



西安电子科技大学  
XIDIAN UNIVERSITY

软件学院  
School of Software

# 并行计算

## 课程实验报告

实验名称：Java 并程序序设计

任课教师：徐悦牲

课程班级：15 级 云计算方向

学号姓名：15130130273 石明皓

提交日期：2018 年 5 月 23 日

# 课程实验报告

## 一、实验名称

第 2 次实验：Java 并行程序设计

## 二、实验日期

2018 年 5 月 23 日 软件学院实验室 G346

## 三、实验学生

15130130273 石明皓

## 四、实验目的

本次实验通过给出 2 个适用于 Fork/Join 框架并行执行的情景，要求使用 Fork/Join 框架正确编写实现其逻辑的 Java 多线程并程序。学习 Fork/Join 框架的基本使用方法，理解处理并行任务的分解、合并、创建线程池等步骤，掌握 Java 基础并行程序设计的方法。

## 五、实验内容

题目一：

分别从两个列表中提取出数字，各自相加后，比较大小。两个列表的内容如下：

`list_1={ "*", "%", "3", "#", "6", "~", "!", "2" }`

`list_2={ "&", "¥", "@", "1", "4", ":", "2", "1" }`

要求正确地使用 Fork/Join 框架，计算出 list\_1 中数字的和 (3+6+2) 与 list\_2 中数字的和 (1+4+2+1)，然后比较两个和的大小。

题目二：

统计两个文件中单词 'book' 出现的总次数。

有两个文件，file\_1.dat 与 file\_2.dat，文件内容如下：

file\_1.dat 内容: and, with, we, me, university, with, book, computer, country, book

file\_2.dat 内容: bag, boy, book, school, teacher, student, book, book

要求正确地使用 Fork/Join 框架, 首先统计出 file\_1.dat 中出现 2 次, file\_2.dat 中出现 3 次, 然后统计出 “book” 出现的总次数 (5 次)。

## 六、程序思路、结构

题目一:

题目要求计算并返回数字之和, 故应当继承有返回值的 RecursiveTask 类;

创建一个静态常量来指定并行任务分解的阈值, 该阈值设为 1;

给 extract\_sum\_compare 类的构造函数 extract\_sum\_compare() 传入的参数为:

list 列表、列表起始位置和列表结束位置;

重写 compute 方法, 按照 if(小于阈值){计算子任务;}else{分解任务为子任务;递归执行子任务;最后合并并行计算结果;} 的编程模式完成并行任务;

通过 Integer.parseInt() 来将表示数字的字符串转换为整型数字, 而非数字的字符串将抛出异常, 跳出循环, 结束本线程的计算;

分解子任务时采用二分法, 将列表分成左右两段, 直到每段长度为 1, 达到阈值;

对左右子任务对象使用 fork() 方法开始并行计算, 并分别将使用 join() 方法返回的计算结果相加合并, 得到最终计算结果, 即列表中的数字之和。

要执行 Fork/Join 框架的并程序, 需创建一个线程池, 通过 invoke() 方法把并行任务 (即 RecursiveTask 类的对象) 导入线程池, 才能得到结果。

题目二:

先分别读入 file\_1.dat 和 file\_2.dat, 将其中的内容转换为字符串数组 list1 和 list2, 这两个字符串数组的元素是字符串文件内容去除空格、以逗号分隔的结果;

题目要求计算并返回 book 的出现次数, 故应当继承有返回值的 RecursiveTask 类;

创建一个静态常量来指定并行任务分解的阈值, 该阈值设为 1;

给 java\_book\_sum 类的构造函数 java\_book\_sum() 传入的参数为: list 列表、列表起始位置和列表结束位置;

重写 compute 方法, 按照 if(小于阈值){计算子任务;}else{分解任务为子任务;递归执行子任务;最后合并并行计算结果;} 的编程模式完成并行任务;

通过 list[start].matches(“book”) 函数正则匹配 book, 元素从头到尾必须完全

匹配才为 true;

