Lab#07

**Objective**: To understand Multithreading.

**Task:** Write the same code for matrix multiplication and divide the code into ten threads. Also note the timestamp at the start and end of the program. Give your conclusion.

**Code for Matrix multiplication:**

import java.lang.\*;

public class MatrixMultiplication{

public static void main(String args[]){

// returns the current time in milliseconds

long start\_time = System.nanoTime();

System.out.print("Current Time in nanoseconds = ");

// float startTime= System.currentTimeMillis();

System.out.println(start\_time);

//creating two matrices

int a[][]={{1,1,1},{2,2,2},{3,3,3}};

int b[][]={{1,1,1},{2,2,2},{3,3,3}};

//creating another matrix to store the multiplication of two matrices

int c[][]=new int[3][3];

//3 rows and 3 columns

System.out.println("Resultant matrix after multiplication is: ");

//multiplying and printing multiplication of 2 matrices

for(int i=0;i<3;i++){

for(int j=0;j<3;j++){

c[i][j]=0;

for(int k=0;k<3;k++)

{

c[i][j]+=a[i][k]\*b[k][j];

}

//end of k loop

System.out.print(c[i][j]+" ");

//printing matrix element

}

//end of j loop

System.out.println();

}

// returns the current time in milliseconds

System.out.print("Current Time in nanoSeconds = ");

// float endTime= System.currentTimeMillis();

//float totalTime= endTime-startTime;

long end\_time = System.nanoTime();

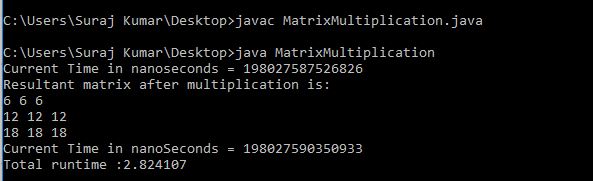
System.out.println(end\_time);

double totalTime = (end\_time - start\_time) / 1e6;

System.out.println("Total runtime :" +totalTime);

}}

**Output:**



**Same Code but divided in 10 threads.**

public class MMParentTh {

//Dividing in threads.

public static final int NUM\_OF\_THREADS = 9;

public static void main(String args[])

{

long start\_time = System.nanoTime();

System.out.print("Current Time in nanoseconds = ");

// float startTime= System.currentTimeMillis();

System.out.println(start\_time);

int row;

int col;

int A[][] = {{1,1,1},{2,2,2},{3,3,3}};

int B[][] = {{1,1,1},{2,2,2},{3,3,3}};

int C[][] = new int[3][3];

int threadcount = 0;

Thread[] thrd = new Thread[NUM\_OF\_THREADS];

try

{

for(row = 0 ; row < 3; row++)

{

for (col = 0 ; col < 3; col++ )

{

// creating thread for multiplications

thrd[threadcount] = new Thread(new MMWorkerTh(row, col, A, B, C));

thrd[threadcount].start(); //thread start

thrd[threadcount].join(); // joining threads

threadcount++;

}

}

}

catch (InterruptedException ie){}

// printing matrix A

System.out.println(" A Matrix : ");

for(row = 0 ; row < 3; row++)

{

for (col = 0 ; col < 2; col++ )

{

System.out.print(" "+A[row][col]);

}

System.out.println();

}

// printing matrix B

System.out.println(" B Matrix : ");

for(row = 0 ; row < 2; row++)

{

for (col = 0 ; col < 3; col++ )

{

System.out.print(" "+B[row][col]);

}

System.out.println();

}

// printing resulting matrix C after multiplication

System.out.println(" Resulting C Matrix : ");

for(row = 0 ; row < 3; row++)

{

for (col = 0 ; col < 3; col++ )

{

System.out.print(" "+C[row][col]);

}

System.out.println();

}

System.out.print("Current Time in nanoSeconds = ");

long end\_time = System.nanoTime();

System.out.println(end\_time);

double totalTime = (end\_time - start\_time) / 1e6;

System.out.println("Total runtime :" +totalTime);

}

}

