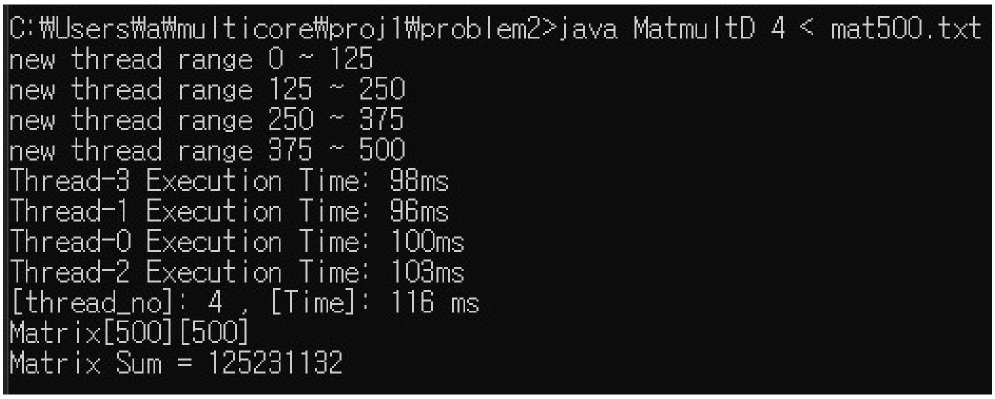
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 1 < mat500.txt
new thread range 0 ~ 500
Thread-0 Execution Time: 256ms
[thread_no]: 1 , [Time]: 271 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #2



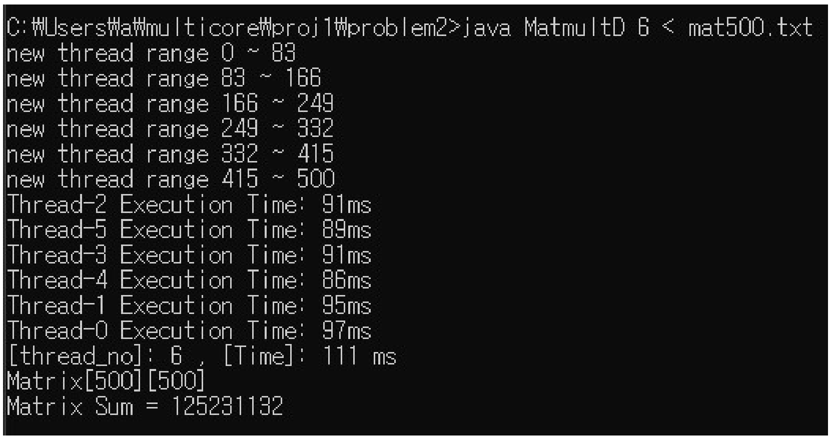
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 2 < mat500.txt
new thread range 0 ~ 250
new thread range 250 ~ 500
Thread-1 Execution Time: 142ms
Thread-0 Execution Time: 142ms
[thread_no]: 2 , [Time]: 157 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #4



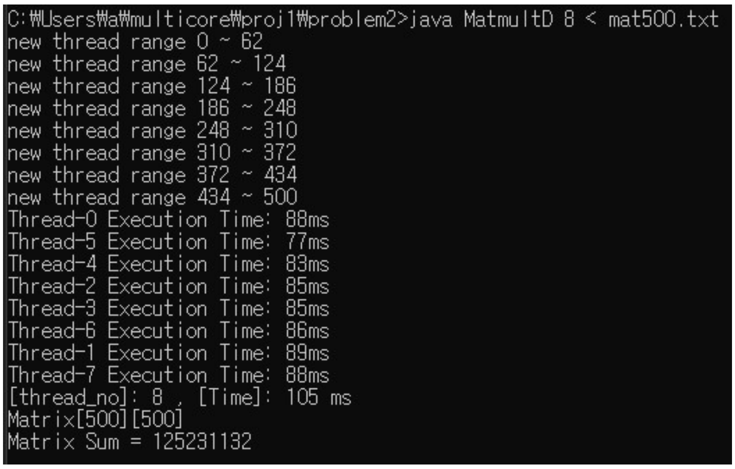
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 4 < mat500.txt
new thread range 0 ~ 125
new thread range 125 ~ 250
new thread range 250 ~ 375
new thread range 375 ~ 500
Thread-3 Execution Time: 98ms
Thread-1 Execution Time: 96ms
Thread-0 Execution Time: 100ms
Thread-2 Execution Time: 103ms
[thread_no]: 4 , [Time]: 116 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #6



