2023.1 Multicore Computing, Project #1

Problem 2

Course / Class:	Multicore Computing / Class 01
Instructor:	Bong-Soo Sohn
Date:	2023. 04. 18
Student ID:	20184757
Student Name:	Youngseok Joo

Table of Contents

1. Environment

- Hardware
  - MacBook Air (M2, 2022)
  - Processor: Apple M2 (8 Core — 4 Efficiency + 4 Performance, Maximum clock speed 3.49 GHz)
  - Memory: 16 GB (SoC — 6,400 MT/s LPDDR5 SDRAM in a unified memory configuration)
- Operating System
  - macOS Ventura 13.3.1
- Testing Environment
  - macOS Terminal (version 2.13) — zsh
  - openjdk 16.0.1 2021-04-20
    OpenJDK Runtime Environment AdoptOpenJDK-16.0.1+9 (build 16.0.1+9)
    OpenJDK 64-Bit Server VM AdoptOpenJDK-16.0.1+9 (build 16.0.1+9, mixed mode, sharing)

2. Table / Graph

1) Execution Time

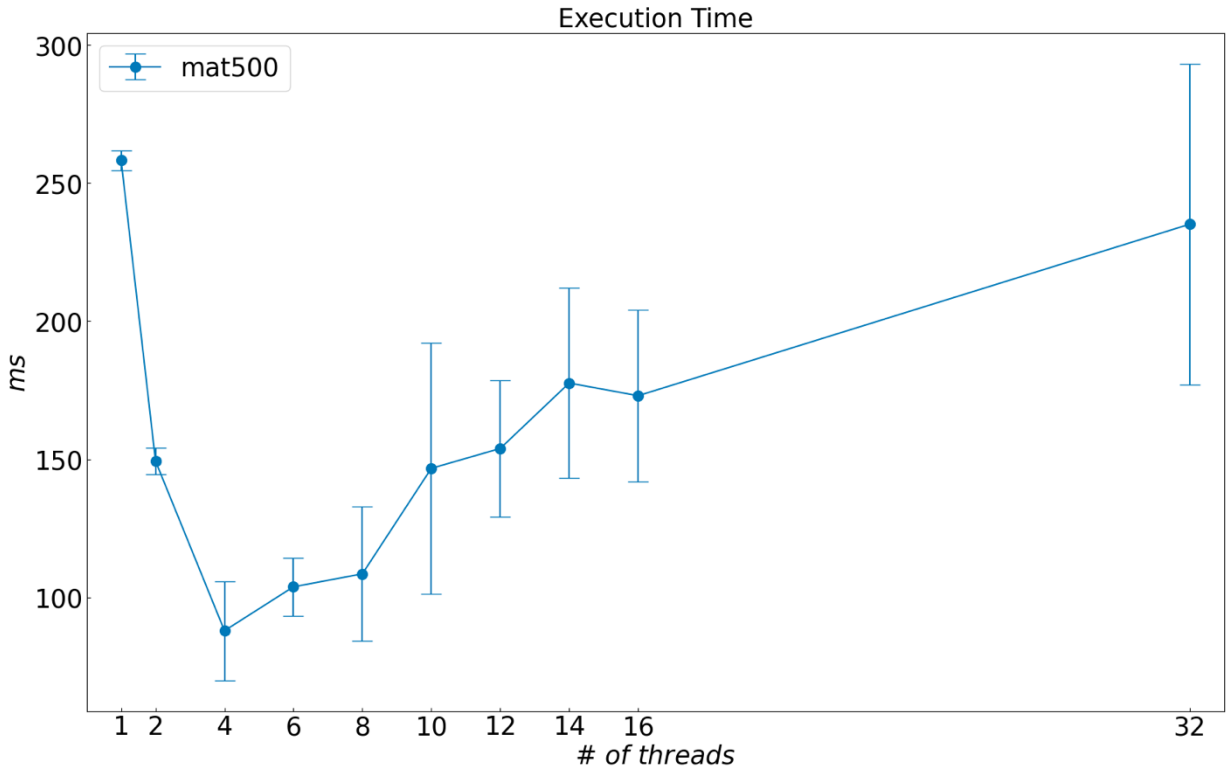


Figure 1. Error Bar Graph of Execution Time using Static (Cyclic) Load Balancing (10-Fold).

Table 1. Table showing Average Execution Time using Static (Cyclic) Load Balancing (10-Fold).

