

MapReduce编程模型

一、课前准备

1. 准备3节点hadoop集群
2. windows下hadoop环境配置好
3. windows下安装IDEA编程工具
4. 安装maven并配置环境变量

二、课堂主题

1. 讲解MapReduce分布式计算框架的原理及使用

三、课堂目标

1. 理解MapReduce编程模型
2. 独立完成一个MapReduce程序并运行成功
3. 了解MapReduce工程流程
4. 掌握并描述出shuffle全过程（面试）
5. 独立编写课堂及作业中的MR程序
6. 理解并解决数据倾斜

四、知识要点

1. MapReduce编程模型（10分钟）

- Hadoop架构图

Hadoop由HDFS分布式存储、MapReduce分布式计算、Yarn资源调度三部分组成

Hadoop 2.X & Hadoop 3.X

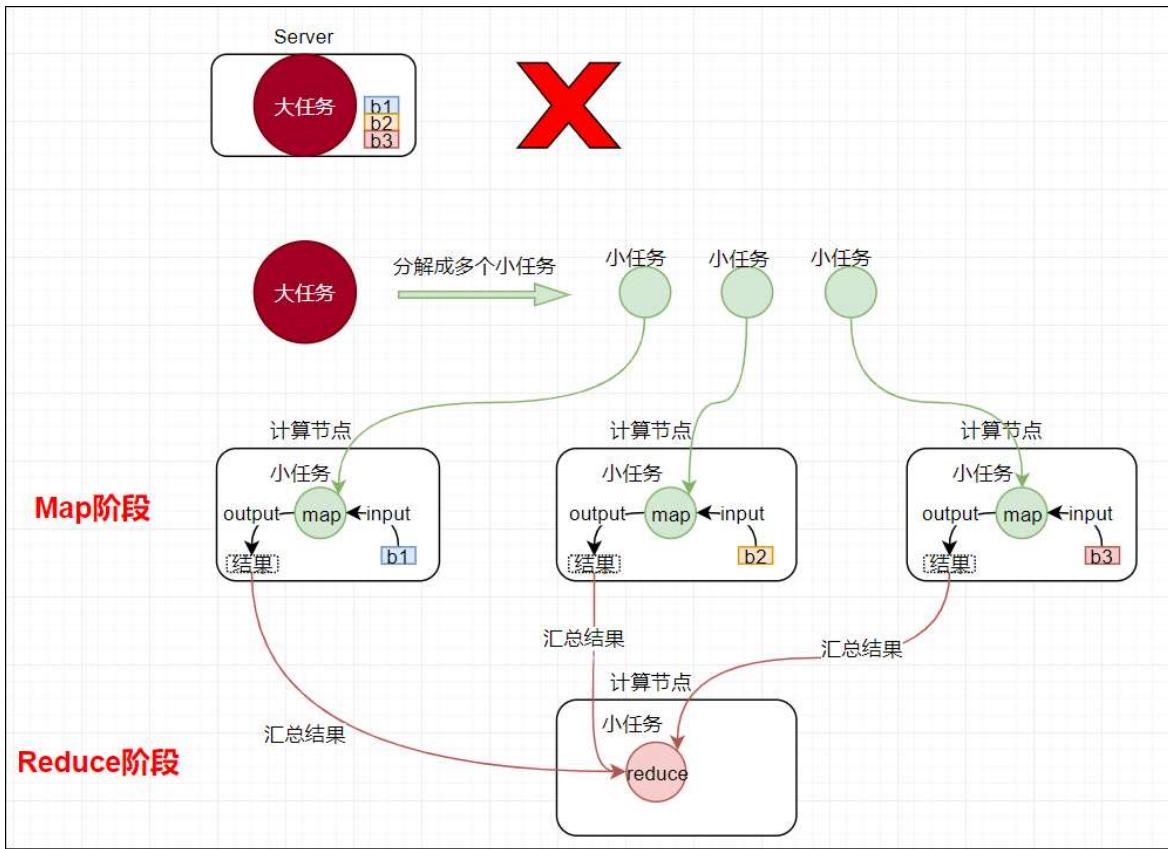
MapReduce
(data processing)

Others
(data processing)

YARN
(cluster resource management)

HDFS
(redundant, reliable storage)

