**MINI PROJECT**

Write a program to implement matrix multiplication. Also implement multithreaded matrix multiplication with either one thread per row or one thread per cell. Analyze and compare their performance.

public class MatrixMultiplication {

// Standard matrix multiplication

public static int[][] multiply(int[][] A, int[][] B) {

int rowsA = A.length;

int colsA = A[0].length;

int colsB = B[0].length;

int[][] C = new int[rowsA][colsB];

for (int i = 0; i < rowsA; i++) {

for (int j = 0; j < colsB; j++) {

for (int k = 0; k < colsA; k++) {

C[i][j] += A[i][k] \* B[k][j];

}

}

}

return C;

}

// Multithreaded matrix multiplication

static class MatrixRowMultiplier extends Thread {

private int row;

private int[][] A;

private int[][] B;

private int[][] C;

MatrixRowMultiplier(int row, int[][] A, int[][] B, int[][] C) {

this.row = row;

this.A = A;

this.B = B;

this.C = C;

}

public void run() {

int colsB = B[0].length;

for (int j = 0; j < colsB; j++) {

for (int k = 0; k < A[0].length; k++) {

C[row][j] += A[row][k] \* B[k][j];

}

}

}

}

public static int[][] multiplyWithThreads(int[][] A, int[][] B) throws InterruptedException {

int rowsA = A.length;

int colsB = B[0].length;

int[][] C = new int[rowsA][colsB];

Thread[] threads = new Thread[rowsA];

for (int i = 0; i < rowsA; i++) {

threads[i] = new MatrixRowMultiplier(i, A, B, C);

threads[i].start();

}

for (int i = 0; i < rowsA; i++) {

threads[i].join();

}

return C;

}

// Method to print the matrix

private static void printMatrix(int[][] matrix) {

for (int[] row : matrix) {

for (int val : row) {

System.out.print(val + " ");

}

System.out.println();

}

}

public static void main(String[] args) throws InterruptedException {

int[][] A = {

{1, 2, 3},

{4, 5, 6}

};

int[][] B = {

{7, 8},

{9, 10},

{11, 12}

};

// Standard multiplication

long startTime = System.nanoTime();

int[][] C1 = multiply(A, B);

long endTime = System.nanoTime();

System.out.println("Standard Matrix Multiplication Result:");

printMatrix(C1);

System.out.println("Time taken (Standard): " + (endTime - startTime) + " ns");

// Multithreaded multiplication

startTime = System.nanoTime();

int[][] C2 = multiplyWithThreads(A, B);

endTime = System.nanoTime();

System.out.println("Multithreaded Matrix Multiplication Result:");

printMatrix(C2);

System.out.println("Time taken (Multithreaded): " + (endTime - startTime) + " ns");

}

}

**OUTPUT:**

Standard Matrix Multiplication Result:

58 64

139 154

Time taken (Standard): 5020 ns

Multithreaded Matrix Multiplication Result:

58 64

139 154

Time taken (Multithreaded): 2631860 ns