

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC LẠC HỒNG



ĐÀO QUỐC PHƯƠNG

BÁO CÁO MÔN HỌC
SONG SONG THUẬT TOÁN VÀ ỨNG DỤNG

ĐỀ TÀI
TÍNH TOÁN ƯỚC LƯỢNG SỐ PI

Đồng Nai – Năm 2024

MỤC LỤC

1. Mô tả bài toán.....	2
2. Thiết kế giải thuật tuần tự.....	2
3. Thiết kế giải thuật song song	2
3.1. Mô hình bộ nhớ chia sẻ	2
3.2. Mô hình bộ nhớ phân tán	4
4. Thực nghiệm đánh giá: độ chính xác và thời gian	4
4.1. Giải thuật tuần tự.....	4
4.2. Giải thuật song song.....	4
4.2.1. Mô hình bộ nhớ chia sẻ.....	4
4.2.2. Mô hình bộ nhớ phân tán	5

1. Mô tả bài toán

Công thức tính số pi

$$\pi = \int_0^1 \frac{4.0}{(1+x^2)} dx$$

2. Thiết kế giải thuật tuần tự

File *PiSequence.java* (<https://github.com/daoquocphuong/PiEstimate/blob/main/PiSequence.java>)

```

1 public class PiSequence {
2
3     public static void main(String[] args) {
4         long numSteps = 100000000;
5
6         long startTime = System.currentTimeMillis();
7
8         double total = 0.0;
9         for (long i = 0; i < numSteps; ++i) {
10             double x = (i + 0.5) / numSteps;
11             total += 4.0 / (1.0 + x * x);
12         }
13         total /= numSteps;
14
15         long stopTime = System.currentTimeMillis();
16         System.out.println("==> pi = " + total);
17         System.out.println("Calculation took " + (stopTime - startTime) + "ms"
18     }
19 }

```

3. Thiết kế giải thuật song song

3.1. Mô hình bộ nhớ chia sẻ

File *PiParallel.java* (<https://github.com/daoquocphuong/PiEstimate/blob/main/PiParallel.java>)

```

1 import java.util.ArrayList;
2
3
4 public class PiParallel extends Thread {
5
6     private long begin;
7     private long end;
8     private long numSteps;
9     private double subtotal;
10
11     public PiParallel(long begin, long end, long numSteps) {
12         this.begin = begin;
13         this.end = end;
14         this.numSteps = numSteps;
15     }
16
17     public double getSubtotal() {
18         return this.subtotal;
19     }
20
21     public void run() {
22         this.subtotal = 0.0;
23         for (long i = this.begin; i < this.end; ++i) {
24             double x = (i + 0.5) / this.numSteps;
25             this.subtotal += 4.0 / (1.0 + x * x);
26         }
27         this.subtotal /= this.numSteps;
28     }
29
30     public static void main(String[] args) {
31         int numThreads = Runtime.getRuntime().availableProcessors();
32         long numSteps = 100000000;
33         if (args.length > 1) {
34             numThreads = Integer.parseInt(args[1]);
35         }
36         if (args.length > 0) {
37             numSteps = Long.parseLong(args[0]);
38         }
39
40         System.out.println("Calculating pi using " + numThreads + " threads in " + numSteps + " steps...");
41
42         long startTime = System.currentTimeMillis();
43
44         ArrayList<PiParallel> threads = new ArrayList<PiParallel>();
45         long begin, end = 0;
46         for (int i = 0; i < numThreads; ++i) {
47             begin = end;
48             end = (i + 1) * numSteps / numThreads;
49             threads.add(new PiParallel(begin, end, numSteps));
50             threads.get(i).start();
51         }
52
53         double total = 0.0;
54         for (int i = 0; i < numThreads; ++i) {
55             try {
56                 threads.get(i).join();
57             } catch (Exception e) {
58                 System.err.println("Exception caught: " + e);
59             }
60             total += threads.get(i).getSubtotal();
61         }
62
63         long stopTime = System.currentTimeMillis();
64
65         System.out.println("==> pi = " + total);
66         System.out.println("Calculation took " + (stopTime - startTime) + "ms");
67     }
68 }