	1	2	4	6	8	10	12	14	16	32
Execution Time *(ms)*	258.3	149.3	87.9	103.8	108.5	146.7	153.9	177.6	173	235.1

2) Performance

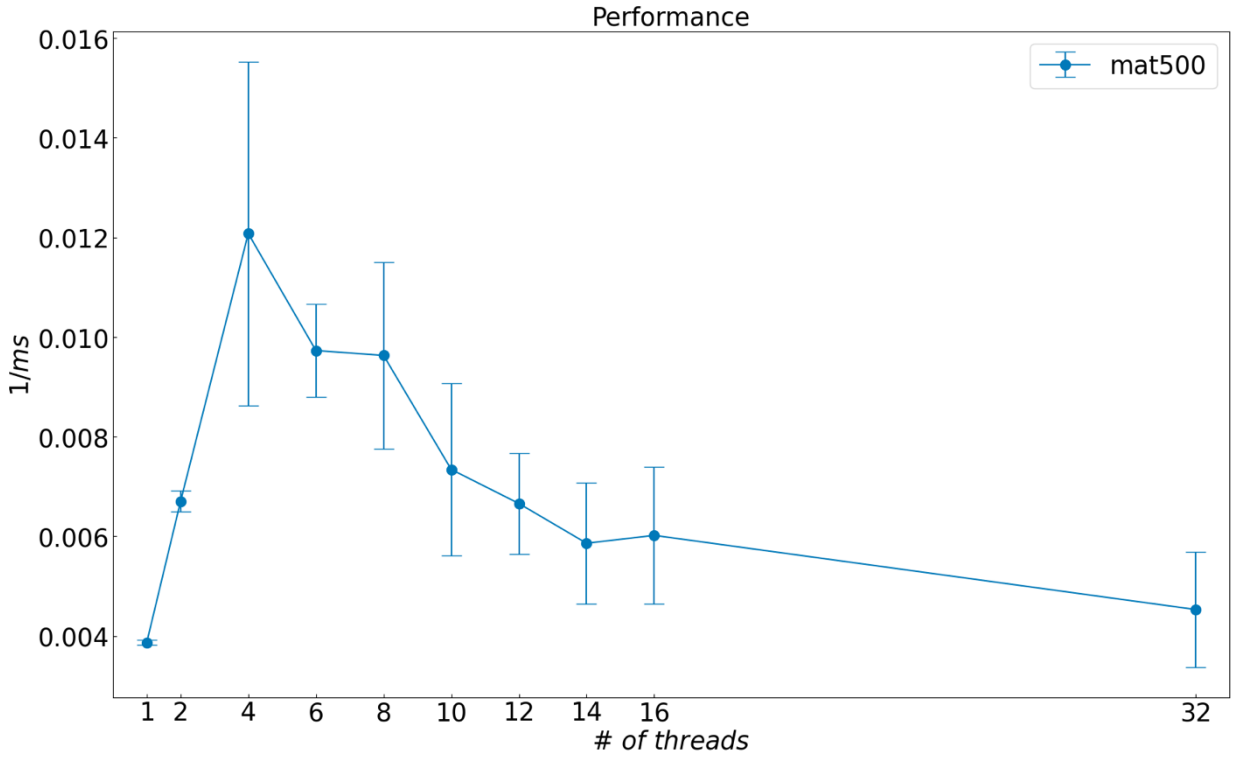


Figure 2. Error Bar Graph of Performance using Static (Cyclic) Load Balancing (10-Fold).

Table 2. Table showing Average Performance using Static (Cyclic) Load Balancing (10-Fold).

	1	2	4	6	8	10	12	14	16	32
Performance *(1/ms)*	0.0038 7221	0.0067 0452	0.0120 7606	0.0097 2814	0.0096 296	0.0073 4175	0.0066 5904	0.0058 6182	0.0060 2004	0.0045 3206

3. Explanation / Analysis

The given work is multiplying two matrices. For testing execution time and performance shown in section 2, input file ‘mat500.txt’ was given. The algorithm for multiplying matrices is three nested for-loops. Breaking down the first loop allows us to achieve multi-threaded programming with a shorter execution time. The multi-threaded version program source code is implemented with a static load balancing using cyclic decomposition.

As shown in Figure 1 and Figure 2, the average execution time decreases, and the average performance increases until using four threads. But, using more than four threads makes the average execution time increase again, lowering the average performance to using 32 threads.