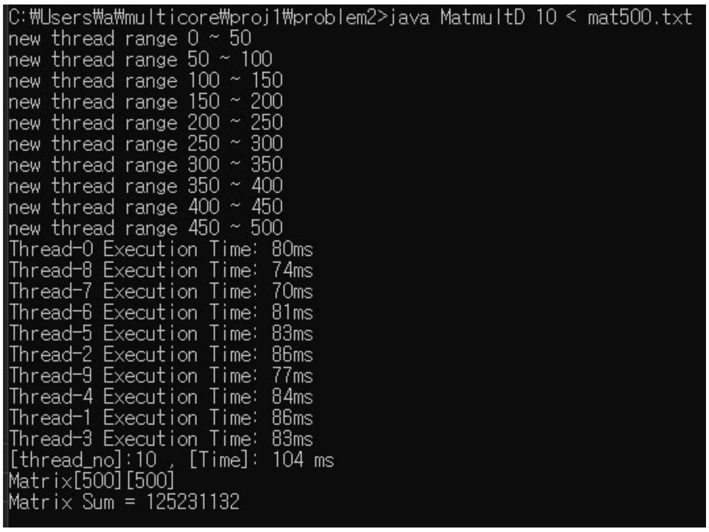
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 6 < mat500.txt
new thread range 0 ~ 83
new thread range 83 ~ 166
new thread range 166 ~ 249
new thread range 249 ~ 332
new thread range 332 ~ 415
new thread range 415 ~ 500
Thread-2 Execution Time: 91ms
Thread-5 Execution Time: 89ms
Thread-3 Execution Time: 91ms
Thread-4 Execution Time: 86ms
Thread-1 Execution Time: 95ms
Thread-0 Execution Time: 97ms
[thread_no]: 6 , [Time]: 111 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #8



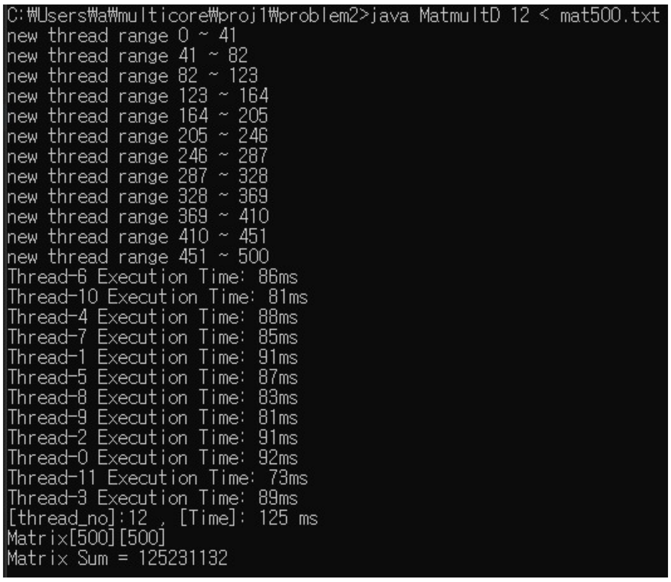
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 8 < mat500.txt
new thread range 0 ~ 62
new thread range 62 ~ 124
new thread range 124 ~ 186
new thread range 186 ~ 248
new thread range 248 ~ 310
new thread range 310 ~ 372
new thread range 372 ~ 434
new thread range 434 ~ 500
Thread-0 Execution Time: 88ms
Thread-5 Execution Time: 77ms
Thread-4 Execution Time: 83ms
Thread-2 Execution Time: 85ms
Thread-3 Execution Time: 85ms
Thread-6 Execution Time: 86ms
Thread-1 Execution Time: 89ms
Thread-7 Execution Time: 88ms
[thread_no]: 8 , [Time]: 105 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #10



