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 并行程序设计

任课教师： 徐悦甦老师

课程班级： 15 级-云计算方向

学号姓名： 15130120141-孙 晖

提交日期： 2018 年 5 月 24 日

目录

软件学院本科生《并行计算》

课程实验报告

一、上次实验总结

这里我想对上一次实验也就是第一次试验做一个解释说明，上次实验报告的名称我写的是 Java 并发与并行程序设计，实际上实验只涉及到 java 并发，并没有涉及到并行，所以实际上实验名称应该是 java 并发程序设计。之所以写错，并不是因为不清楚概念，而是 PPT 总结的四次实验名称分别是（JAVA 并发与并行程序设计、Python 并发与并行程序设计、Mapreduce 等等）因此我将第一次实验的名称误写为 Java 并发与并行程序设计。

二、实验名称

JAVA 并行程序设计（共两道题）

第一题：找出两个列表里的数字相加后比大小，用 fork/join 框架。

第二题：统计两个文件中单词 ‘book’ 出现的总次数（选做）。

三、实验日期

2018 年 05 月 23 日　软件学院实验室 G346

四、实验学生

15130120141 孙晖

五、实验目的

本次实验老师通过给两道题希望我们掌握 fork/join 框架编写 java 并行程序的能力。

六、实验内容

本次实验要求学生独自完成，不存在组队的情况。

[1] 第一题：

实验第一题要求用 java 语言找出两个列表里的数字相加后比大小，用 fork/join 框架。

list_1= { “*”， “%”， “3”， “#”，” 6”， “~”， “!”，” 2” } ；

list_2= { “&”， “￥”， “@”， “1”，” 4”， “:”， “2”，” 1” }。

[2] 第二题：

实验第二题要求用 java 语言来统计两个文件中单词 ‘book’ 出现的总次数，两个

文件分别为 file_1.dat 与 file_2.dat（文件类型也可以是其它文本文件类型，如.txt)。

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

七、程序思路、结构

[1] 第一题：

关于 fork/join 框架，很多网上的博客都是用一个程序来讲解，很是费时费力，还看着逻辑不清楚，因此我总结一个规定死的步骤去使用 fork/join 框架，学习阿三精神（代码都规定死，长一个样子)。

程序思路：

第 0 步：如何分割任务?

这个任务是否值得分割？如果值得分割要如何分割?

一个 if...else 来判断是否值得分割。常用给的分割方法是二等分。

第 1 步：首先需要从 RecursiveTask extends 出来一个类 A，在这个类中初始化阈值、start、end，其中阈值想设置可以设置，也可以不设置，主要是为了在大于阈值的时候把整个任务分开。接着需要重写构造函数用来传入 start 和 end 参数。

第 2 步：重写上述类中的 compute 方法，先用第 0 步里说的 if...else 判断一下是否要并行。Else 里面写并行代码块。实例化两个子任务（类名 A 实例 1；类名 A 实例 2.....)。执行子任务（实例 1.fork()；实例 2.fork().....）.用（实例名.join()）把分开的执行结果合并到一起。

第 3 步：ForkJoinPool 生成一个任务。

ForkJoinPool ex = new ForkJoinPool();//生成一个任务

类名 A task = new 类名（参数 1，参数 2...）;//执行一个任务

Future result = ex.submit(task);//结果，返回的是 compute return 的结果。

结构：

如何使用 fork/join

```
class A extends RecursiveTask{
        变量；
        构造器
        protected 类型 compute(){
                if(){}else{
                        A instance1 = new A(参数1，...);
                        A instance2 = new A(参数1，...);
                        instance1.fork();
                        instance2.fork();
                        result1=instance1.join();
                        result2=instance2.join();
                        sum = result1+result2;
                }
                return ;
        }
main函数：ForkJoinPool  a = new ForkJoinPool();
          A task= new A(参数); Future result = a.submit(task);
```

