《大数据分析技术》课程论文

题目：城市空气质量分析

——基于 **MapReduce** 框架实现

评分：

评语：

时间：**2020.07.02**

目录

1. 摘要

本实验题目为“城市空气质量分析——基于 MapReduce 框架实现”。本实验的目标是利用 MapReduce 框架，分析各城市的空气质量水平。

首先以 PM25 的空气质量分指数（IAQI）为衡量指标，比较 2018 年 8 月至 2019 年 6 月间各城市的空气质量水平，结果表明：给定时间范围内，城市的 PM25 浓度可以划分为四个梯队，污染最轻的城市为海口和昆明，污染最重的城市为郑州。

其次，本实验以北京、上海和成都三个城市为例，以 AQI 为分析指标，统计出春节期间三个城市的空气质量等级分布情况。结果表明，上海的空气质量水平优于北京，成都的空气质量水平最差。

最后，本实验构建空气质量综合指数，将 12 个城市全部纳入分析框架，计算结果为每个城市得到一个综合评分，结果表明：空气质量综合水平较高的城市有青岛、厦门、海口、成都、上海、昆明等，而空气质量综合水平较差的有乌鲁木齐、北京、呼和浩特、天津、郑州等，该结果也与 PM25 指标分析结果较为接近。

2.课程论文依托的实验环境

本实验依托的环境为：

Hadoop 版本：2.6.5；

虚拟机软件：VMware 11.0.0 build-2305329

操作系统：Linux master 2.6.32-504.el6.x86_64 #1 SMP Wed Oct 15 04:27:16 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux

Linux 发行版本：CentOS release 6.6 (Final)

Hadoop 集群：一台 master，两台 slaver1，slaver2

3. 实验需要的理论和技术介绍

本实验主要依托的理论和技术主要是 MapReduce 技术。MapReduce 是一个分布式运算程序的编程框架，用于开发分布式的数据分析系统；核心功能是将用户编写的业务逻辑代码和自带默认组件整合成一个完整的分布式运算程序，并发运行在一个 Hadoop 集群上。

3.1 运行机制

一个 MapReduce 程序在分布式运行时有三种进程：

1、MRAppMaster：负责整个程序的过程调度及状态协调；

2、MapTask：负责 map 阶段的整个数据处理流程；

3、ReduceTask：负责 reduce 阶段的整个数据处理流程。

3.2 运行流程

一个 MapReduce 程序启动的时候，最先启动的是 MRAppMaster，MRAppMaster 启动后根据本次 job 的描述信息，计算出需要的 MapTask 实例数量，然后向集群申请机器启动相应数量的 MapTask 进程

MapTask 进程启动之后，根据给定的数据切片范围进行数据处理，主体流程如图 1 所示。

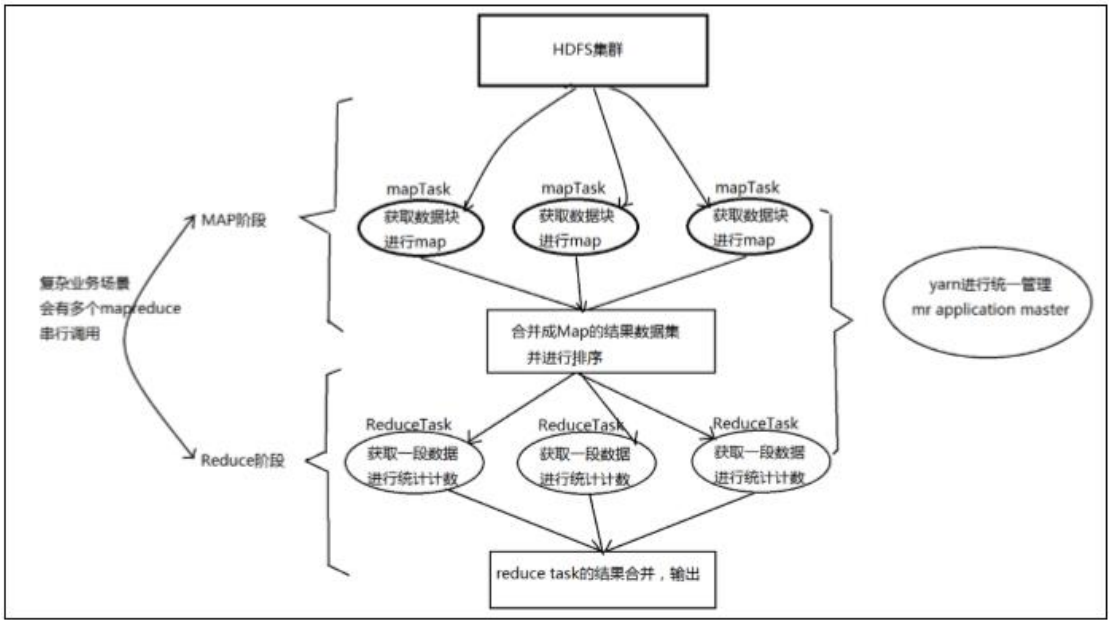
第一，利用客户指定的 inputFormat 来获取 RecordReader【数据记录阅读器】读取数据，形成输入键值对（<K,V> ：[（张三，1），（李四，1）]）；

