Problem2 report

Environnement

Os	Pop!_OS 22.04 LTS x86
CPU	Intel i7-8665U (8) @ 1.900GHz
Memory	16Gb
Java version	openjdk 17.0.6 2023-01-17

Build

In the `problem2` directory

```
javac MatmultD.java
```

How to use

```
java MatmultD NUM_THREAD < mat500.txt
```

If you want to see the execution time of each thread:

```
java MatmultD NUM_THREAD < mat500.txt | head -n NUM_THREAD
```

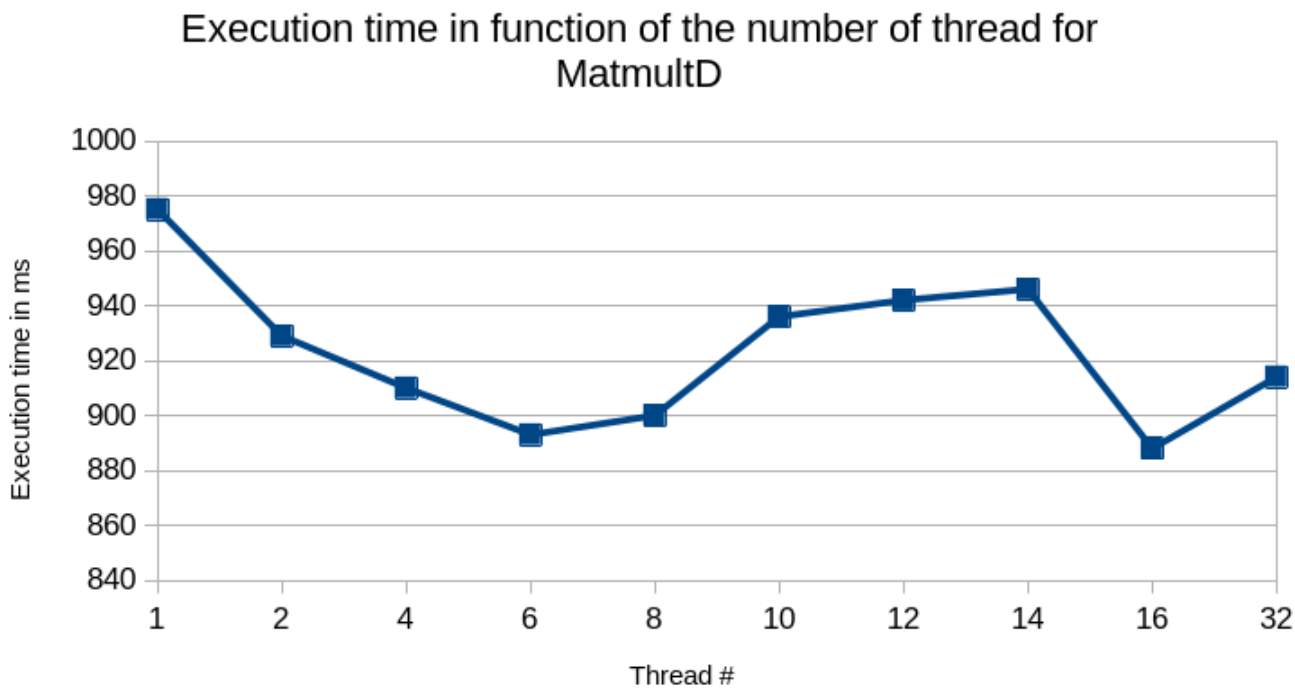
Results

Raw

Number of threads	Execution time in ms
1	975.000
2	929.000
4	910.000
6	893.000
8	900.000
10	936.000
12	942.000

