# 前言：

本文档的总结是基于“优化日志文件统计顺序”：读取一个序列化文件，根据要求统计2016年1月份和2月份每天的登录次数，并且将最终的输出结果根据月份保存到两个不同的文件中。

# 1.自定义键值类型封装属性（Pojo）

import java.io.DataInput;  
import java.io.DataOutput;  
import java.io.IOException;  
import org.apache.hadoop.io.WritableComparable;  
  
public class MemberLogTime implements WritableComparable<MemberLogTime>{  
 private String member\_name; // 用户名  
 private String logTime; // 登录日期

public MemberLogTime() {  
 }  
 public MemberLogTime(String member\_name,String logTime){  
 this.member\_name=member\_name;  
 this.logTime=logTime;  
 }  
 public String getMember\_name() {  
 return member\_name;  
 }  
 public void setMember\_name(String member\_name) {  
 this.member\_name = member\_name;  
 }  
 public String getLogTime() {  
 return logTime;  
 }  
 public void setLogTime(String logTime) {  
 this.logTime = logTime;  
 }

// 反序列化方法，注意和序列化的顺序一致  
 @Override  
 public void readFields(DataInput in) throws IOException {  
 this.member\_name=in.readUTF();  
 this.logTime=in.readUTF();  
 }  
 // 输出的时候要先序列化

@Override  
 public void write(DataOutput out) throws IOException {  
 out.writeUTF(member\_name);  
 out.writeUTF(logTime);  
 }  
 @Override  
 public int compareTo(MemberLogTime o) {

// 这里只是根据用户名进行排序，也可以根据返回值嵌套比较

// 用当前的用户名与属性的中的用户名比较:

// 如果指定的数与参数差是 0;

// 如果指定的数与参数差是 -1;

// 如果指定的数与参数差是 1;  
 return this.getMember\_name() - o.getMember\_name();  
 }  
 @Override  
 public String toString() {  
 return this.member\_name+","+this.logTime;  
 }  
}

# 2.Mapper

import java.io.IOException;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.io.Text;  
import org.apache.hadoop.mapreduce.Mapper;

public class LogCountMapper extends Mapper<Text, Text, MemberLogTime, IntWritable> {  
 private MemberLogTime mt = new MemberLogTime();  
 private IntWritable one = new IntWritable(1);

// 定义两个计数变量，然后替换下边那个枚举类型的计数器

// int num1=0; int num2=0;

// 枚举类型的名称即组的名称，枚举类型的字段是计数器的名称  
 enum LogCounter{  
 *January*,  
 *February* }  
 @Override  
 protected void map(Text key, Text value, Context context)  
 throws IOException, InterruptedException {  
 String member\_name=key.toString();  
 String logTime=value.toString();  
 if(logTime.contains("2016-01")){

// mum1++;  
 context.getCounter(LogCounter.*January*).increment(1);  
 }else if(logTime.contains("2016-02")){

// num2++;  
 context.getCounter(LogCounter.*February*).increment(1);  
 }  
 mt.setMember\_name(member\_name);  
 mt.setLogTime(logTime);  
 context.write(mt, one);  
 }  
}

# 3.Combiner

**Combiner是一个迷你的Reducer，它只处理单台机器生成的数据，它发生在Map端，会接收特定节点上的Map作为输入，对Map的数据先做一次合并，**

**再把输出结果发送到Reducer。注意，Combiner的加入不影响原逻辑，影响的只是效率。例如在求和、求最大值、最小值的时候可以考虑使用Combiner.**

**该过程的实现就是对每一个Map任务的输出做一次输出**

import java.io.IOException;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.mapreduce.Reducer;  
  
public class LogCountCombiner extends Reducer<MemberLogTime, IntWritable, MemberLogTime, IntWritable> {  
 @Override  
 protected void reduce(MemberLogTime key, Iterable<IntWritable> value, Context context)  
 throws IOException, InterruptedException {  
 int sum=0;  
 for (IntWritable val : value) {  
 sum+=val.get();  
 }  
 context.write(key, new IntWritable(sum));  
 }  
}