This decrease in performance when using more than four threads may be caused by the limit of the physical core. As stated in Section 1, Apple M2 CPU has four performance cores and four efficiency cores. This information estimates that only four performance cores have worked when using four or more threads. When using more than four threads, CPU scheduling has caused the execution time to increase.

4. Source Code / Execution Result

1) Source Code

```
import java.util.*;

class MatmulThread extends Thread {
  int[][] ans;
  int[][] a;
  int[][] b;
  int thread_idx;
  int num_threads;
  long timeDiff;

  // Constructor, gets pointer of answer matrix and original matrices a and b,
  // gets thread index and number of threads for cyclic static load balancing
  MatmulThread(int[][] ans, int[][] a, int[][] b, int i, int n) {
    this.ans = ans;
    this.a = a;
    this.b = b;
    thread_idx = i;
    num_threads = n;
  }

  // run()
  public void run() {
    long startTime = System.currentTimeMillis();
    int n = a[0].length;
    int m = a.length;
    int p = b[0].length;
    for (int i = thread_idx; i < m; i += num_threads) { // cyclic loop
      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();
    timeDiff = endTime - startTime;
  }
}

public class MatmultD_multithread {
  private static Scanner sc = new Scanner(System.in);
  private static int NUM_THREADS = 8;
  private static long[] thread_exec_times;

  public static void main(String[] args) {
    if (args.length == 1) // if 1 argument
      NUM_THREADS = Integer.valueOf(args[0]); // set as NUM_THREADS

    thread_exec_times = new long[NUM_THREADS];

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

    long startTime = System.currentTimeMillis();
    int[][] c = multMatrix(a, b);
    long endTime = System.currentTimeMillis();
    long timeDiff = endTime - startTime;
    for (int i = 0; i < NUM_THREADS; i++) { // print thread execution time
      System.out.println("Thread #" + i + " Execution Time:" + thread_exec_times[i] + "ms");
    }
    System.out.println();
    System.out.println("Program Execution Time: " + timeDiff + "ms"); // print program execution time
    System.out.println();
    printMatrix(c, false); // print sum of elements in result matrix
  }

  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, boolean printAll) {
    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++) {
        if (printAll) // if printAll print elements
          System.out.printf("%4d ", mat[i][j]);
        sum += mat[i][j];
      }
      if (printAll)
        System.out.println();
    }
    if (printAll)
      System.out.println();
    System.out.println("Matrix Sum = " + sum + "\n"); // print sum of elements
  }

  public static int[][] multMatrix(int a[][], int b[][]) {// a[m][n], b[n][p]
    if (a.length == 0)
      return new int[0][0];
    if (a[0].length != b.length)
      return null; // invalid dims

    int m = a.length;
    int p = b[0].length;
    int ans[][] = new int[m][p];
    int i;

    MatmulThread[] matmulThreads = new MatmulThread[NUM_THREADS]; // thread array
    for (i = 0; i < NUM_THREADS; i++) {
      matmulThreads[i] = new MatmulThread(ans, a, b, i, NUM_THREADS); // make thread
      matmulThreads[i].start(); // run thread
    }

    try {
      for (i = 0; i < NUM_THREADS; i++) {
        matmulThreads[i].join(); // wait for thread end
        thread_exec_times[i] = matmulThreads[i].timeDiff;
      }
    } catch (InterruptedException e) {
    }

    return ans;
  }
}
```

This source code can be compiled in terminal with the following command.

```
$ javac MatmultD_multithread.java
```

And the compiled program can be executed with the following command.

```
$ java MatmultD_multithread < mat500.txt
```

The default thread number is 8, but thread number can be specified using command line argument as below command, adding integer after the above command. Also, you can specify input file.

```
$ java MatmultD_multithread [NUM_THREADS] < [INPUT_FILE]
```

(Example. "`$ java MatmultD_multithread 32 < mat500.txt`")

