JAVA 编程进阶上机报告

学　院　智能与计算学部

专　业　软件工程

班　级　二班

学　号　3018216084

姓　名　田家硕

一、实验要求

要求

• 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。

• 使用串行方式实现矩阵乘法。

• 使用多线程方式实现矩阵乘法。

• 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

MatrixGenerator.java

```
import java.util.Random;

public class MatrixGenerator {
    //随机生成任意大小的矩阵，矩阵单元使用 double 存储。
    public static double[][] matrix()
    {
        Random r = new Random();
        double[][] randomMatrix = new double[r.nextInt(1000)]
[r.nextInt(1000)];
        for(int i=0;i<randomMatrix.length;i++)
        {
            for(int j=0;j<randomMatrix[0].length;j++)
            {
                double ran1 = r.nextDouble()*1000;
                randomMatrix[i][j] = ran1;
            }
        }
        return randomMatrix;
```

```
    }

    public static double[][] matrix(int column)
    {
        Random r = new Random();
        double[][] randomMatrix = new double[column][r.nextInt(100)];
        for(int i=0;i<randomMatrix.length;i++)
        {
            for(int j=0;j<randomMatrix[0].length;j++)
            {
                double ran1 = r.nextDouble()*1000;
                randomMatrix[i][j] = ran1;
            }
        }
        return randomMatrix;
    }
}
```

Parallel.java

```
import java.util.concurrent.CountDownLatch;
public class Parallel extends Thread {
    private double[][] A;
    private double[][] B;
    private int index;
    private int gap;
    private double[][] result;
    private CountDownLatch countDownLatch;

    public Parallel(double[][] A, double[][] B, int index, int gap,
double[][] result, CountDownLatch countDownLatch) {
        this.A = A;
        this.B = B;
        this.index = index;
        this.gap = gap;
        this.result = result;
        this.countDownLatch = countDownLatch;
    }
    public static void parallel (double[][] A, double[][] B)throws
InterruptedException {
```

```
        long startTime;
        long endTime;
        int threadNum = 2;
        CountDownLatch countDownLatch = new CountDownLatch(threadNum);

        double[][] parallel_result = new double[A.length][B[0].length];

        int gap = A.length / (threadNum-1);
        Parallel parallel1 = new Parallel(A, B, 0, gap, parallel_result,
countDownLatch);
        Parallel parallel2 = new Parallel(A, B, 1, gap, parallel_result,
countDownLatch);
        Parallel parallel3 = new Parallel(A, B, 2, gap, parallel_result,
countDownLatch);
        Parallel parallel4 = new Parallel(A, B, 3, gap, parallel_result,
countDownLatch);
        startTime = System.currentTimeMillis();
        parallel1.start();
        parallel2.start();
        parallel3.start();
        parallel4.start();
        parallel1.join();
        parallel2.join();
        parallel3.join();
        parallel4.join();
        endTime = System.currentTimeMillis();

        System.out.println("并行计算开始时刻:" + (startTime));
        System.out.println("并行计算结束时刻:" + (endTime));
        System.out.println("并行计算运行时间:" + (endTime - startTime));
    }
    public void run() {
        for (int i = index * gap; i < Math.min((index + 1)*gap,A.length) ;
i++)
            for (int j = 0; j < B[0].length; j++) {
                for (int k = 0; k < B.length; k++)
                    result[i][j] += A[i][k] * B[k][j];
            }
        countDownLatch.countDown();
    }
```

```
}
```

Serial.java

```
import com.sun.org.apache.bcel.internal.generic.ARETURN;
public class Serial {
    public static void serial(double[][] A, double[][] B){
        double[][] result = new double[A.length][B[0].length];
        long startTime = System.currentTimeMillis();
        for (int i = 0; i < A.length; i++) {
            for (int j = 0; j < B[0].length; j++) {
                for (int k = 0; k < A[0].length; k++) {
                    result[i][j] += A[i][k]*B[k][j];
                }
            }
        }
        long endTime = System.currentTimeMillis();
        System.out.println("串行计算开始时刻:" + (startTime));
        System.out.println("串行计算结束时刻:" + (endTime));
        System.out.println("串行计算运行时间:" + (endTime - startTime));
    }
}
```

