**CSE2008 (Operating Systems) Lab-4**

**Khan Mohd. Owais Raza**
**20BCD7138**

2D Convolution - consider a matrix of size 4096 x 4096 and mask of size 9x9. Compare performance improvement over non multithreaded 2D convolution program.

```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138 */
/* CSE2008 (Operating Systems) Lab Practical */
/* Consider matrix of size 4096x4096 and mask of size 9x9.
   Compare performance improvement over non-multithreaded
   2D convolution program
 */
package owaisraza_20bcd7138;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;
class MyClass {
int ROW1;
int COL1;
int ROW2;
int COL2;
int A[][];
int B[][];
int C[][];
int D[][];
MyClass() {
ROW1 = 2048;
COL1 = 2048;
ROW2 = 2048;
COL2 = 2048;
A = new int[ROW1][COL1];
B = new int[ROW2][COL2];
C = new int[ROW1][COL2];
D = new int[ROW1][COL2];
}
public static void main(String args[]) {
MyClass ob = new MyClass();
int c = 0;
for (int i = 0; i < ob.ROW1; i++) {
for (int j = 0; j < ob.COL1; j++) {
ob.A[i][j] = c++;
}}
c = 0;
for (int i = 0; i < ob.ROW2; i++) {
for (int j = 0; j < ob.COL2; j++) {
ob.B[i][j] = c++;
}}
ob.MyClass2();
}
public void MyClass2() {
long startTime = System.currentTimeMillis();
matrixMultiply();
long stopTime = System.currentTimeMillis();
long elapsedTime = stopTime - startTime;
```

```
System.out.println(" \nEXECUTION TIME = " + elapsedTime + " ms\n");
try {
ExecutorService executor = Executors.newFixedThreadPool(this.COL2);
startTime = System.currentTimeMillis();
for (int i = 0; i < ROW1; i++) {
for (int j = 0; j < COL2; j++) {
RunnableClass ob = new RunnableClass(i, j, this);
executor.execute(ob);
}}
executor.shutdown();
while (!executor.isTerminated()) {
}
executor.awaitTermination(Long.MAX_VALUE, TimeUnit.NANOSECONDS);
stopTime = System.currentTimeMillis();
elapsedTime = stopTime - startTime;
System.out.println(" \nEXECUTION TIME (USING MULTI-THREADING) = " + elapsedTime +
" ms\n");
} catch (Exception e) {
}}
void matrixMultiply() {
for (int i = 0; i < ROW1; i++) {
for (int j = 0; j < COL2; j++) {
for (int k = 0; k < ROW2; k++) {
C[i][j] += A[i][k] * B[k][j];
}}
}}
void printMatrix(int ar[][]) {
int row = ar.length;
int col = ar[0].length;
for (int i = 0; i < row; i++) {
for (int j = 0; j < col; j++) {
System.out.print(" " + ar[i][j]);
}
System.out.print("\n");
}}
}
class RunnableClass implements Runnable {
int i, j;
MyClass ob;
RunnableClass(int ii, int jj, MyClass ob1) {
i = ii;
j = jj;
ob = ob1;
}
public void run() {
int sum = 0;
for (int k = 0; k < ob.ROW2; k++) {
sum += ob.A[i][k] * ob.B[k][j];
}
ob.D[i][j] = sum;
}}
```

```
<terminated> MyClass [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe

EXECUTION TIME = 53085 ms


EXECUTION TIME (USING MULTI-THREADING) = 30689 ms
```

Matrix matrix multiplication. Assume square matrices of size 4096x4096. Compare performance improvement over non multithreaded matrix matrix multiplication program.

```
/* KHAN MOHD OWAIS RAZA */
/* 20BCD7138 */
/* CSE2008 (Operating Systems) Lab Practical */
/* Write a multi-threaded java program for matrix multiplication.
   Assume square matrices of size 4096x4096. Compare performance
   improvement over non-multithreaded matrix multiplication program
 */
package owaisraza_20bcd7138;
import java.io.*;
class MyClass {
static void printMatrix(int M[][],
int ROW_SIZE, int COLUMN_SIZE)
{
for (int i = 0; i < ROW_SIZE; i++) {
for (int j = 0; j < COLUMN_SIZE; j++)
System.out.print(M[i][j] + " ");
System.out.println();
}}
static void multiplyMatrix(
int ROW1, int COL1, int A[][],
int ROW2, int COL2, int B[][])
{
int i, j, k;
System.out.println("\nMatrix A =");
printMatrix(A, ROW1, COL1);
System.out.println("\nMatrix B =");
printMatrix(B, ROW2, COL2);
if (ROW2 != COL1) {
System.out.println();
return;
}
int C[][] = new int[ROW1][COL2];
for (i = 0; i < ROW1; i++) {
for (j = 0; j < COL2; j++) {
for (k = 0; k < ROW2; k++)
C[i][j] += A[i][k] * B[k][j];
}
}
System.out.println("\nResultant Matrix =");
printMatrix(C, ROW1, COL2);
}
public static void main(String[] args)
{
int ROW1 = 4, COL1 = 4, ROW2 = 4, COL2 = 4;
int A[][] = { { 1, 1, 1, 3},
{ 2, 2, 2, 8 },
{ 3, 3, 3, 7},
{ 3,2,1,5 }
};
int B[][] = { { 1, 1, 1, 1 },
{ 2, 2, 2, 2 },
{ 3, 3, 3, 3 },
{ 7, 7, 6, 4 } };
multiplyMatrix(ROW1, COL1, A, ROW2, COL2, B);
}}
```

```
<terminated> MyClass [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe

Matrix A =
1 1 1 3
2 2 2 8
3 3 3 7
3 2 1 5

Matrix B =
1 1 1 1
2 2 2 2
3 3 3 3
7 7 6 4

Resultant Matrix =
27 27 24 18
68 68 60 44
67 67 60 46
45 45 40 30
```