Number of threads	Execution time in ms
14	946.000
16	888.000
32	914.000

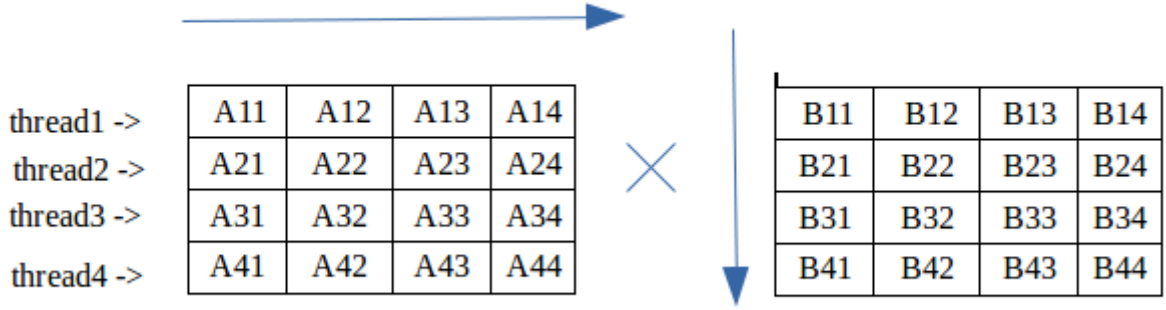
Graph



Interpretation

For this exercise, static load balencing seems to be the best way to calculate the product of 2 matrix. As the size of each matrix is known, we can divide the process in multiple threads.

Based on the following picture from geeks for geeks



I decided to use the `2D CYCLIC,*` task repartition, where one line of the matrix A is one task.

> We can also use the `2D BLOCK,*` task repartition

If we look at the results we can see that the speed increase between 6 and 16 threads. So we can say that 6 threads are enought for a `500 * 500` matrix multiplied by another `500 * 500` matrix.

Source code

MatmultD.java

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

class MatMulThread extends Thread {
  int _matrixA[][];
  int _matrixB[][];
  int _matrixResult[][];
  int _startLine;
  int _nThread;

  MatMulThread(int a[][], int b[][], int c[][], int startLine, int nThread)
{
    _matrixA = a;
    _matrixB = b;
    _matrixResult = c;
    _startLine = startLine;
    _nThread = nThread;
  }

  public void run() {
    long startTime = System.currentTimeMillis();
    multMatrix();
    long endTime = System.currentTimeMillis();
    long timeDiff = endTime - startTime;
    System.out.printf("Execution Time of thread %d : %d ms\n",
Thread.currentThread().getId(), timeDiff);
  }

  private void multMatrix() {
    if (_matrixA.length == 0)
      return;
    if (_matrixA[0].length != _matrixB.length)
      return;

    int A_MatrixLineSize = _matrixA[0].length;
    int A_MatrixColSize = _matrixA.length;
    int B_MatrixColSize = _matrixB.length;

    for (int i = _startLine; i < A_MatrixColSize; i += _nThread) {
      for (int j = 0; j < B_MatrixColSize; j++) {
        for (int k = 0; k < A_MatrixLineSize; k++) {
          // no lock because we can assume that only one thread will access
this memory
```

```
            // location
            _matrixResult[i][j] += _matrixA[i][k] * _matrixB[k][j];
          }
        }
      }
    }

  }

  public class MatmultD {
    private static Scanner sc = new Scanner(System.in);

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

      int a[][] = readMatrix();
      int b[][] = readMatrix();
      int m = a.length;
      int p = b[0].length;
      int c[][] = new int[m][p];
      MatMulThread threads[] = new MatMulThread[thread_no];

      long startTime = System.currentTimeMillis();

      for (int i = 0; i < threads.length; ++i) {
        threads[i] = new MatMulThread(a, b, c, i, thread_no);
        threads[i].start();
      }
      for (int i = 0; i < threads.length; ++i) {
        try {
          threads[i].join();
        } catch (InterruptedException e) {
        }
      }
      long endTime = System.currentTimeMillis();
      printMatrix(c);
      System.out.printf("[thread_no]:%2d , [Time]:%4d ms\n", thread_no,
  endTime - startTime);
    }