[2] 第二题：

程序思路：

我在写程序的时候，思路就是先实现这个功能再说，并行在实现功能之后再改。我不知道这种思想对不对，这样的一种方法浪费了很多时间，因为把程序改成并行的程序需要对参数传递进行转换，这期间为了调参数的传递耗费了我大量的时间，因为对于 java 各种自带的方法使用不熟练，导致 Arraylist 和 String[]之间的转换错误百出，再加上两者的方法有很多，具体用哪种方法又成为一个头疼的问题，总的来说并行不是问题，问题的是 java 语言的使用出问题。

结构：

main 方法里对传入的参数进行预先处理，处理好后传入到 PFindBook 类里面，然后用 fork/join 框架，其中 if...else 那里出了很多问题，导致浪费了大量的时间去调试。

八、程序代码

[1] 第一题代码：

```
package Java_Study;
import java.util.List;
import java.util.concurrent.ForkJoinPool;
```

```java
import java.util.ArrayList;
import java.util.Arrays;
import java.util.concurrent.Future;
import java.util.concurrent.RecursiveTask;

class Max extends RecursiveTask<Integer>{
    private int threshold;
    private   List<String> list;
    //private   List<String> list1;

    //private   List<String> list2;

    public Max(List<String> list,int threshold){
        this.list = list;
        this.threshold = threshold;
    }
    protected Integer compute(){

        Integer number=0;
        if (list.size() < threshold) {
            for(int i=0;i<list.size();i++){
                if((list.get(i)!=null&&list.get(i).matches("^[0-9]+$"))){
                    number += Integer.parseInt(list.get(i));
                }
            }

        }else {
            int middle = list.size() / 2;
            List<String> leftList = list.subList(0, middle);
            List<String> rightList = list.subList(middle, list.size());
            Max left = new Max(leftList, threshold);
            Max right = new Max(rightList, threshold);
            left.fork();
            right.fork();
            Integer join1 = left.join();
            Integer join2 = right.join();
            number = join1 + join2;
        }
        return number;
    }
}

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

```
        String[] list_1 = {"*", "%", "3", "#", "6", "~", "!", "2"};
        String[] list_2 = {"&", "￥", "@", "1", "4", ":", "2", "1"};
        List<String> stringList1 = new ArrayList<>(Arrays.asList(list_1));
        List<String> stringList2 = new ArrayList<>(Arrays.asList(list_2));
        ForkJoinPool a= new ForkJoinPool();
        Max task1 = new Max(stringList1,3);
        Max task2 = new Max(stringList2,3);
        Future<Integer> result1 = a.submit(task1);
        Future<Integer> result2 = a.submit(task2);
        try{
                int x = result1.get();
                int y = result2.get();
                if(x>y){
                    System.out.println(x+">"+y+" 即 list_1>list_2");
                }else{
                    System.out.println(y+">"+x+" 即 list_1<list_2");
                }
        }catch (Exception e){}
    }
}
```

[2] }第二题代码：

```
package Java_Study;
import java.io.*;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.ForkJoinPool;
import java.util.concurrent.Future;
import java.util.concurrent.RecursiveTask;
/*
class Findtimes{
    public static int FindtimesFunction(File file){
        try {
            FileInputStream fim = new FileInputStream(file);
            BufferedReader br = new BufferedReader(new InputStreamReader(fim));
            String str = "";//初始化
            String result = "";
            while ((str = br.readLine()) != null) {
                result += str;
            }
            System.out.println("文件内容：" + result);
            String[] arr = result.split(",");
            int num=0;
```

```
                for(int i=0;i<arr.length;i++){
                    if(arr[i].equals("book")){
                        num++;
                    }
                }
                System.out.println("文件含有"+num+"个 book。");
            }catch(FileNotFoundException e){}
            catch(IOException e){}
            return 0;
        }
    }
    */
    class PFindTimes extends RecursiveTask<Integer> {
        private int threshold;
        String[] arr;
        int start,end;
        PFindTimes(int start,int end,int threshold,String[] arr){
            this.start = start;
            this.end = end;
            this.threshold = threshold;
            this.arr=arr;
        }

        protected Integer compute(){
            int num=0;
            List<String> ex = Arrays.asList(arr);
            if (ex.size()<threshold){
                for(int i=0;i<ex.size();i++){
                    if(arr[i].equals("book")){
                        num++;
                    }
                }
            }else{
                int middle = arr.length / 2;
                String[] leftString=new String[5];
                String[] rightString=new String[5];

                for(int i=0;i<arr.length;i++){
                    if(i<middle){

                        leftString[i]=arr[i];