第二，将输入键值对（<K,V>）传递给客户定义的 map()方法，做逻辑运算，并将 map()方法 输出的键值对（<K,V>）对收集到缓存；

第三，将缓存中的键值对（<K,V>）对按照 K 分区排序后不断写到磁盘文件。MRAppMaster 监控到所有 MapTask 进程任务完成之后，会根据客户指定的参数启动相应数量的 ReduceTask 进程，并告知 ReduceTask 进程要处理的数据范围（数据分区）；

第四，ReduceTask 进程启动之后，根据 MRAppMaster 告知的待处理数据所在位置，从若干台 MapTask 运行所在机器上获取到若干个 MapTask 输出结果文件，并在本地进行重新 归并排序，然后按照相同 key 的<K,V>为一个组，调用客户定义的 reduce()方法进行逻辑运算，并收集运算输出的结果<K,V>，然后调用客户指定的 outputformat 将结果数据输出到外部存储。

图 1　MapReduce 运行流程示意图



4.实验题目需求分析

4.1 数据来源和格式

本实验主要依托空气质量数据，数据来源于网络，数据规模达到 40 万行，时间从 2018.08.01 的零点至 2019.06.10 的 23 点，共 16 个字段。字段分别为：站号，经度，纬度，PM25，PM10，NO2，SO2，O3-1，O3-8h，CO，AQI，等级，年，月，日，小时，城市。每个字段之间以"，"进行分割。其中，“城市”字段为字符串类型，其余字段均为数值型数据。

此外，城市包括北京、上海、天津、青岛、济南、厦门、郑州、乌鲁木齐、成都、呼和浩特、海口和昆明，共 12 个城市；每个城市有不同数量的空气质量监测站点，站点的采集频率为每小时一条记录。

4.2 解决问题思路

4.2.1 第一题解决思路

第一题中，需求是计算给定时间范围内，各个城市的 PM25 统计指标，比较并排序。解决问题的关键点有二：其一是 PM25 统计指标的选取，其二是基于空气质量数据计算出各城市的 PM25 数值。

首先确定 PM25 指标。本实验选取空气质量分指数 IAQI（Individual Air Quality Index, IAQI）作为 PM25 的度量指标。指标选取的原因是，IAQI 是中华人民共和国国家环境保护标准（HJ 633—2012）规定的，用于衡量单项污染物的空气质量指数。IAQI 指数的具体计算方法是：对照各项污染物的分级浓度限值，以细颗粒物（PM2.5）、可吸入颗粒物（PM10）、二氧化硫（SO2）、二氧化氮（NO2）、臭氧（O3）、一氧化碳（CO）等各项污染物的实测浓度值（其中 PM2.5、PM10 为 24 小时平均浓度）分别计算得出空气质量分指数。据此，本实验中可用 PM25 空气质量分指数来度量 PM25 指标。根据环境空气质量指数（AQI）技术规定（试行），空气质量分指数级别及对应的污染物项目浓度限值见表 1，其中 PM25 的浓度限值即为最后一列——颗粒物（粒径小于等于2.5μm）24 小时平均/($\mu g/m^3$)的浓度限值。IAQI 的计算公式如下：

$$IAQI_P = \frac{IAQI_{Hi} - IAQI_{Lo}}{BP_{Hi} - BP_{Lo}}(C_P - BP_{Lo}) + IAQI_{Lo} \quad (1)$$

式中，$IAQI_P$表征污染物项目P的空气质量分指数，本实验中特指 PM25；C_P是污染物项目P的质量浓度值；BP_{Hi}为表 1 中与C_P相近的污染物浓度限值的高位值；BP_{Lo}为表 1 中与C_P相近的污染物浓度限值的低位值；$IAQI_{Hi}$为表 1 中与BP_{Hi}对应的空气质量分指数；$IAQI_{Lo}$为表 1 中与BP_{Lo}对应的空气质量分指数。

其中，$IAQI_{Hi} - IAQI_{Lo}$的含义是 PM25 在表 1 中对应的空气质量分指数高、低之差值，是一个无量纲的数，它的每一级别值均是一个固定值。$BP_{Hi} - BP_{Lo}$的含义是表 1 中与 PM25 质量浓度值相近的污染物浓度限值的高、低位值之差值，它的单位与浓度单位一样，而且它的每一级别值也是一个固定值。