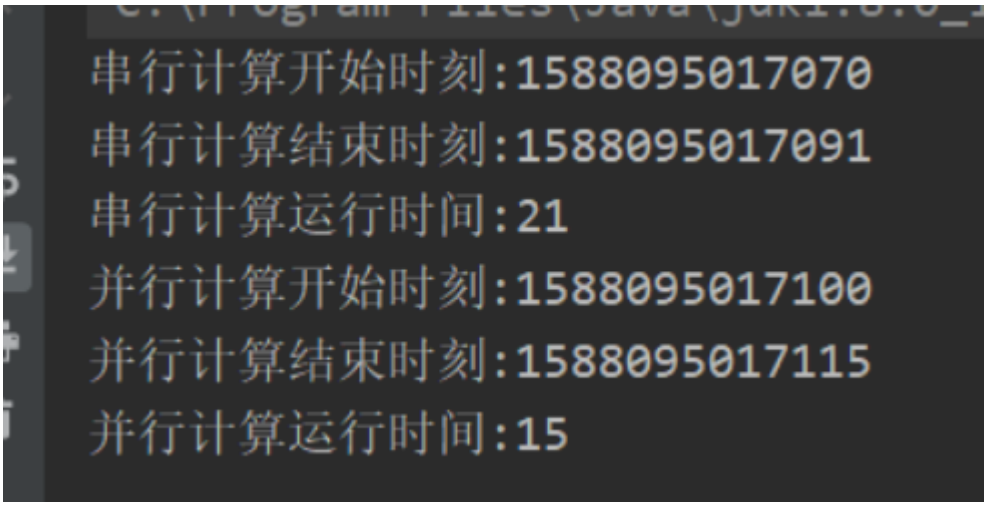
demo.java

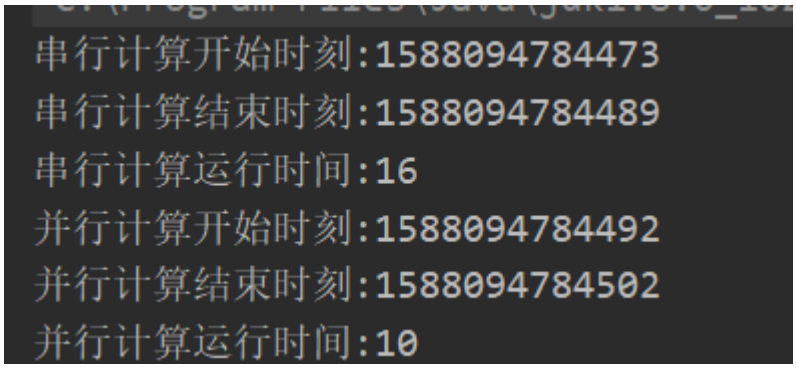
```
import java.util.concurrent.CountDownLatch;

public class demo {
    public static void main(String[] args) throws InterruptedException {
        double[][] A = MatrixGenerator.matrix();
        double[][] B = MatrixGenerator.matrix(A[0].length);
        Serial.serial(A,B);
        Parallel.parallel(A,B);
    }
}
```

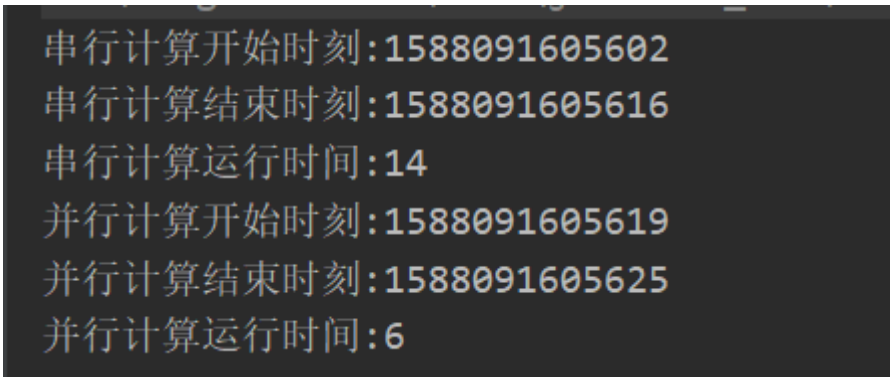
串行计算直接使用三重循环嵌套。

并行计算使用分块的方法，将矩阵分成线程数量个块，然后分别开线程计算，最后使用join方法检测线程结束。

三、实验结果


串行计算开始时刻:1588095017070
串行计算结束时刻:1588095017091
串行计算运行时间:21
并行计算开始时刻:1588095017100
并行计算结束时刻:1588095017115
并行计算运行时间:15


串行计算开始时刻:1588094784473
串行计算结束时刻:1588094784489
串行计算运行时间:16
并行计算开始时刻:1588094784492
并行计算结束时刻:1588094784502
并行计算运行时间:10


串行计算开始时刻:1588091605602
串行计算结束时刻:1588091605616
串行计算运行时间:14
并行计算开始时刻:1588091605619
并行计算结束时刻:1588091605625
并行计算运行时间:6