JAVA编程进阶上机报告

学 院 智能与计算

专 业 软件工程

班 级 一班

学 号 3018216063

姓 名 陈宇涛

一、实验要求

1. 编写矩阵随机生成类 MatrixGenerator 类，随机生成任意大小的矩阵，矩阵单元使用 double 存储。
2. 使用串行方式实现矩阵乘法。
3. 使用多线程方式实现矩阵乘法。
4. 比较串行和并行两种方式使用的时间，利用第三次使用中使用过的 jvm状态查看命令，分析产生时间差异的原因是什么。

二、源代码

runnable类 :实现Runnable接口，传入待计算矩阵、线程数、第几个线程

重写run函数使其计算矩阵相乘结果

计算矩阵相乘结果：将第一个矩阵按行分成多个部分（=线程数），每个线程由起始位置开始计算与第二个矩阵相乘（第i行、第j列等于第一个矩阵第i行每一列乘以第二个矩阵第j列每一行的和）并每隔线程数计算一次

```
package lab4;

public class runnable implements Runnable{
    double[][] Matrix1;//第一个矩阵
    double[][] Matrix2;//第二个矩阵
    double[][] Matrix;//存储结果
    int Start;//起始位置
    int Number;//线程数
    public runnable(double[][] matrix1,double[][] matrix2,int start,int number)
{
        //构造函数，获得所需值
```

```
        Matrix1 = matrix1;
        Matrix2 = matrix2;
        Start = start;
        Matrix = new double[matrix1.length][matrix2[0].length];
        Number = number;
    }
    public void run() {
        //重写run函数，计算矩阵相乘结果
          try {
             for(int i=Start;i<Matrix1.length;i+=Number) {
                 for(int j=0;j<Matrix2[0].length;j++) {
                     for(int k=0;k<Matrix1[0].length;k++) {
                         Matrix[i][j] += Matrix1[i][k] * Matrix2[k][j];
                     }
                 }

             }
          }catch (Exception e) {
               e.printStackTrace();
          }

       }
    }
}
```

MatrixGenerator函数：根据传入参数构造矩阵（size1：行数，size2：列数）

```
public static double[][] MatrixGenerator (int size1,int size2){
        double[][] matrix = new double[size1][size2];
        //每个数都用Random().nextDouble()生成0~1的double类型随机数
        for(int i=0;i<size1;i++) {
            for(int j=0;j<size2;j++) {
                matrix[i][j] = new Random().nextDouble();
            }
        }
        return matrix;
    }
```

serial函数：根据传入矩阵串行计算矩阵相乘结果

```
public static double[][] serial(double[][] Matrix1,double[][] Matrix2){
        double[][] matrix = new double[Matrix1.length][Matrix2[0].length];
    //存储结果

        for (int i = 0; i < Matrix1.length; i++) {
            for (int j = 0; j < Matrix2[0].length; j++) {
                for (int k = 0; k < Matrix1[0].length; k++)
                    matrix[i][j] += Matrix1[i][k] * Matrix2[k][j];
                //第i行、第j列等于第一个矩阵第i行每一列乘以第二个矩阵第j列每一行的和
            }
        }
        return matrix;
    }
```

multiThread1函数：根据传入矩阵用两个线程计算矩阵相乘结果

```
public static double[][] multiThread1(double[][] Matrix1,double[][] Matrix2){
        double[][] matrix = new double[Matrix1.length][Matrix2[0].length];
        runnable r1 = new runnable(Matrix1,Matrix2,0,2);
        runnable r2 = new runnable(Matrix1,Matrix2,1,2);
        r1.run();
        r2.run();
        for(int i=0;i<Matrix1.length;i++) {
            for(int j=0;j<Matrix2[0].length;j++) {
                //将线程计算结果写入matrix中
                if(i%2==0) {
                    matrix[i][j] = r1.Matrix[i][j];
                }else {
                    matrix[i][j] = r2.Matrix[i][j];
                }

            }
        }
        return matrix;
    }
```

multiThread2函数：根据传入矩阵用三个线程计算矩阵相乘结果

```
public static double[][] multiThread2(double[][] Matrix1,double[][] Matrix2){
        double[][] matrix = new double[Matrix1.length][Matrix2[0].length];
        runnable r1 = new runnable(Matrix1,Matrix2,0,3);
        runnable r2 = new runnable(Matrix1,Matrix2,1,3);
        runnable r3 = new runnable(Matrix1,Matrix2,2,3);
        r1.run();
        r2.run();
        r3.run();
        for(int i=0;i<Matrix1.length;i++) {
            for(int j=0;j<Matrix2[0].length;j++) {
                //将线程计算结果写入matrix中
                if(i%3==0) {
                    matrix[i][j] = r1.Matrix[i][j];
                }else if(i%3==1){
                    matrix[i][j] = r2.Matrix[i][j];
                }else {
                    matrix[i][j] = r3.Matrix[i][j];
                }

            }
        }
        return matrix;
    }
```