                    }else{
                        rightString[i-middle]=arr[i];
```

```
                }
            }

            PFindTimes left = new PFindTimes(0,middle, threshold,leftString);
            PFindTimes right = new
PFindTimes(middle,arr.length,threshold,rightString);

            left.fork();
            right.fork();
            Integer join1 = left.join();
            Integer join2 = right.join();
            num = join1 + join2;
        }
        return num;
    }
}

public class FindBookFinal {
    public static void main(String[] args){
        File file_1 = new File("file_1.txt");
        File file_2 = new File("file_2.txt");
        try {
            FileInputStream fim1 = new FileInputStream(file_1);
            BufferedReader br1 = new BufferedReader(new
InputStreamReader(fim1));
            String result1 = "";//初始化
            String str1 = "";
            while ((str1 = br1.readLine()) != null) {
                result1 += str1;
            }
            System.out.println("文件 1 内容：" + result1);
            String[] arr1 = result1.split(",");

            FileInputStream fim2 = new FileInputStream(file_2);
            BufferedReader br2 = new BufferedReader(new
InputStreamReader(fim2));
            String result2 = "";//初始化
            String str2 = "";
            while ((str2 = br2.readLine()) != null) {
                result2 += str2;
            }
            System.out.println("文件 2 内容：" + result2);
            String[] arr2 = result2.split(",");
```

```
                ForkJoinPool a= new ForkJoinPool();
                PFindTimes task1 = new PFindTimes(0,9,6,arr1);
                PFindTimes task2 = new PFindTimes(0,7,6,arr2);

                Future<Integer> r1 = a.submit(task1);
                Future<Integer> r2 = a.submit(task2);
                try{
                    int x = r1.get();
                    int y = r2.get();
                    System.out.println("文件 1 中含有："+x+"个 book");
                    System.out.println("文件 2 中含有："+y+"个 book");
                }catch (Exception e){}
            }catch (FileNotFoundException e){}
            catch(IOException e){}
        }
    }
```

九、实验结果

[1] 第一题结果：

```
"C:\Program Files\Java\jdk1.8.0_101\bin\java" ...
11>8 即 list_1>list_2
```

图 1 计算结果 11>8，就是说 list_1 里数字的和大于 list_2 里的数字的和

[2] 第二题结果：

```
文件1内容：and,with,we,me,university,with,book,computer,country,book
文件2内容：bag,boy,book,school,teacher,student,book,book,sun,hui
文件1中含有：2个book
文件2中含有：3个book
```

图 2 文件 1 中的 book 数量为两个，文件 2 中为 3 个

十、总结建议

[1] Java 的异常需要总结一下，不能傻傻的用着却不知道用的到底是什么原理。

[2] Java int 和 Integer?

[3] 这次实验好难，很多不会的东西，弄了很长时间，照葫芦画瓢都画不好，不懂的地方很多，错都不知道错哪，调试毫无思路，全靠碰运气。要学的东西好多。好难。

[4] Fork 一定要关闭吗，看老师 PPT 上是手动关闭的。

[5] 曾经一度自以为 java 的 collection 学的 ok 了，结果今天才发现很差劲，String[] 到 Array 到 List 各种转换，各种接口或者类内的方法全都记不住。导致不知道用 List 好还是 Set 好还是 Map 好，各种疑惑，各种需要查，这一块需要好好画个图总结总结。

[6] Fork/join 我总结并画图后还是很好记住的，所以 java 并行我认为是没什么大问题的。