$C_P - BP_{Lo}$的含义是表 1 中与 PM25 质量浓度值相近的污染物浓度限值的低位值之差，它的单位也是与浓度单位一样，它不是一个固定的值。

因此根据该公式，可计算出每小时下各城市各站点监测到的 PM25 对应的空气质量分指数数值。

表 1 空气质量分指数及对应的污染物项目浓度限值

空气质量分指数（IAQI）	污染物项目浓度限值									
	二氧化硫（SO_2）24 小时平均/（μg/m³）	二氧化硫（SO_2）1 小时平均/（μg/m³）〔1〕	二氧化氮（NO_2）24 小时平均/（μg/m³）	二氧化氮（NO_2）1 小时平均/（μg/m³）〔1〕	颗粒物（粒径小于等于10μm）24 小时平均/（μg/m³）	一氧化碳（CO）24 小时平均/（mg/m³）	一氧化碳（CO）1 小时平均/（mg/m³）〔1〕	臭氧（O_3）1 小时平均/（μg/m³）	臭氧（O_3）8 小时滑动平均/（μg/m³）	颗粒物（粒径小于等于2.5μm）24 小时平均/（μg/m³）
0	0	0	0	0	0	0	0	0	0	0
50	50	150	40	100	50	2	5	160	100	35
100	150	500	80	200	150	4	10	200	160	75
150	475	650	180	700	250	14	35	300	215	115
200	800	800	280	1 200	350	24	60	400	265	150
300	1 600	〔2〕	565	2 340	420	36	90	800	800	250
400	2 100	〔2〕	750	3 090	500	48	120	1 000	〔3〕	350
500	2 620	〔2〕	940	3 840	600	60	150	1 200	〔3〕	500
说明：	〔1〕二氧化硫（SO_2）、二氧化氮（NO_2）和一氧化碳（CO）的 1 小时平均浓度限值仅用于实时报，在日报中需使用相应污染物的 24 小时平均浓度限值。 〔2〕二氧化硫（SO_2）1 小时平均浓度值高于 800 μg/m³ 的，不再进行其空气质量分指数计算，二氧化硫（SO_2）空气质量分指数按 24 小时平均浓度计算的分指数报告。 〔3〕臭氧（O_3）8 小时平均浓度值高于 800 μg/m³ 的，不再进行其空气质量分指数计算，臭氧（O_3）空气质量分指数按 1 小时平均浓度计算的分指数报告。									

其次计算出各城市的指标数值。具体的计算方法是：基于一定时间范围内、该城市各站点的空气质量数据，采用简单算术平均法，计算出该城市在某一时段内的 PM25 的 IAQI 指标，再排序。

4.2.2 第二题解决思路

第二题中，难点在于限制条件较多。首先限制时间范围是 2019 年 2 月份，需加入 if 条件判断。且统计单位为每天，需按照时间计算每日 AQI 平均值。其次限制城市为北京、上海、成都三个城市，也需加入 if 条件判断，并分别计算每个城市的空气质量分布情况。最后，在每个城市中，需要对 AQI 数值进行分类并统计。分类标准需参照国家颁布的标准，具体如下图所示。数据已给出 AQI 等级，字段为“等级”。综上，问题转换为求出给定时间范围内，给定城市下，以天为单位，统计不同 AQI 级别的天数分布。

解决方法上，本题采用 MapReduce 框架的话，笔者思考出两种方式：

第一种，若分成两部走，第一步统计出每个城市、每日的日均 AQI，第二步再以第一步的输出结果作为输入文件，统计出 AQI 级别的天数分布。该方式思路简单直接，代码量虽大，但实现难度不大。

第二种，仅运行一次，将计算日均 AQI 和计算 AQI 等级分布两个部分都在 Reducer 中实现。首先将日期与对应的多个 AQI 值存储在 HashMap 中，其中日期为 key，多个 AQI 值组成一个 ArrayList，作为 HashMap 的 value；其次计算出日均 AQI 并将其存储在 ArrayList 的末尾；最后构建二维数组，分别存储三个城市和日均 AQI 值，循环遍历，统计不同空气质量水平下的天数分布。该方

法实现起来较复杂，难度较大，考察对 HashMap 和 ArrayList 的掌握水平，以及二维数组的构建和遍历。因此本实验为锻炼自己，增大任务挑战性，选取该方式，具体实现过程见核心代码部分。

图 2 我国空气质量等级

AQI数值	AQI级别	AQI类别及表示颜色		对健康的影响	建议采取的措施
0～50	一级	优	绿色	空气质量令人满意，基本无空气污染	各类人群可正常活动
51～100	二级	良	黄色	空气质量可接受，但某些污染物可能对极少数异常敏感人群健康有较弱影响	极少数异常敏感人群应减少户外活动
101～150	三级	轻度污染	橙色	易感人群症状有轻度加剧，健康人群出现刺激症状	儿童、老年人及心脏病、呼吸系统疾病患者应减少长时间、高强度的户外锻炼
151～200	四级	中度污染	红色	进一步加剧易感人群症状，可能对健康人群心脏、呼吸系统有影响	儿童、老年人及心脏病、呼吸系统疾病患者应避免长时间、高强度的户外锻炼，一般人群适量减少户外运动
201～300	五级	重度污染	紫色	心脏病和肺病患者症状显著加剧，运动耐受力降低，健康人群普遍出现症状	儿童、老年人及心脏病、呼吸系统疾病患者应停留在室内，停止户外活动，一般人群应避免户外活动
>300	六级	严重污染	褐红色	健康人群运动耐受力降低，有明显强烈症状，提前出现某些疾病	儿童、老年人及心脏病、呼吸系统疾病患者应停留在室内，避免体力消耗，一般人群应避免户外活动

4.2.3 第三题解决思路

第三题中，本实验拟订的需求为：计算给定城市在给定时间范围内的空气质量综合指数。给定城市为全部 12 个城市，给定时间范围为 2018.08.01 的零点至 2019.06.10 的 23 点。