multiThread3函数：根据传入矩阵用四个线程计算矩阵相乘结果

```
public static double[][] multiThread3(double[][] Matrix1,double[][] Matrix2){
        double[][] matrix = new double[Matrix1.length][Matrix2[0].length];
        runnable r1 = new runnable(Matrix1,Matrix2,0,4);
        runnable r2 = new runnable(Matrix1,Matrix2,1,4);
        runnable r3 = new runnable(Matrix1,Matrix2,2,4);
        runnable r4 = new runnable(Matrix1,Matrix2,3,4);
        r1.run();
        r2.run();
```

```
        r3.run();
        r4.run();
        for(int i=0;i<Matrix1.length;i++) {
            for(int j=0;j<Matrix2[0].length;j++) {
                //将线程计算结果写入matrix中
                if(i%4==0) {
                    matrix[i][j] = r1.Matrix[i][j];
                }else if(i%4==1){
                    matrix[i][j] = r2.Matrix[i][j];
                }else if(i%4==2){
                    matrix[i][j] = r3.Matrix[i][j];
                }else {
                    matrix[i][j] = r4.Matrix[i][j];
                }

            }
        }
        return matrix;
    }
```

check函数：使用断言判断两个矩阵是否相同

```
public static void check(double[][] Matrix1,double[][] Matrix2) {
        for(int i=0;i<Matrix1.length;i++) {
            for(int j=0;j<Matrix2[0].length;j++) {
                assert Matrix1[i][j] == Matrix2[i][j] : "The anwser is false";

            }
        }

        System.out.println("The anwser is true");
    }
```

main函数：产生三个不同范围随机数来用MatrixGenerator函数产生两个待乘矩阵，分别用串行、两个线程、三个线程、四个线程计算矩阵相乘结果，计算计算时间，并用check函数检查计算结果是否正确

```
public static void main(String args[]) {

        for(int i=0;i<=9;i++) {
            int randInt1 = new Random().nextInt(100) + i*100;
            int randInt2 = new Random().nextInt(100) + i*100;
            int randInt3 = new Random().nextInt(100) + i*100;
            System.out.println(randInt1+" "+randInt2+" "+randInt3);
            double[][] matrix1 = new double[randInt1][randInt2];
            double[][] matrix2 = new double[randInt2][randInt3];
            double[][] matrix3 = new double[randInt1][randInt3];
            double[][] matrix4 = new double[randInt1][randInt3];
            double[][] matrix5 = new double[randInt1][randInt3];
            double[][] matrix6 = new double[randInt1][randInt3];
            matrix1 = MatrixGenerator (randInt1,randInt2);
            matrix2 = MatrixGenerator (randInt2,randInt3);
            double startTime1 = System.currentTimeMillis();
            matrix3 = serial(matrix1,matrix2);
            double endTime1 = System.currentTimeMillis();
            System.out.println("串行计算时间:" + (endTime1 - startTime1));
```