```

3.2. Mô hình bộ nhớ phân tán

File *PiMPI.java* (<https://github.com/daoquocphuong/PiEstimate/blob/main/PiMPI.java>)

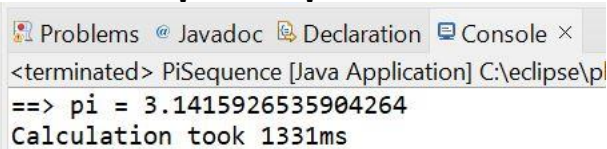
```

1 import mpi.MPI;
2
3 public class PiMPI {
4
5     public static void main(String[] args) {
6         MPI.Init(args);
7         long startTime = System.currentTimeMillis();
8
9         int rank = MPI.COMM_WORLD.Rank();
10        int size = MPI.COMM_WORLD.Size();
11        int nint = 100000000; // Intervals.
12        double h = 1.0/(double)nint, sum = 0.0;
13
14        for (int i=rank+1; i<=nint; i+=size) {
15            double x = h * ((double)i - 0.5);
16            sum += (4.0 / (1.0 + x * x));
17        }
18
19        double sBuf[] = { h * sum };
20        double rBuf[] = new double[1];
21
22        MPI.COMM_WORLD.Reduce(sBuf, 0, rBuf, 0, 1, MPI.DOUBLE, MPI.SUM, 0);
23
24        if (rank == 0) {
25            System.out.println("PI: " + rBuf[0]);
26        }
27
28        MPI.Finalize();
29
30        long stopTime = System.currentTimeMillis();
31        System.out.println("Rank [" + rank + "] Time Duration: " + (stopTime - startTime) + "ms");
32    }
33 }
34

```

4. Thực nghiệm đánh giá: độ chính xác và thời gian

4.1. Giải thuật tuần tự



```

Problems @ Javadoc Declaration Console ×
<terminated> PiSequence [Java Application] C:\eclipse\p
==> pi = 3.1415926535904264
Calculation took 1331ms

```

Giải thuật tuần tự với thời gian **1331ms** cho 100.000.000 bước

4.2. Giải thuật song song

4.2.1. Mô hình bộ nhớ chia sẻ



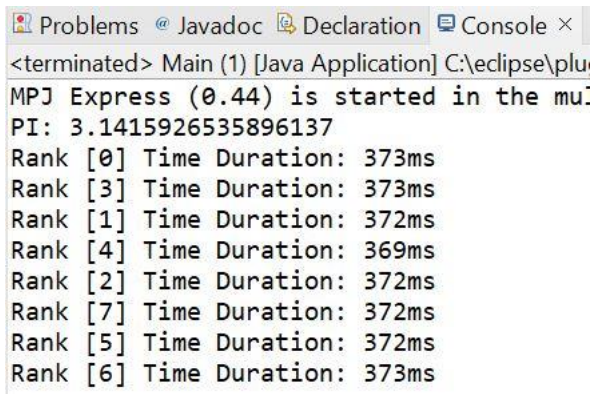
```

Problems @ Javadoc Declaration Console ×
<terminated> PiParallel [Java Application] C:\eclipse\plugins\org.eclipse.justj.
Calculating pi using 8 threads in 100000000 steps...
==> pi = 3.1415926535898153
Calculation took 330ms

```

Giải thuật song song bộ nhớ chia sẻ với 8 threads và kết quả thời gian chạy là **330ms** cho 100.000.000 bước

4.2.2. Mô hình bộ nhớ phân tán



```

Problems @ Javadoc Declaration Console ×
<terminated> Main (1) [Java Application] C:\eclipse\plu
MPJ Express (0.44) is started in the mu.
PI: 3.1415926535896137
Rank [0] Time Duration: 373ms
Rank [3] Time Duration: 373ms
Rank [1] Time Duration: 372ms
Rank [4] Time Duration: 369ms
Rank [2] Time Duration: 372ms
Rank [7] Time Duration: 372ms
Rank [5] Time Duration: 372ms
Rank [6] Time Duration: 373ms

```

Giải thuật song song bộ nhớ chia sẻ với 8 multi cores và kết quả thời gian **373ms** cho 100.000.000 bước

5. Tài liệu báo cáo