空气质量综合指数，是中华人民共和国环境保护部用于对 74 个重点城市空气质量进行评价、排名的一种工具。因此计算该指数具有很强的现实意义和实践意义。

空气质量综合指数，亦可称环境空气质量综合指数，是描述城市环境空气质量综合状况的无量纲指数，综合考虑了《环境空气质量指数（AQI）技术规定(试行)》（HJ633-2012）中规定的：SO_2、NO_2、PM_{10}、$PM_{2.5}$、CO、O_3等六种污染物污染程度，空气质量综合指数值越大表明综合污染程度越重。

空气质量综合指数的计算方法如下：

（1）计算各污染物的统计量浓度值

统计各城市的 SO2、NO2、PM10、PM2.5 的月均浓度，并统计一氧化碳（CO）日均值的第 95 百分位数以及臭氧（O3）日最大 8 小时值的第 90 百分位数。

（2）计算各污染物的单项指数

污染物 i 的单项指数 Ii 按式 1 计算：

$$I_i = \frac{C_i}{S_i} \tag{2}$$

式中：C_i为污染物 i 的浓度值，当 i 为 SO2、NO2、PM10 及 PM2.5 时，C_i为月均值，当 i 为 CO 和 O3 时，C_i为式中相应的百分位数浓度值；

S_i为污染物 i 的年均值二级标准（当 i 为 CO 时，为日均值二级标准；当 i 为 O3 时，为 8 小时均值二级标准）。该标准见表 2。

（3）计算空气质量综合指数

空气质量综合指数的计算需涵盖全部六项污染物，计算方法如下所示：

$$I_{sum} = \sum_{i=1}^{6} I_i \tag{3}$$

式中：I_{sum}为环境空气质量综合指数；I_i为污染物 i 的单项指数，i 包括全部六项指标。

当环境空气质量综合指数相同时，排名以并列计。

因此本实验根据上述公式计算空气质量综合指数，该指数越小，代表该城市的空气质量越好，指数越大，表明该城市的空气质量越差。

表 2 环境空气污染物基本项目浓度限值表

污染物项目	平均时间	浓度限值		单位
		一级	二级	
SO_2	年平均	20	60	μg/m³
	24 小时平均	50	150	
	1 小时平均	150	500	
NO_2	年平均	40	40	
	24 小时平均	80	80	
	1 小时平均	200	200	
CO	24 小时平均	4	4	mg/m ³
	1 小时平均	10	10	
O_3	8 小时平均	100	160	μg/m³
	1 小时平均	160	200	
PM_{10}	年平均	40	70	
	24 小时平均	50	150	
$PM_{2.5}$	年平均	15	35	
	24 小时平均	35	75	

4.3 核心代码

4.3.1 第一题核心代码

此处省略导入包、get 和 set 方法，仅展示首次 Runner.java

1. AQIBean.java

```
package AQI;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.WritableComparable;

public class AQIBean implements WritableComparable<AQIBean>{
    double PM25;
    double IAQI;
    String city;

    public AQIBean(){
    }

    public AQIBean(double PM25, String city){
```

```
            super();
            this.PM25 = PM25;
            this.city = city;

            //数组初始化
            //pollutant_levels 存储 PM25 浓度等级，IAQI_levels 存储空气质量分指数 IAQI 等级
            int[] pollutant_levels = {0, 35, 75, 115, 150, 250, 350, 500};
            int[] IAQI_levels = {0, 50, 100, 150, 200, 300, 400, 500};

            //计算 IAQI
            for (int i = 0; i < 7; i++){
                if (PM25 > pollutant_levels[i] && PM25 < pollutant_levels[i+1]){
                    this.IAQI = (IAQI_levels[i+1] - IAQI_levels[i]) * (PM25 - pollutant_levels[i]) / (pollutant_levels[i+1] - pollutant_levels[i]) + IAQI_levels[i];
                }
            }
        }
       //此处省略 get、set

        @Override
        public void write(DataOutput out) throws IOException {
            out.writeDouble(PM25);
            out.writeDouble(IAQI);
            out.writeUTF(city);
        }

        @Override
        public void readFields(DataInput in) throws IOException {
            PM25 = in.readDouble();
            IAQI = in.readDouble();
            city = in.readUTF();
        }
        @Override
        public int compareTo(AQIBean a) {
       // TODO Auto-generated method stub
       return (this.IAQI > a.getIAQI())?-1:1;
       }

        @Override
        public String toString() {
       // TODO Auto-generated method stub
       return city+"\t"+IAQI;
       }
```

```
}
```

2. AQIMapper.Java

```
public class AQIMapper extends Mapper<LongWritable, Text, Text, AQIBean>{

	@Override
	protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
		if (key.toString().equals("0")) {
			return;
	    }
		String line = value.toString();
		String[] fields = line.split(",");
		//System.out.println(fields[3]);
		double PM25 = Double.parseDouble(fields[3]);
		String city = fields[16];
		context.write(new Text(city), new AQIBean(PM25, city));
	}
}
```