分解子任务时采用二分法，将字符串数组分成左右两段，直到每段长度为 1，达到阈值；

对左右子任务对象使用 fork()方法开始并行计算，并分别将使用 join()方法返回的计算结果相加合并，得到最终计算结果，即 book 的出现次数；

要执行 Fork/Join 框架的并程序序，需创建一个线程池，通过 invoke()方法把并行任务（即 RecursiveTask 类的对象）导入线程池，才能得到结果。

## 七、程序代码

题目一：

```
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;
public class extract_sum_compare extends RecursiveTask<Integer> {
    private static final int threshold = 1; //阈值
    private String[] list;
    private int start;
    private int end;
    public extract_sum_compare(String[] list, int start, int end) {
        this.list = list;
        this.start = start;
        this.end = end;
    }
    @Override
    protected Integer compute() {
        int sum = 0;
        boolean can_compute = (end - start) < threshold;
        //end-start 等于 0 时才为 true
        if (can_compute) {
            for (int i = start; i <= end; i++) {
                //只循环 start 到 start+1
                try {
                    Integer.parseInt(list[i]); //字符串转换为整型数字
                } catch (Exception e) {
                    //非数字不能转换，抛出异常，跳出循环，结束本线程的计算
                    break;
                }
                sum = Integer.parseInt(list[i]);
                //数字计入结果中，下一步就直接返回结果了
            }
        }
    }
}
```

```

    } else {
        int middle = (start + end) / 2;
        //二分任务，直到每个小任务 list 长度仅为 1

        //创建左右子任务
        extract_sum_compare left = new
extract_sum_compare(this.list, start, middle);
        extract_sum_compare right = new
extract_sum_compare(this.list, middle + 1, end);

        //开始并行计算，也可写成 invokeAll(left, right);
        left.fork();
        right.fork();

        //合并计算结果
        sum = left.join() + right.join();
    }
    return sum;
}

public static void main(String[] args) { //并行实现
    String[] list_1 = {"*", "%", "3", "#", "6", "~", "!", "2"};
    String[] list_2 = {"&", "¥", "@", "1", "4", ":", "2", "1"};
    ForkJoinPool fork_join_pool_1=new ForkJoinPool();//创建线程池
    extract_sum_compare test1 = new extract_sum_compare(list_1,
0, list_1.length - 1); //创建任务
    int result1 = fork_join_pool_1.invoke(test1);
    //invoke() 返回计算结果
    System.out.println(result1);
    ForkJoinPool fork_join_pool_2 = new ForkJoinPool();
    extract_sum_compare test2 = new extract_sum_compare(list_2,
0, list_2.length - 1);
    int result2 = fork_join_pool_2.invoke(test2);
    System.out.println(result2);
    if (result1 > result2) {
        System.out.println("list_1 数字之和 > list_2 数字之和");
    } else if (result1 == result2) {
        System.out.println("list_1 数字之和 == list_2 数字之和");
    } else {
        System.out.println("list_1 数字之和 < list_2 数字之和");
    }
}
}

```

## 题目二:

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.RecursiveTask;
public class java_book_sum extends RecursiveTask<Integer> {
    private static final int threshold = 1; // 阈值
    private String[] list;
    private int start;
    private int end;
    public java_book_sum(String[] list, int start, int end) {
        this.list = list;
        this.start = start;
        this.end = end;
    }
    @Override
    protected Integer compute() {
        int count = 0;
        boolean can_compute = (end - start) < threshold;
        // end-start 等于 0 时才为 true
        if (can_compute) {
            if (list[start].matches("book")) {
                // matches() 函数匹配 book, 从头到尾完全匹配才为 true
                count++;
            }
        } else {
            int middle = (start + end) / 2;
            // 二分任务, 直到每个小任务 list 长度仅为 1

            // 创建左右子任务
            java_book_sum left = new java_book_sum(this.list, start,
middle);
            java_book_sum right = new java_book_sum(this.list, middle
+ 1, end);

            // 开始并行计算, 也可写成 invokeAll(left, right);
            left.fork();
            right.fork();