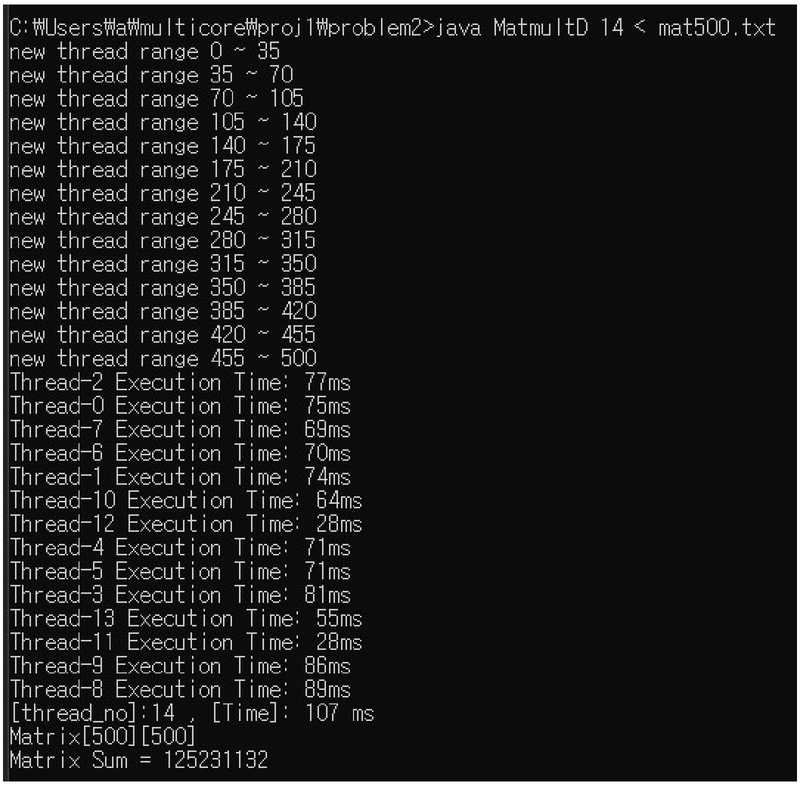
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 10 < mat500.txt
new thread range 0 ~ 50
new thread range 50 ~ 100
new thread range 100 ~ 150
new thread range 150 ~ 200
new thread range 200 ~ 250
new thread range 250 ~ 300
new thread range 300 ~ 350
new thread range 350 ~ 400
new thread range 400 ~ 450
new thread range 450 ~ 500
Thread-0 Execution Time: 80ms
Thread-8 Execution Time: 74ms
Thread-7 Execution Time: 70ms
Thread-6 Execution Time: 81ms
Thread-5 Execution Time: 83ms
Thread-2 Execution Time: 86ms
Thread-9 Execution Time: 77ms
Thread-4 Execution Time: 84ms
Thread-1 Execution Time: 86ms
Thread-3 Execution Time: 83ms
[thread_no]:10 , [Time]: 104 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #12



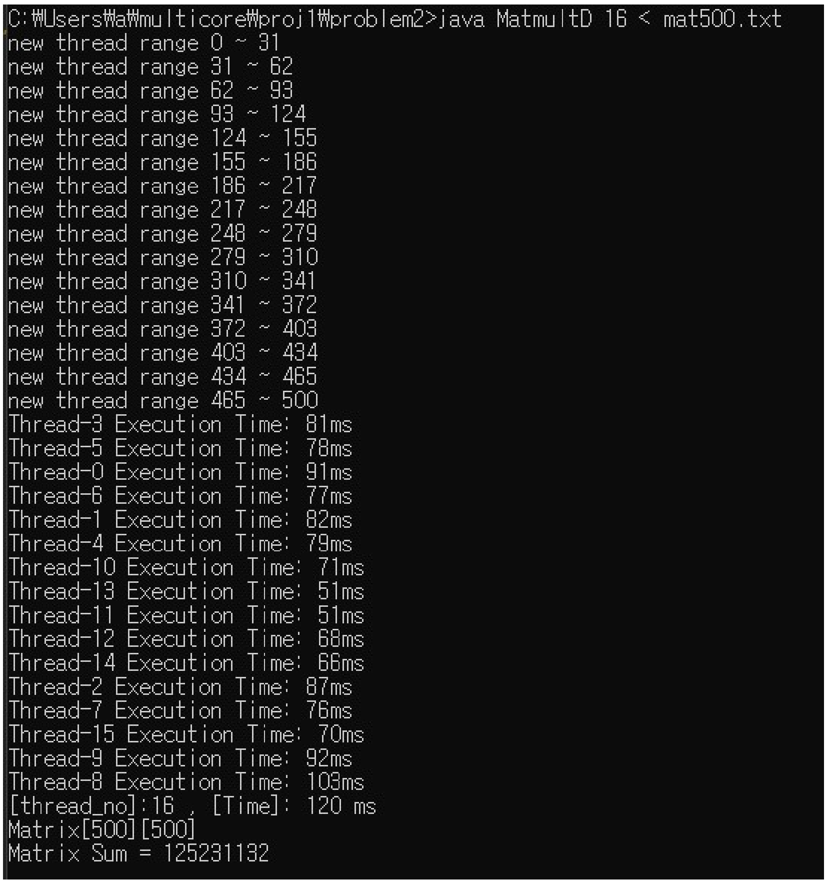
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 12 < mat500.txt
new thread range 0 ~ 41
new thread range 41 ~ 82
new thread range 82 ~ 123
new thread range 123 ~ 164
new thread range 164 ~ 205
new thread range 205 ~ 246
new thread range 246 ~ 287
new thread range 287 ~ 328
new thread range 328 ~ 369
new thread range 369 ~ 410
new thread range 410 ~ 451
new thread range 451 ~ 500
Thread-6 Execution Time: 86ms
Thread-10 Execution Time: 81ms
Thread-4 Execution Time: 88ms
Thread-7 Execution Time: 85ms
Thread-1 Execution Time: 91ms
Thread-5 Execution Time: 87ms
Thread-8 Execution Time: 83ms
Thread-9 Execution Time: 81ms
Thread-2 Execution Time: 91ms
Thread-0 Execution Time: 92ms
Thread-11 Execution Time: 73ms
Thread-3 Execution Time: 89ms
[thread_no]:12 , [Time]: 125 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #14