3. AQIReducer.java

```
public class AQIReducer extends Reducer<Text, AQIBean, Text, Text>{

	@Override
	protected void reduce(Text key, Iterable<AQIBean> values, Context context)
throws IOException, InterruptedException {

		double IAQI_sum = 0;
		double count = 0;
		for (AQIBean value : values) {
			IAQI_sum += value.getIAQI();
			count += 1;
		}
		double IAQI_average = IAQI_sum / count;

		context.write(new Text(key), new Text(String.valueOf(IAQI_average)));
	}
}
```

4. FlowRunner.java

```
public class FlowRunner {
	public static void main(String[] args) throws IOException, InterruptedException,
ClassNotFoundException {
		BasicConfigurator.configure();
		Configuration con = new Configuration();
		Job job = Job.getInstance(con);
		job.setJarByClass(FlowRunner.class);
```

```
        job.setMapperClass(FlowMapper.class);
        job.setReducerClass(FlowReducer.class);

        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(FlowBean.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(FlowBean.class);

        FileInputFormat.setInputPaths(job, new
Path("hdfs://192.168.1.192:9000/user/root/flowinput"));
        FileOutputFormat.setOutputPath(job, new
Path("hdfs://192.168.1.192:9000/user/root/flowoutput"));

        job.waitForCompletion(true);
    }
}
```

5. CityAQISortTest.java

```
package mycp7;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

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

        String[] cityArr = {"上海","乌鲁木齐","北京","厦门","呼和浩特","天津",
            "成都","昆明","济南","海口","郑州","青岛"};
        Double[] AQIArr =
{55.18360058440512,105.50207880024911,70.24525967238151,37.904536662225
2,

55.0762874278499,72.85804564495083,74.10958049198001,32.625129602441895,1
06.56531048165962,
            22.56553826042011,119.8591981616064,78.55928168903378};

        List<CityAQI> list = new ArrayList<CityAQI>();
        for (int i=0;i<cityArr.length;i++){
            CityAQI caqi = new CityAQI(cityArr[i], AQIArr[i]);
            list.add(caqi);
        }
```

```
        Collections.sort(list, new Comparator<CityAQI>() {
            @Override
            public int compare(CityAQI o1, CityAQI o2) {
                String s1 = o1.getcity();
                String s2 = o2.getcity();
                Double m1 = o1.getaqi();
                Double m2 = o2.getaqi();
                return m1.compareTo(m2);
            }
        });
        for (CityAQI value:list){
            System.out.println(value);
        }

    }

    public static class CityAQI{

      public String city;
      public double aqi;

      public CityAQI(String city,    double aqi) {
        this.city = city;
        this.aqi = aqi;
      }

      public String getcity() {
        return city;
      }

      public double getaqi() {
        return aqi;
      }

      @Override
      public String toString() {
        return city+ "\t" + aqi ;
      }
    }
}
```

5.2.1 第二题核心代码

1. AQIClassfyMapper.java

```
public class AQIClassfyMapper extends Mapper<LongWritable, Text, Text, Text> {

    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        if (key.toString().equals("0")) {
            return;
    }
        String line = value.toString();
        String[] fields = line.split(",");
        double AQI = Double.parseDouble(fields[10]);
        String year = fields[12];
        String month = fields[13];
        String day = fields[14];
        String date = year + month + day;
        String city = fields[16];
        String date_city = date+" "+city;

        // 限制时间条件
        if ("2019".equals(year) && "2".equals(month) && ("北京".equals(city) || "
上海".equals(city) || "成都".equals(city))){
            context.write(new Text(city), new Text(date+"
"+String.valueOf(AQI)));
        }
    }
}
```

2. AQIClassfyReducer.java

```
public class AQIClassfyReducer extends Reducer<Text, Text, Text, Text>{
    //@Override
    protected void reduce(Text key, Iterable<Text> values, Context context) throws
IOException, InterruptedException {
        String city = key.toString();
        //创建一个日期-aqi 数组的 map
        HashMap<String,ArrayList<Double>> date_aqi = new HashMap<>();
        for(Text val : values){
            double aqi = Double.parseDouble(val.toString().split(" ")[1]);
            String dateString = val.toString().split(" ")[0];
            if (date_aqi.containsKey(dateString)) {
                    ArrayList<Double> aqi_list = date_aqi.get(dateString);
                    aqi_list.add(aqi);
                }
            else{
                    ArrayList<Double> aqi_list = new ArrayList<>();
                    aqi_list.add(aqi);
```

```
                date_aqi.put(dateString, aqi_list);
            }
}
// 计算每日平均 AQI

for(Map.Entry<String, ArrayList<Double>> entry : date_aqi.entrySet()){
    double aqi_sum = 0;
    ArrayList<Double> aqiList = entry.getValue();
     for (int i = 0; i < aqiList.size(); i++) {
                aqi_sum = aqi_sum + aqiList.get(i);
            }
    double aqi_avg = aqi_sum/aqiList.size();
    aqiList.add(aqi_avg);
}

//System.out.println(aqi_avg);

int[][] city_count = new int[3][6];
HashMap<String,Integer> city_code = new HashMap();
String[] citys = {"北京","上海","成都"};
for(int t = 0; t < citys.length; t++){
    city_code.put(citys[t], t);
}
for(Map.Entry<String, ArrayList<Double>> entry : date_aqi.entrySet()){
    ArrayList<Double> aqiList2 = entry.getValue();
    Double aqi_avg = aqiList2.get(aqiList2.size()-1) ;
if (aqi_avg<=50) {
    city_count[city_code.get(city)][0] += 1;
}
else if (aqi_avg<=100) {
    city_count[city_code.get(city)][1] += 1;
}
else if (aqi_avg<=150) {
    city_count[city_code.get(city)][2] += 1;
}
else if (aqi_avg<=200) {
    city_count[city_code.get(city)][3] += 1;
}
else if (aqi_avg<=300) {
    city_count[city_code.get(city)][4] += 1;
}
else{
    city_count[city_code.get(city)][5] += 1;
}
```

```
            }

            String[] results = new String[3];
            for (int i = 0;i<citys.length;i++){
                  results[i] = Arrays.toString(city_count[i]);
            }
            //String result = results.toString("\n");
            context.write(new Text(city), new Text(results[city_code.get(city)]));
      }
}
```