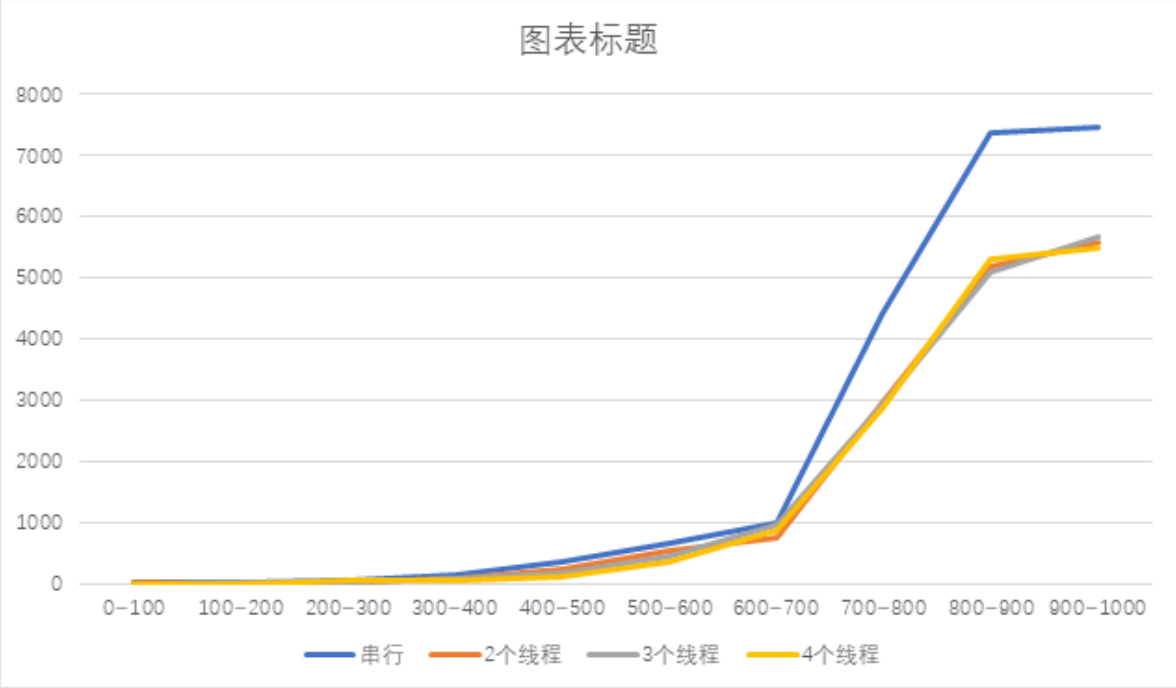
```
            double startTime2 = System.currentTimeMillis();
            matrix4 = multiThread1(matrix1,matrix2);
            double endTime2 = System.currentTimeMillis();
            System.out.println("2线程并行计算时间:" + (endTime2 - startTime2));
            check(matrix3,matrix4);
            double startTime3 = System.currentTimeMillis();
            matrix5 = multiThread1(matrix1,matrix2);
            double endTime3 = System.currentTimeMillis();
            System.out.println("3线程并行计算时间:" + (endTime3 - startTime3));
            check(matrix3,matrix5);
            double startTime4 = System.currentTimeMillis();
            matrix6 = multiThread1(matrix1,matrix2);
            double endTime4 = System.currentTimeMillis();
            System.out.println("4线程并行计算时间:" + (endTime4 - startTime4));
            check(matrix3,matrix6);
        }
    }
```

三、实验结果

```
52 26 87
串行计算时间:16.0
2线程并行计算时间:15.0
The anwser is true
3线程并行计算时间:0.0
The anwser is true
4线程并行计算时间:0.0
The anwser is true
134 193 108
串行计算时间:16.0
2线程并行计算时间:0.0
The anwser is true
3线程并行计算时间:15.0
The anwser is true
4线程并行计算时间:7.0
The anwser is true
290 271 259
串行计算时间:63.0
2线程并行计算时间:38.0
The anwser is true
3线程并行计算时间:31.0
The anwser is true
4线程并行计算时间:53.0
The anwser is true
366 377 375
串行计算时间:131.0
2线程并行计算时间:85.0
The anwser is true
3线程并行计算时间:85.0
The anwser is true
4线程并行计算时间:69.0
The anwser is true
```

```
464 451 458
串行计算时间:354.0
2线程并行计算时间:232.0
The anwser is true
3线程并行计算时间:162.0
The anwser is true
4线程并行计算时间:123.0
The anwser is true
563 566 552
串行计算时间:650.0
2线程并行计算时间:556.0
The anwser is true
3线程并行计算时间:464.0
The anwser is true
4线程并行计算时间:356.0
The anwser is true
666 696 632
串行计算时间:992.0
2线程并行计算时间:757.0
The anwser is true
3线程并行计算时间:963.0
The anwser is true
4线程并行计算时间:890.0
The anwser is true
763 766 767
串行计算时间:4432.0
2线程并行计算时间:2978.0
The anwser is true
3线程并行计算时间:2941.0
The anwser is true
4线程并行计算时间:2868.0
The anwser is true
```

```
879 865 843
串行计算时间:7377.0
2线程并行计算时间:5184.0
The anwser is true
3线程并行计算时间:5097.0
The anwser is true
4线程并行计算时间:5296.0
The anwser is true
955 982 980
串行计算时间:7471.0
2线程并行计算时间:5596.0
The anwser is true
3线程并行计算时间:5680.0
The anwser is true
4线程并行计算时间:5483.0
The anwser is true
```



分析：在矩阵边长<400时，串行计算与多线程计算时间差不多；在矩阵边长>400时，串行计算所用时间远远大于多线程计算时间

可能原因：计算量较少时，多线程进行矩阵乘法计算时创建线程消耗的时间大于计算时间，导致计算效率与串行计算差不多甚至小于串行计算，且线程数越多计算效率越低；在计算量较大时，多线程进行矩阵乘法计算时创建线程消耗的时间远远小于计算时间，导致计算效率大于串行计算，且线程数越多，时间越短