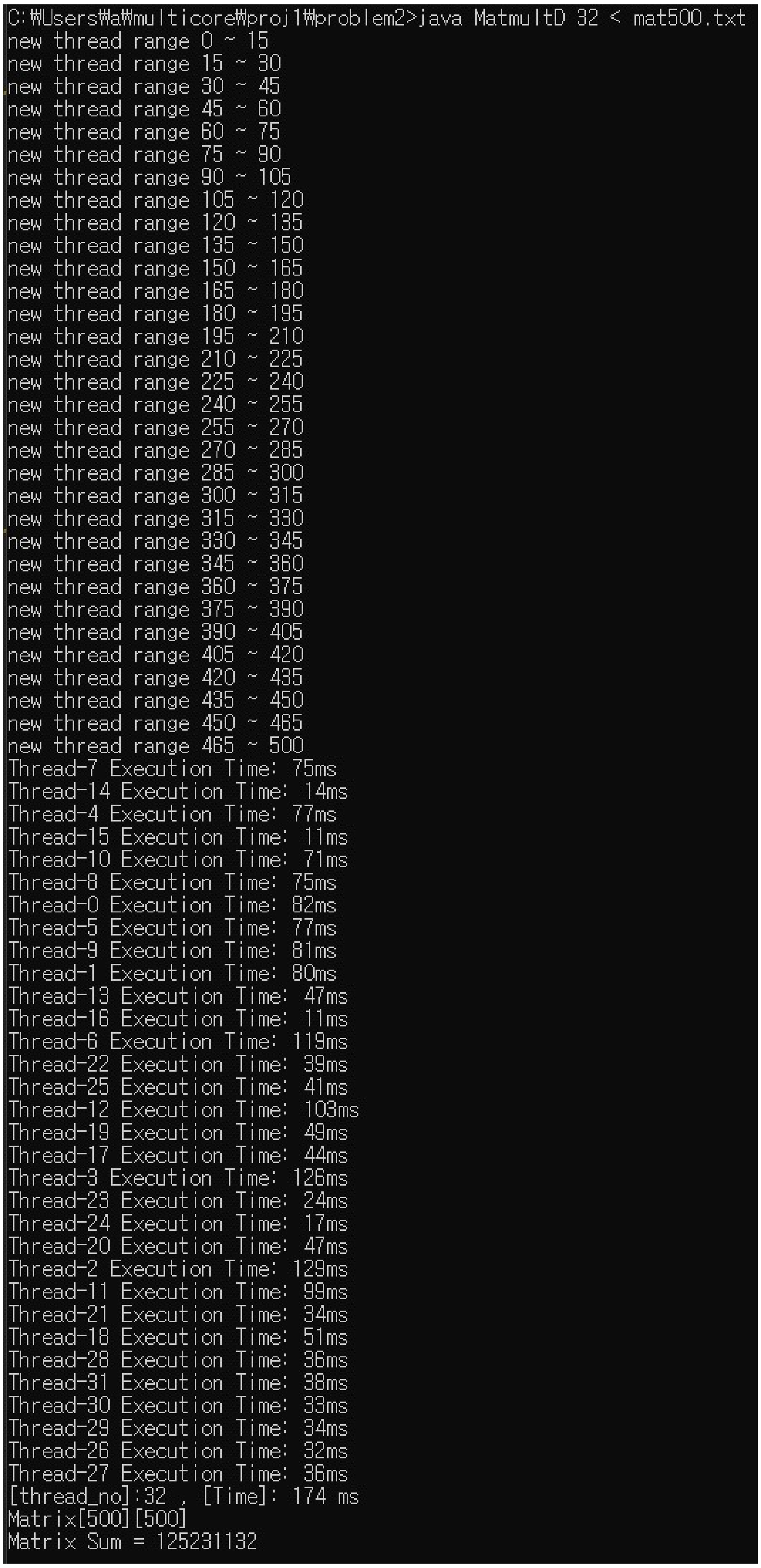
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 14 < mat500.txt
new thread range 0 ~ 35
new thread range 35 ~ 70
new thread range 70 ~ 105
new thread range 105 ~ 140
new thread range 140 ~ 175
new thread range 175 ~ 210
new thread range 210 ~ 245
new thread range 245 ~ 280
new thread range 280 ~ 315
new thread range 315 ~ 350
new thread range 350 ~ 385
new thread range 385 ~ 420
new thread range 420 ~ 455
new thread range 455 ~ 500
Thread-2 Execution Time: 77ms
Thread-0 Execution Time: 75ms
Thread-7 Execution Time: 69ms
Thread-6 Execution Time: 70ms
Thread-1 Execution Time: 74ms
Thread-10 Execution Time: 64ms
Thread-12 Execution Time: 28ms
Thread-4 Execution Time: 71ms
Thread-5 Execution Time: 71ms
Thread-3 Execution Time: 81ms
Thread-13 Execution Time: 55ms
Thread-11 Execution Time: 28ms
Thread-9 Execution Time: 86ms
Thread-8 Execution Time: 89ms
[thread_no]:14 , [Time]: 107 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #16