            // 合并计算结果
            count = left.join() + right.join();
        }
    }
}
```

```

        return count;
    }

    public static void main(String[] args) {
        String[] list1 = null;
        String[] list2 = null;
        try {
            //读入 file_1.dat, 将其中的内容转换为字符串数组 list1 (去
            除空格, 以逗号分隔)
            String file_path1 = "D:\\Kent's
Workspace\\java\\Parallel_Computing\\2nd\\src\\file_1.dat";
            File file1 = new File(file_path1);
            FileInputStream stream1 = new FileInputStream(file1);
            InputStreamReader reader1 = new
            InputStreamReader(stream1);
            BufferedReader buffer1 = new BufferedReader(reader1);
            String line1;
            while ((line1 = buffer1.readLine()) != null) {
                list1 = line1.trim().replace(" ", "").split(",");
            }
            //读入 file_2.dat, 将其中的内容转换为字符串数组 list2 (去
            除空格, 以逗号分隔)
            String file_path2 = "D:\\Kent's
Workspace\\java\\Parallel_Computing\\2nd\\src\\file_2.dat";
            File file2 = new File(file_path2);
            FileInputStream stream2 = new FileInputStream(file2);
            InputStreamReader reader2 = new
            InputStreamReader(stream2);
            BufferedReader buffer2 = new BufferedReader(reader2);
            String line2;
            while ((line2 = buffer2.readLine()) != null) {
                list2 = line2.trim().replace(" ", "").split(",");
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        ForkJoinPool fork_join_pool_1=new ForkJoinPool();//创建线程池
        java_book_sum test1 = new java_book_sum(list1, 0,
list1.length - 1); //创建任务
        int result1 = fork_join_pool_1.invoke(test1);
        //invoke() 返回计算结果
        System.out.println("file_1.dat 中的出现次数: " + result1);
        ForkJoinPool fork_join_pool_2 = new ForkJoinPool();
        java_book_sum test2 = new java_book_sum(list2, 0,
list2.length - 1);
    }
}

```


```

    int result2 = fork_join_pool_2.invoke(test2);
    System.out.println("file_2.dat 中的出现次数: " + result2);
    int result = result1 + result2;
    System.out.println("book 出现的总次数: " + result);
}
}

```

## 八、实验结果

题目一：



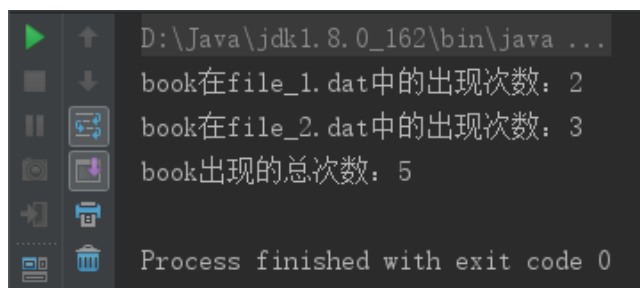
```

D:\Java\jdk1.8.0_162\bin\java ...
11
8
list_1中数字之和 > list_2中数字之和
Process finished with exit code 0

```

从程序运行结果可知，list\_1 中的数字之和为 11，list\_2 中的数字之和为 8，list\_1 中的数字之和大于 list\_2 中的数字之和，结果无误，说明 Fork/Join 框架的并行程序编写成功。

题目二：



```

D:\Java\jdk1.8.0_162\bin\java ...
book在file_1.dat中的出现次数: 2
book在file_2.dat中的出现次数: 3
book出现的总次数: 5
Process finished with exit code 0

```

从程序运行结果可知，book 在 file\_1.dat 中出现次数为 2，在 file\_2.dat 中出现次数为 3，出现的总次数为 5，结果无误，说明 Fork/Join 框架的并行程序编写成功。

## 九、总结建议

经过本次实验，我对并行程序设计的原理有了更深的认识，学会运用 Fork/Join 框架解决简单的并行计算问题；对于 Fork/Join 框架，最重要的是找到合理且有效的任务分解方式，给出符合系统算力的阈值；同时我通过进一步学习，对线程池的概念有了一定了解；另外实验的代码编写环节加深了我在 Java 多线程并行方面编程的熟练度，进而提高了个人的 Java 编程能力。