# 4.Partitioner

**在进行MapReduce计算时，有时候需要把最终的输出数据分到不同的文件中，比如按照省份划分的话，需要把同一省份的数据放到一个文件中；按照性别划分的话，需要把同一性别的数据放到一个文件中。我们知道最终的输出数据是来自于Reducer任务。那么，如果要得到多个文件，意味着有同样数量的Reducer任务在运行。Reducer任务的数据来自于Mapper任务，也就说Mapper任务要划分数据，对于不同的数据分配给不同的Reducer任务运行。Mapper任务划分数据的过程就称作Partition。负责实现划分数据的类称作Partitioner。**

**Partitioner组件的功能是让让Map对Key进行分区，从而将key分发到不同的Reducer中进行处理。一般情况下MR会默认使用HashPartitioner分区，但这里是特殊要求的，1月份的输出的一个文件中，2月份的输出到另一个文件中。这个时候就可以自定义Partitioner来实现这个要求。**

import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.mapreduce.Partitioner;

public class LogCountPartitioner extends Partitioner<MemberLogTime, IntWritable> {  
 @Override  
 public int getPartition(MemberLogTime key, IntWritable value, int numPartitions) {  
 String date=key.getLogTime();  
 if(date.contains("2016-01")){  
 return 0 % numPartitions;  
 }else{  
 return 1 % numPartitions;  
 }  
 }  
}

# 5.Reducer

import java.io.IOException;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.mapreduce.Reducer;

public class LogCountReducer extends Reducer<MemberLogTime, IntWritable, MemberLogTime, IntWritable> {  
 @Override  
 protected void reduce(MemberLogTime key, Iterable<IntWritable> value,Context context)  
 throws IOException, InterruptedException {  
 if(key.getLogTime().contains("2016-01")){  
 context.getCounter("OutputCounter","JanuaryResult").increment(1);  
 }else if(key.getLogTime().contains("2016-02")){  
 context.getCounter("OutputCounter","FebruaryResult").increment(1);  
 }  
 int sum=0;  
 for (IntWritable val : value) {  
 sum+=val.get();  
 }  
 context.write(key, new IntWritable(sum));  
 }  
}

# 6.Main

import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.conf.Configured;  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.IntWritable;  
import org.apache.hadoop.mapreduce.Job;  
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;  
import org.apache.hadoop.mapreduce.lib.input.SequenceFileAsTextInputFormat;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;  
import org.apache.hadoop.util.Tool;  
import org.apache.hadoop.util.ToolRunner;

public class LogCount extends Configured implements Tool{  
 public static void main(String[] args){  
 String[] myArgs={  
 "/user/root/user\_login/JanFeb/part-m-00000",  
 "/user/root/user\_login/logcount"  
 };  
  
 try {  
 // 利用ToolRunner工具类传入配置对象、主类、参数列表  
 ToolRunner.*run*(LogCount.getMyConfiguration(),

new LogCount(), myArgs);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }

@Override  
 public int run(String[] args) throws Exception {  
 Configuration conf=LogCount.*getMyConf*();  
 Job job=Job.*getInstance*(conf);  
 job.setJarByClass(LogCount.class);  
 job.setMapperClass(LogCountMapper.class);  
 job.setReducerClass(LogCountReducer.class);

// 加入Combiner  
 job.setCombinerClass(LogCountCombiner.class);   
 // 加入Partitioner

job.setPartitionerClass(LogCountPartitioner.class);

// 设置Reducer的个数是2  
 job.setNumReduceTasks(2);   
 job.setOutputKeyClass(MemberLogTime.class);  
 job.setOutputValueClass(IntWritable.class);  
 job.setInputFormatClass(SequenceFileAsTextInputFormat.class);  
 job.setOutputFormatClass(TextOutputFormat.class);

// 文件输入路径  
 FileInputFormat.*addInputPath*(job, new Path(args[0]));   
 FileSystem.*get*(conf).delete(new Path(args[1]), true);

// 文件输入路径  
 FileOutputFormat.*setOutputPath*(job, new Path(args[1]));   
 return job.waitForCompletion(true)?-1:1;  
 }  
 public static Configuration getMyConf(){  
 //声明配置  
 Configuration conf = new Configuration();

// 指定namenode,建立与hdfs的连接  
 conf.set("fs.defaultFS", "hdfs://lky01:9000");   
 return conf;  
 }  
}