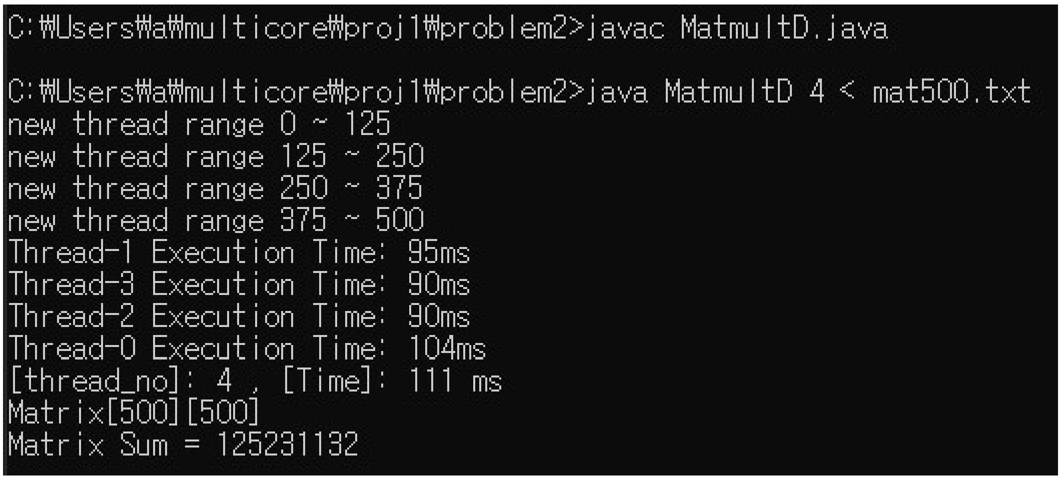
```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 16 < mat500.txt
new thread range 0 ~ 31
new thread range 31 ~ 62
new thread range 62 ~ 93
new thread range 93 ~ 124
new thread range 124 ~ 155
new thread range 155 ~ 186
new thread range 186 ~ 217
new thread range 217 ~ 248
new thread range 248 ~ 279
new thread range 279 ~ 310
new thread range 310 ~ 341
new thread range 341 ~ 372
new thread range 372 ~ 403
new thread range 403 ~ 434
new thread range 434 ~ 465
new thread range 465 ~ 500
Thread-3 Execution Time: 81ms
Thread-5 Execution Time: 78ms
Thread-0 Execution Time: 91ms
Thread-6 Execution Time: 77ms
Thread-1 Execution Time: 82ms
Thread-4 Execution Time: 79ms
Thread-10 Execution Time: 71ms
Thread-13 Execution Time: 51ms
Thread-11 Execution Time: 51ms
Thread-12 Execution Time: 68ms
Thread-14 Execution Time: 66ms
Thread-2 Execution Time: 87ms
Thread-7 Execution Time: 76ms
Thread-15 Execution Time: 70ms
Thread-9 Execution Time: 92ms
Thread-8 Execution Time: 103ms
[thread_no]:16 , [Time]: 120 ms
Matrix[500][500]
Matrix Sum = 125231132
```