5.2.2 第三题核心代码

1. AQIIndexMapper.java

```
public class AQIIndexMapper extends Mapper<LongWritable, Text, Text, Text>{

      protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
            if (key.toString().equals("0")) {
                  return;
      }
            String line = value.toString();
            String[] fields = line.split(",");
            //double AQI = Double.parseDouble(fields[10]);
            String year = fields[12];
            String month = fields[13];
            String day = fields[14];
            String date = year + month + day;
            String city = fields[16];
            String PM25 = fields[3];
            String PM10 = fields[4];
            String NO2    = fields[5];
            String SO2 = fields[6];
            String O3 = fields[8];
            String CO = fields[9];

            String date_air =
date+"\t"+PM25+"\t"+PM10+"\t"+NO2+"\t"+SO2+"\t"+O3+"\t"+CO;

            // 限制时间条件
            context.write(new Text(city), new Text(date_air));
            }
      }
```

2. AQIIndexReducer.java

```
public class AQIIndexReducer extends Reducer<Text, Text, Text, Text>{
```

```
private static final double[] LIMITS = {75.0,150.0,80.0,150.0,160.0,4.0};
protected void reduce(Text key, Iterable<Text> values, Context context)
                                        throws IOException, InterruptedException {
    String city = key.toString();
    //创建一个日期-ap 数组的 map
    HashMap<String,ArrayList<Double[]>> date_ap = new HashMap<>();
    for(Text val : values){
        Double PM25 = Double.parseDouble(val.toString().split("\t")[1]);
        Double PM10 = Double.parseDouble(val.toString().split("\t")[2]);
        Double NO2    = Double.parseDouble(val.toString().split("\t")[3]);
        Double SO2 = Double.parseDouble(val.toString().split("\t")[4]);
        Double O3 = Double.parseDouble(val.toString().split("\t")[5]);
        Double CO = Double.parseDouble(val.toString().split("\t")[6]);
        String dateString = val.toString().split("\t")[0];
        Double[] AP = {PM25,PM10,NO2,SO2,O3,CO};
        if (date_ap.containsKey(dateString)) {
                ArrayList<Double[]> ap_list = date_ap.get(dateString);
                ap_list.add(AP);
            }
        else{
                ArrayList<Double[]> ap_list = new ArrayList<Double[]>();
                ap_list.add(AP);
                date_ap.put(dateString,ap_list);
            }
    }
    // 计算每日各项污染物的平均水平并计算单项指数
    for(Map.Entry<String, ArrayList<Double[]>> entry : date_ap.entrySet()){
        double[] ap_sum = {0,0,0,0,0,0};
        ArrayList<Double[]> apList = entry.getValue();
        for (int i = 0; i < apList.size(); i++) {
            Double[] ap_i = apList.get(i);
            for (int j =0;j<ap_sum.length;j++){
                    ap_sum[j] = ap_sum[j] + ap_i[j];
                }
        }

        Double[] ap_avg_index = new Double[6];
        for (int i = 0; i<ap_sum.length;i++ ){
            ap_avg_index[i] = (ap_sum[i]/apList.size())/LIMITS[i];
        }
        apList.add(ap_avg_index);
    }
```

```
            Double[] city_index = new Double[12];
            HashMap<String,Integer> city_code = new HashMap();
            String[] citys =
            {"北京","上海","成都","天津",
             "青岛","济南", "厦门","郑州",
             "乌鲁木齐","呼和浩特","海口","昆明"};
            for(int t = 0; t < citys.length; t++){
                  city_code.put(citys[t], t);
            }
            for(Map.Entry<String, ArrayList<Double[]>> entry : date_ap.entrySet()){
                  ArrayList<Double[]> apList2 = entry.getValue();
                  Double[] ap_avg_index = apList2.get(apList2.size()-1) ;
                  for(int t = 0; t < ap_avg_index.length; t++){
                        city_index[city_code.get(city)] = ap_avg_index[t];
            }
}
            //String result = results.toString("\n");
            context.write(key, new Text((city_index[city_code.get(city)]).toString()));
}
}
```

