Name : Nachiket N Jagtap
En.No.: 21221079

Assignment No. 6

Multithreaded and Distributed Algorithms

Multiplication of Matrix using threads

Java Code:

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;
import java.util.concurrent.Callable;

public class MatrixMultiplication {
public static void main(String[] args) {
int[][] matrixA = {{1, 2, 3}, {4, 5, 6}};
int[][] matrixB = {{7, 8}, {9, 10}, {11, 12}};

int numRowsA = matrixA.length;
int numColsA = matrixA[0].length;
int numRowsB = matrixB.length;
int numColsB = matrixB[0].length;

if (numColsA != numRowsB) {
System.out.println("Matrix multiplication is not possible.");
return;
}

int[][] result = new int[numRowsA][numColsB];

// Create a thread pool with a fixed number of threads (e.g., 4)
int numThreads = 4;
ExecutorService executor = Executors.newFixedThreadPool(numThreads);

// Perform matrix multiplication using threads
for (int i = 0; i < numRowsA; i++) {
for (int j = 0; j < numColsB; j++) {
Callable<Integer> task = new MatrixMultiplicationTask(matrixA, matrixB, result, i, j);
Future<Integer> future = executor.submit(task);
}
}

// Shutdown the executor
executor.shutdown();

// Print the result
for (int i = 0; i < numRowsA; i++) {
```

```
for (int j = 0; j < numColsB; j++) {
System.out.print(result[i][j] + " ");
}
System.out.println();
}
}
}

class MatrixMultiplicationTask implements Callable<Integer> {
private int[][] matrixA;
private int[][] matrixB;
private int[][] result;
private int row;
private int col;

public MatrixMultiplicationTask(int[][] matrixA, int[][] matrixB, int[][] result, int row, int col)
{
this.matrixA = matrixA;
this.matrixB = matrixB;
this.result = result;
this.row = row;
this.col = col;
}

@Override
public Integer call() {
int sum = 0;
for (int i = 0; i < matrixA[0].length; i++) {
sum += matrixA[row][i] * matrixB[i][col];
}
result[row][col] = sum;
return sum;
}
}
```

Output :