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

Execution screenshots

1 thread

2 threads

```
 500  567  560  473  545  538  466  550  502  570  541  513  555  492  505  555  538  499  480  495  519  507  509  481  539  505  500
512  544  507  516  516  501  503  506  534  511  547  520  470  501  498  515  531  487  485  512  496  547  530  508  478  547  522  5
34  488  547  505  531  532  497  504  500  568  518  495  526  497  525  507  517  519  441  518  496  499  500  525  483  499  515  56
3  532  502  493  531  538  536  476  558  557  515  497  500  478  554  523  525  494  524  567  538  511  543  501  513  517  523  508
  514  470  567  531  519  520  504  536  497  529  574  486  476  481  516  515  527  511  542  495  483  521  516  560  533  549  534
 488  521  528  486  546  555  509  529  503  505  515  528  544  517  543  548  475  531  544  527  559  525  485  513  525  498  489
519  505  529  535  511  492  500  500  521  506  535  508  503  511  539  489  540  559  545  505  555  547  495  521  547  567  541  4
88  492  504  531  493  530  535  508  480  531  526  567  534  510  516  540  536  508  523  485  499  497  548  523  519  524  505  52
7  526  537  532  558  512  507  533  565  504  529
 505  525  565  535  457  517  543  530  558  547  515  547  511  586  543  545  486  532  520  554  517  564  564  540  517  544  513
521  562  583  531  498  511  517  503  526  543  549  542  511  532  502  592  531  541  527  565  537  515  496  569  552  547  521  5
10  525  510  539  537  544  521  528  506  471  503  512  518  549  556  549  528  547  560  526  536  558  534  553  514  536  513  54
1  520  572  517  522  559  532  554  515  527  532  537  526  555  543  522  533  573  544  530  521  548  551  507  525  582  578  500
  524  521  560  505  536  568  541  526  515  548  517  534  566  535  514  512  555  541  502  531  526  571  511  515  545  506  532
 498  572  544  528  577  551  537  540  553  530  519  559  494  551  525  528  528  609  519  518  504  588  530  489  501  541  581
517  540  552  588  512  549  565  543  533  537  521  498  524  550  563  525  526  566  577  510  506  552  530  544  517  492  525  4
80  523  550  560  559  546  535  512  533  544  541  478  553  531  507  548  559  540  510  500  564  589  539  516  498  503  565  50
7  530  536  529  599  589  490  514  549  517  577  505  501  534  534  561  493  558  562  517  513  532  474  512  567  524  541  589
  566  523  514  502  574  559  561  505  576  549  560  517  556  535  526  563  557  569  517  555  507  536  510  499  546  540  502
 504  547  565  549  548  529  502  552  546  587  589  552  563  527  540  541  563  506  514  512  545  564  532  511  542  551  524
532  569  539  537  525  468  538  509  529  573  573  514  505  530  507  551  548  493  518  560  524  538  564  514  487  545  513  5
06  513  527  520  517  537  531  488  497  566  539  528  536  516  549  511  550  552  503  545  514  541  529  537  514  554  511  57
5  546  486  542  552  526  536  533  577  570  541  521  517  506  557  547  523  554  559  554  544  521  512  533  544  529  542  529
  572  521  574  545  542  545  540  556  519  517  593  529  476  546  528  509  559  518  535  529  548  567  511  540  558  581  537
 504  540  498  517  546  557  570  514  515  512  537  560  525  554  554  554  533  550  529  551  541  561  511  510  544  495  519
537  552  567  537  496  538  533  519  479  556  535  511  502  527  553  540  546  548  548  509  585  540  510  530  569  524  529  5
30  514  517  535  534  561  547  536  552  538  552  546  567  551  548  516  578  526  512  497  520  502  558  528  548  532  529  53
1  536  535  535  580  534  528  524  561  484  536

Matrix Sum = 125231132

[thread_no]: 2 , [Time]: 122 ms
```

```
~/C/M/Multicore_assignment_JAVA_1  *  proj1/problem2  java MatmultD 2 < mat500.txt | head -n 2                    0 5.3s  16:52
Execution Time of thread 14 : 118 ms
Execution Time of thread 15 : 119 ms
```