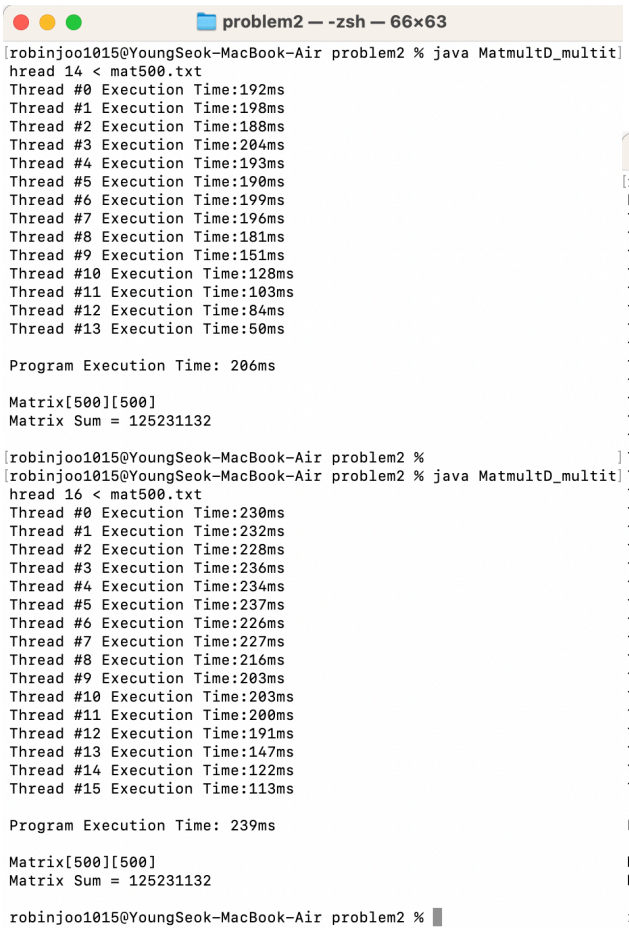
2) Execution Result

problem2 — -zsh — 66×63

```
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multit
hread 1 < mat500.txt
Thread #0 Execution Time:267ms

Program Execution Time: 269ms

Matrix[500][500]
Matrix Sum = 125231132

robinjoo1015@YoungSeok-MacBook-Air problem2 %
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multit
hread 2 < mat500.txt
Thread #0 Execution Time:154ms
Thread #1 Execution Time:154ms

Program Execution Time: 157ms

Matrix[500][500]
Matrix Sum = 125231132

robinjoo1015@YoungSeok-MacBook-Air problem2 %
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multit
hread 4 < mat500.txt
Thread #0 Execution Time:110ms
Thread #1 Execution Time:105ms
Thread #2 Execution Time:109ms
Thread #3 Execution Time:105ms

Program Execution Time: 111ms

Matrix[500][500]
Matrix Sum = 125231132

robinjoo1015@YoungSeok-MacBook-Air problem2 %
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multit
hread 6 < mat500.txt
Thread #0 Execution Time:134ms
Thread #1 Execution Time:134ms
Thread #2 Execution Time:125ms
Thread #3 Execution Time:134ms
Thread #4 Execution Time:132ms
Thread #5 Execution Time:131ms

Program Execution Time: 137ms

Matrix[500][500]
Matrix Sum = 125231132

robinjoo1015@YoungSeok-MacBook-Air problem2 %
```

problem2 — -zsh — 66×63

```
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multit
hread 8 < mat500.txt
Thread #0 Execution Time:98ms
Thread #1 Execution Time:92ms
Thread #2 Execution Time:105ms
Thread #3 Execution Time:99ms
Thread #4 Execution Time:104ms
Thread #5 Execution Time:94ms
Thread #6 Execution Time:96ms
Thread #7 Execution Time:114ms

Program Execution Time: 118ms

Matrix[500][500]
Matrix Sum = 125231132

robinjoo1015@YoungSeok-MacBook-Air problem2 %
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multit
hread 10 < mat500.txt
Thread #0 Execution Time:105ms
Thread #1 Execution Time:103ms
Thread #2 Execution Time:97ms
Thread #3 Execution Time:99ms
Thread #4 Execution Time:93ms
Thread #5 Execution Time:97ms
Thread #6 Execution Time:96ms
Thread #7 Execution Time:93ms
Thread #8 Execution Time:87ms
Thread #9 Execution Time:83ms

Program Execution Time: 107ms

Matrix[500][500]
Matrix Sum = 125231132

robinjoo1015@YoungSeok-MacBook-Air problem2 %
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multit
hread 12 < mat500.txt
Thread #0 Execution Time:236ms
Thread #1 Execution Time:238ms
Thread #2 Execution Time:235ms
Thread #3 Execution Time:233ms
Thread #4 Execution Time:231ms
Thread #5 Execution Time:238ms
Thread #6 Execution Time:236ms
Thread #7 Execution Time:230ms
Thread #8 Execution Time:225ms
Thread #9 Execution Time:209ms
Thread #10 Execution Time:174ms
Thread #11 Execution Time:202ms

Program Execution Time: 242ms

Matrix[500][500]
Matrix Sum = 125231132

robinjoo1015@YoungSeok-MacBook-Air problem2 %
```