MatmultD thread #32



```
C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 32 < mat500.txt
new thread range 0 ~ 15
new thread range 15 ~ 30
new thread range 30 ~ 45
new thread range 45 ~ 60
new thread range 60 ~ 75
new thread range 75 ~ 90
new thread range 90 ~ 105
new thread range 105 ~ 120
new thread range 120 ~ 135
new thread range 135 ~ 150
new thread range 150 ~ 165
new thread range 165 ~ 180
new thread range 180 ~ 195
new thread range 195 ~ 210
new thread range 210 ~ 225
new thread range 225 ~ 240
new thread range 240 ~ 255
new thread range 255 ~ 270
new thread range 270 ~ 285
new thread range 285 ~ 300
new thread range 300 ~ 315
new thread range 315 ~ 330
new thread range 330 ~ 345
new thread range 345 ~ 360
new thread range 360 ~ 375
new thread range 375 ~ 390
new thread range 390 ~ 405
new thread range 405 ~ 420
new thread range 420 ~ 435
new thread range 435 ~ 450
new thread range 450 ~ 465
new thread range 465 ~ 500
Thread-7 Execution Time: 75ms
Thread-14 Execution Time: 14ms
Thread-4 Execution Time: 77ms
Thread-15 Execution Time: 11ms
Thread-10 Execution Time: 71ms
Thread-8 Execution Time: 75ms
Thread-0 Execution Time: 82ms
Thread-5 Execution Time: 77ms
Thread-9 Execution Time: 81ms
Thread-1 Execution Time: 80ms
Thread-13 Execution Time: 47ms
Thread-16 Execution Time: 11ms
Thread-6 Execution Time: 119ms
Thread-22 Execution Time: 39ms
Thread-25 Execution Time: 41ms
Thread-12 Execution Time: 103ms
Thread-19 Execution Time: 49ms
Thread-17 Execution Time: 44ms
Thread-3 Execution Time: 126ms
Thread-23 Execution Time: 24ms
Thread-24 Execution Time: 17ms
Thread-20 Execution Time: 47ms
Thread-2 Execution Time: 129ms
Thread-11 Execution Time: 99ms
Thread-21 Execution Time: 34ms
Thread-18 Execution Time: 51ms
Thread-28 Execution Time: 36ms
Thread-31 Execution Time: 38ms
Thread-30 Execution Time: 33ms
Thread-29 Execution Time: 34ms
Thread-26 Execution Time: 32ms
Thread-27 Execution Time: 36ms
[thread_no]:32 , [Time]: 174 ms
Matrix[500][500]
Matrix Sum = 125231132
```

## How to compile and execute



```
C:₩Users₩a₩multicore₩proj1₩problem2>javac MatmultD.java

C:₩Users₩a₩multicore₩proj1₩problem2>java MatmultD 4 < mat500.txt
new thread range 0 ~ 125
new thread range 125 ~ 250
new thread range 250 ~ 375
new thread range 375 ~ 500
Thread-1 Execution Time: 95ms
Thread-3 Execution Time: 90ms
Thread-2 Execution Time: 90ms
Thread-0 Execution Time: 104ms
[thread_no]: 4 , [Time]: 111 ms
Matrix[500][500]
Matrix Sum = 125231132
```

Just use 'javac' to compile in my directory(problem2) and run 'java MatmultD #num_thread < matrix file name like above.