4 threads

```
 500  567  560  473  545  538  466  550  502  570  541  513  555  492  505  555  538  499  480  495  519  507  509  481  539  505  500
512  544  507  516  516  501  503  506  534  511  547  520  470  501  498  515  531  487  485  512  496  547  530  508  478  547  522  5
34  488  547  505  531  532  497  504  500  568  518  495  526  497  525  507  517  519  441  518  496  499  500  525  483  499  515  56
3  532  502  493  531  538  536  476  558  557  515  497  500  478  554  523  525  494  524  567  538  511  543  501  513  517  523  508
  514  470  567  531  519  520  504  536  497  529  574  486  476  481  516  515  527  511  542  495  483  521  516  560  533  549  534
 488  521  528  486  546  555  509  529  503  505  515  528  544  517  543  548  475  531  544  527  559  525  485  513  525  498  489
519  505  529  535  511  492  500  500  521  506  535  508  503  511  539  489  540  559  545  505  555  547  495  521  547  567  541  4
88  492  504  531  493  530  535  508  480  531  526  567  534  510  516  540  536  508  523  485  499  497  548  523  519  524  505  52
7  526  537  532  558  512  507  533  565  504  529
 505  525  565  535  457  517  543  530  558  547  515  547  511  586  543  545  486  532  520  554  517  564  564  540  517  544  513
521  562  583  531  498  511  517  503  526  543  549  542  511  532  502  592  531  541  527  565  537  515  496  569  552  547  521  5
10  525  510  539  537  544  521  528  506  471  503  512  518  549  556  549  528  547  560  526  536  558  534  553  514  536  513  54
1  520  572  517  522  559  532  554  515  527  532  537  526  555  543  522  533  573  544  530  521  548  551  507  525  582  578  500
  524  521  560  505  536  568  541  526  515  548  517  534  566  535  514  512  555  541  502  531  526  571  511  515  545  506  532
 498  572  544  528  577  551  537  540  553  530  519  559  494  551  525  528  528  609  519  518  504  588  530  489  501  541  581
517  540  552  588  512  549  565  543  533  537  521  498  524  550  563  525  526  566  577  510  506  552  530  544  517  492  525  4
80  523  550  560  559  546  535  512  533  544  541  478  553  531  507  548  559  540  510  500  564  589  539  516  498  503  565  50
7  530  536  529  599  589  490  514  549  517  577  505  501  534  534  561  493  558  562  517  513  532  474  512  567  524  541  589
  566  523  514  502  574  559  561  505  576  549  560  517  556  535  526  563  557  569  517  555  507  536  510  499  546  540  502
 504  547  565  549  548  529  502  552  546  587  589  552  563  527  540  541  563  506  514  512  545  564  532  511  542  551  524
532  569  539  537  525  468  538  509  529  573  573  514  505  530  507  551  548  493  518  560  524  538  564  514  487  545  513  5
06  513  527  520  517  537  531  488  497  566  539  528  536  516  549  511  550  552  503  545  514  541  529  537  514  554  511  57
5  546  486  542  552  526  536  533  577  570  541  521  517  506  557  547  523  554  559  554  544  521  512  533  544  529  542  529
  572  521  574  545  542  545  540  556  519  517  593  529  476  546  528  509  559  518  535  529  548  567  511  540  558  581  537
 504  540  498  517  546  557  570  514  515  512  537  560  525  554  554  554  533  550  529  551  541  561  511  510  544  495  519
537  552  567  537  496  538  533  519  479  556  535  511  502  527  553  540  546  548  548  509  585  540  510  530  569  524  529  5
30  514  517  535  534  561  547  536  552  538  552  546  567  551  548  516  578  526  512  497  520  502  558  528  548  532  529  53
1  536  535  535  580  534  528  524  561  484  536

Matrix Sum = 125231132

[thread_no]: 4 , [Time]:  89 ms
```

```
~/C/M/Multicore_assignment_JAVA_1  *  proj1/problem2  java MatmultD 4 < mat500.txt | head -n 4                    0 1639ms  16:51
Execution Time of thread 16 : 88 ms
Execution Time of thread 15 : 88 ms
Execution Time of thread 14 : 89 ms
Execution Time of thread 17 : 88 ms
```