problem2 — -zsh — 66×63
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multithread 14 < mat500.txt
Thread #0 Execution Time:192ms
Thread #1 Execution Time:198ms
Thread #2 Execution Time:188ms
Thread #3 Execution Time:204ms
Thread #4 Execution Time:193ms
Thread #5 Execution Time:190ms
Thread #6 Execution Time:199ms
Thread #7 Execution Time:196ms
Thread #8 Execution Time:181ms
Thread #9 Execution Time:151ms
Thread #10 Execution Time:128ms
Thread #11 Execution Time:103ms
Thread #12 Execution Time:84ms
Thread #13 Execution Time:50ms
Program Execution Time: 206ms
Matrix[500][500]
Matrix Sum = 125231132
robinjoo1015@YoungSeok-MacBook-Air problem2 %
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multithread 16 < mat500.txt
Thread #0 Execution Time:230ms
Thread #1 Execution Time:232ms
Thread #2 Execution Time:228ms
Thread #3 Execution Time:236ms
Thread #4 Execution Time:234ms
Thread #5 Execution Time:237ms
Thread #6 Execution Time:226ms
Thread #7 Execution Time:227ms
Thread #8 Execution Time:216ms
Thread #9 Execution Time:203ms
Thread #10 Execution Time:203ms
Thread #11 Execution Time:200ms
Thread #12 Execution Time:191ms
Thread #13 Execution Time:147ms
Thread #14 Execution Time:122ms
Thread #15 Execution Time:113ms
Program Execution Time: 239ms
Matrix[500][500]
Matrix Sum = 125231132
robinjoo1015@YoungSeok-MacBook-Air problem2 %


problem2 — -zsh — 66×63
robinjoo1015@YoungSeok-MacBook-Air problem2 % java MatmultD_multithread 32 < mat500.txt
Thread #0 Execution Time:211ms
Thread #1 Execution Time:175ms
Thread #2 Execution Time:200ms
Thread #3 Execution Time:211ms
Thread #4 Execution Time:209ms
Thread #5 Execution Time:203ms
Thread #6 Execution Time:194ms
Thread #7 Execution Time:184ms
Thread #8 Execution Time:196ms
Thread #9 Execution Time:192ms
Thread #10 Execution Time:180ms
Thread #11 Execution Time:159ms
Thread #12 Execution Time:187ms
Thread #13 Execution Time:170ms
Thread #14 Execution Time:172ms
Thread #15 Execution Time:162ms
Thread #16 Execution Time:155ms
Thread #17 Execution Time:152ms
Thread #18 Execution Time:168ms
Thread #19 Execution Time:151ms
Thread #20 Execution Time:158ms
Thread #21 Execution Time:140ms
Thread #22 Execution Time:159ms
Thread #23 Execution Time:155ms
Thread #24 Execution Time:132ms
Thread #25 Execution Time:105ms
Thread #26 Execution Time:75ms
Thread #27 Execution Time:19ms
Thread #28 Execution Time:31ms
Thread #29 Execution Time:24ms
Thread #30 Execution Time:9ms
Thread #31 Execution Time:18ms
Program Execution Time: 216ms
Matrix[500][500]
Matrix Sum = 125231132
robinjoo1015@YoungSeok-MacBook-Air problem2 %

5. Supplementary Table

Table 3. Supplementary Table showing all 10 execution times.

	1	2	4	6	8	10	12	14	16	32
Execution Time *(ms)*	254	148	51	95	111	246	148	117	165	245
	253	147	102	96	89	109	119	159	217	303
	256	151	96	122	89	116	133	198	156	281
	258	146	103	102	92	161	190	156	175	337
	256	145	94	116	161	137	174	153	176	199
	257	144	88	109	127	217	133	149	207	254
	263	148	95	93	91	116	139	205	172	164
	261	161	97	93	83	121	155	232	154	155
	264	154	55	116	137	134	149	188	103	176
	261	149	98	96	105	110	199	219	205	237