3. TextArrayWritable.java

```
public class TextArrayWritable extends ArrayWritable {
      private Text[] myValue = new Text[0];

      public Text[] getMyValue() {
            return myValue;
      }
 public TextArrayWritable() {
 super(Text.class);
 }

 public TextArrayWritable(String[] strings) {
 super(Text.class);
 Text[] texts = new Text[strings.length];
 for (int i = 0; i < strings.length; i++) {
 texts[i] = new Text(strings[i]);
 }
 set(texts);
 }
  @Override
      public String toString() {

            StringBuffer result = new StringBuffer();
```

```
            for(int i =0; i < this.getMyValue().length; i++){
                if(i == this.getMyValue().length -1)
                    result.append(this.getMyValue()[i].toString());
                else
                    result.append(this.getMyValue()[i].toString()).append(",");
            }

            return result.toString();
        }
}
```

5.3 解决过程和运行结果截图展示

5.3.1 第一题运行结果

第一题中我首先遇到报错“java.lang.Exception: java.lang.NumberFormatException: For input string: "PM25"”。定位错误代码为“double PM25 = Double.parseDouble(fields[3]);”。该行代码本身没有问题，目的是将 String 转换为 double 类型，我的解决方法是在其上一行输出 fields[3]字段内容，即加一行“System.out.println(fields[3]);”。程序输出结果是“PM25”，为该变量的字段名。因此找到错误原因是，没有跳过第一行字段名称，需要加 if 判断语句。总结经验是应该细心，导入数据时要注意是否有表头字段。

其次我遇到中文乱码的问题，如图所示。首先排除字符编码的问题，具体是在 Linux 系统中通过“iconv -f gbk -tutf8 PM25city.txt > PM25city_utf8.txt”命令将文件从 gbk 转换为 UTF-8 编码，发现问题没有解决。进一步排查发现问题是 SecureCRT 编码导致的。改变 SecureCRT 编码为 UTF-8 后，问题解决。

图 3 中文乱码问题

```
[root@master ~]# hadoop fs -cat /user/root/AQIoutput/part-r-00000
20/07/09 02:25:54 WARN util.NativeCodeLoader: Unable to load native-
re applicable
ä¸Šæµ·  55
ä¹Œé²  æœ¨é½     105
åŒ—äº¬  70
åž¦é—¨  37
å¹¼å¿Œæµ©ç‰¹     55
å¤©æ´¥  72
æ˜†éƒ½  74
æ¿Ÿå—  32
æµŽå—  106
æµ·å£  22
éƒ‘å·ž  119
é¹¤å²›  78
```

第一题的结果如下图所示。以 PM25 的空气质量分指数（IAQI）为衡量指标，比较 2018 年 8 月至 2019 年 6 月间各城市的空气质量水平，结果表明：给定时间范围内，海口和昆明的 PM25 浓度最低，是最宜居的城市；厦门、呼和浩特、上海、北京以及天津的 PM25 浓度为第二梯队，是污染较轻的城市；成都、青岛、乌鲁木齐和济南的 PM25 浓度为第三梯队，城市污染程度不理想，

仍需注重污染物排放的环境管理；郑州的 PM25 浓度为第四梯队，污染较为严重，在注重经济发展的同时不能忽视环境保护。

图 4 第一题结果图（排序前）

```
[root@master ~]# hadoop fs -cat /user/root/AQIoutput1/part-r-00000
20/07/09 17:56:11 WARN util.NativeCodeLoader: Unable to load native-
re applicable
上海    55.18360058440512
乌鲁木齐        105.50207880024911
北京    70.24525967238151
厦门    37.90453666222592
呼和浩特        55.0762874278499
天津    72.85804564495083
成都    74.10958049198001
昆明    32.625129602441895
济南    106.56531048165962
海口    22.56553826042011
郑州    119.8591981616064
青岛    78.55928168903378
```

图 5 第一题结果图（排序后）

```
run:
海口    22.56553826042011
昆明    32.625129602441895
厦门    37.90453666222592
呼和浩特 55.0762874278499
上海    55.18360058440512
北京    70.24525967238151
天津    72.85804564495083
成都    74.10958049198001
青岛    78.55928168903378
乌鲁木齐 105.50207880024911
济南    106.56531048165962
郑州    119.8591981616064
成功构建（总时间：0 秒）
```

5.3.2 第二题运行结果

第二题的结果如下图所示。以北京、上海和成都三个城市为例，以 AQI 为分析指标，统计出春节期间三个城市的空气质量等级分布情况。结果表明，上海的空气质量水平优于北京，成都的空气质量水平最差。具体来看，上海的空气质量为优的天数最多；北京的中度污染以及重度污染的天数最多，空气质量相对较差；成都的大部分天气均为良，，空气质量为优的天数仅有两天，空气质量堪忧。

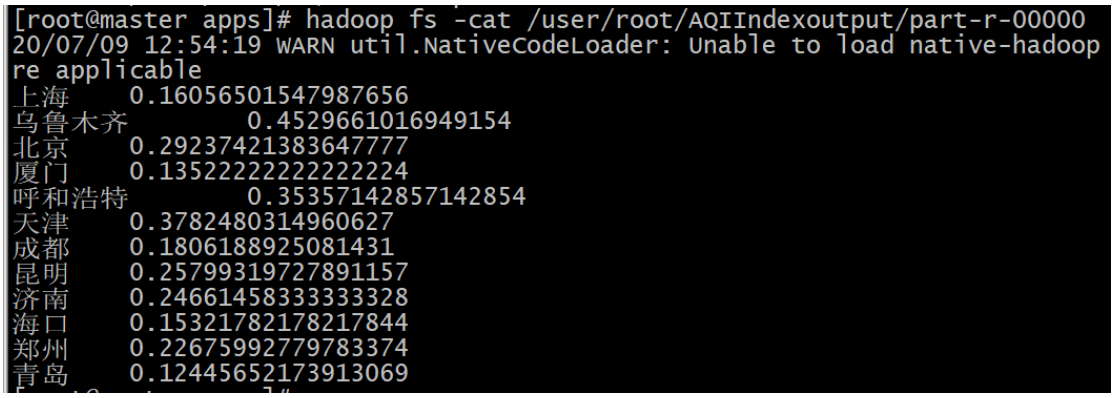
图 6 第二题结果图

```
[root@master apps]# hadoop fs -cat /user/root/AQICountoutput/part-r-00000
20/07/09 09:59:19 WARN util.NativeCodeLoader: Unable to load native-hadoop
re applicable
上海    [12, 12, 2, 2, 0, 0]
北京    [6, 11, 5, 5, 1, 0]
成都    [2, 22, 4, 0, 0, 0]
```

5.3.3 第三题运行结果

第三题的结果如下图所示。从图中可以看出，空气质量综合指数较低的有青岛、厦门、海口、成都、上海、昆明等，表明这些城市的空气质量综合水平较高。而空气质量综合指数较低的有乌鲁木齐、北京、呼和浩特、天津、郑州等，表明这些城市的空气质量综合水平较差。该结果也与上一题中 PM25 指标结果较为接近。

图 7 第三题结果图



5. 结论

本实验基于 MapReduce 框架完成，针对空气质量分析问题进行了较为全面的分析，也极大地提升了自己对 MapReduce 框架和 Java 语言的掌握水平。经过实践，笔者发现以下几点有待改进或完善的方面：

第一，在第二题部分，需求是求出给定时间范围内，给定城市下，以天为单位，统计不同 AQI 级别的天数分布。在解决思路的分析部分，笔者已剖析用 MapReduce 框架实现的两种思路。但本题极其适合用 Hive 实现，对代码的要求低，对时间和城市分别 group by，即可实现。因此在生产实践中，应当选最适合需求问题的工具和解法。

第二，本实验的不足之处是，空气质量还有很多影响因素，如当地的 GDP 水平、工业水平、企业排污状况等经济因素，和当时的风力、温度、湿度、气压等自然环境因素。若想更全面地找出影响空气质量的因素并进行机器学习建模预测，仍需更多的数据。老师以后可以在 Kaggle 等竞赛平台上下载数据，提供给同学们。

第三，除了基于 Java 语言学习 Hadoop 外，业界也多用 python 实现大数据处理。希望老师可以教授 Linux 中 python 环境的配置、python 机器学习在 Hadoop 中的实现。我自己在搭建环境、导入 python 库上遇到了一些困难，所以希望系统学习该方面的知识。

第四，最后非常感谢老师教我们搭建环境，这一点在我的实习中尤为受益。我的实习工作需要在 Linux 系统上配置 python 环境、MySQL 数据库、PostgreSQL 数据库，以及各种生产工具，现在配置环境、解决报错信息方面有了极大的提升。

6. 参考文献

1. http://www.mee.gov.cn/ywgz/fgbz/bz/bzwb/jcffbz/201203/t20120302_224166.shtml；“环境空气质量指数（AQI）技术规定（试行）”；2020-07-02。
2. https://kns.cnki.net/KCMS/detail/detail.aspx?dbcode=CJFQ&dbname=CJFD2014&filename=HLJO201402011&v=MjQ4MDFYMUx1eFlTN0RoMVQzcVRyV00xRnJDVVI3cWZZT1ptRnkzbFVyckFMU0hCWWJHNEg5WE1yWTlFWllSOGU=；“如何理解空气质量分指数(IAQI)计算公式并速算”；2020-07-02。
3. https://jingyan.baidu.com/album/19192ad8ec8837e53e5707ef.html?picindex=2；SecureCRT 中文乱码解决方法；2020-07-02。
4. https://cloud.tencent.com/developer/article/1414431；“MapReduce 之输出结果排序”；2020-07-04。
5. https://zh.wikipedia.org/wiki/%E7%A9%BA%E6%B0%94%E8%B4%A8%E9%87%8F%E6%8C%87%E6%95%B0；“空气质量指数”；2020-07-04。
6. https://blog.csdn.net/FORLOVEHUAN/article/details/102956833；“空气质量综合指数 JAVA 算法”；2020-07-06。
7. http://aidata100.com/bigdata/zppic/bd0211.pdf；“MapReduce 理论”；2020-07-07。
8. http://www.wenshui.gov.cn/zmhd/zsk/201805/t20180529_603362.html；“空气质量综合指数解读”；2020-07-07。
9. https://baike.baidu.com/item/%E7%A9%BA%E6%B0%94%E8%B4%A8%E9%87%8F%E7%BB%BC%E5%90%88%E6%8C%87%E6%95%B0/20840551；“空气质量综合指数”；2020-07-07。