- MapReduce是采用一种**分而治之**的思想设计出来的分布式计算框架
- MapReduce由两个阶段组成：
 - Map阶段（切分成一个个小的任务）
 - Reduce阶段（汇总小任务的结果）
- 那什么是分而治之呢？
 - 比如一复杂、计算量大、耗时长的任务，暂且称为“大任务”；
 - 此时使用单台服务器无法计算或较短时间内计算出结果时，可将此大任务切分成一个个小的任务，小任务分别在不同的服务器上**并行**的执行
 - 最终再汇总每个小任务的结果



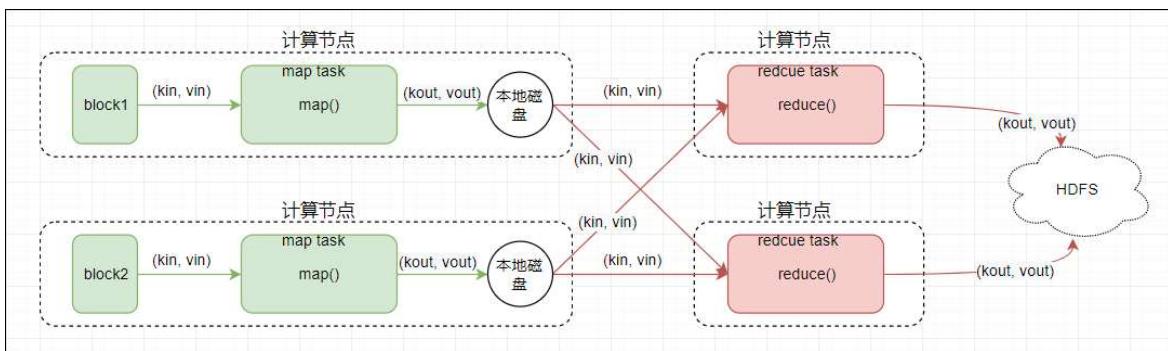
1.1 Map阶段

- map阶段有一个关键的map()函数；
- 此函数的输入是键值对
- 输出是一系列键值对，输出写入本地磁盘。

1.2 Reduce阶段

- reduce阶段有一个关键的函数reduce()函数
- 此函数的输入也是键值对（即map的输出（kv对））
- 输出也是一系列键值对，结果最终写入HDFS

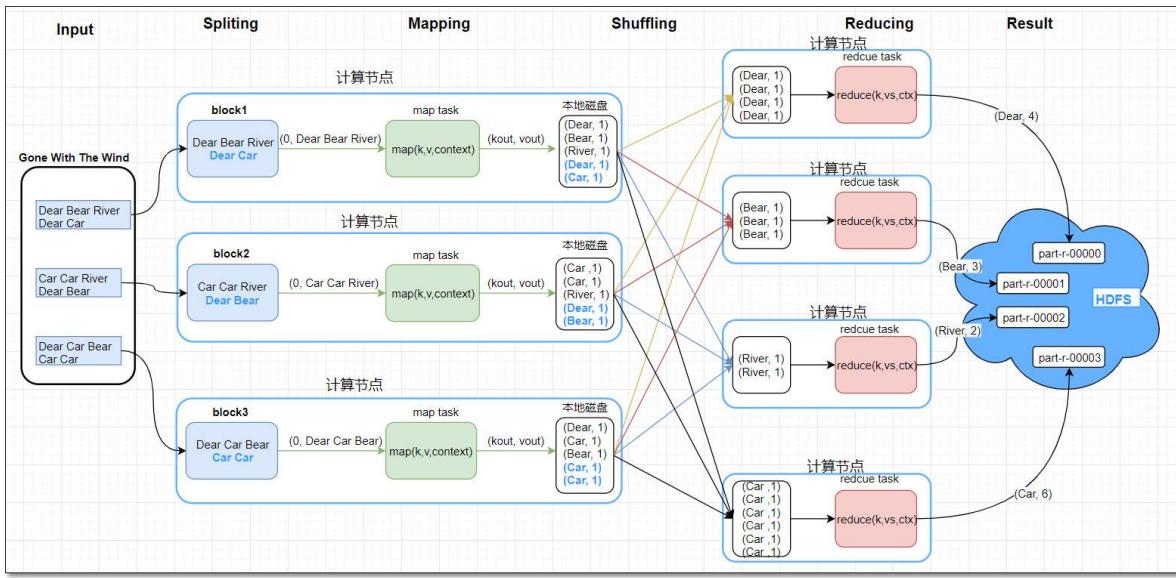
1.3 Map&Reduce



2. MapReduce编程示例 (重点 40分钟)

- 以MapReduce的词频统计为例：统计一批英文文章当中，每个单词出现的总次数

2.1 MapReduce原理图 (10分钟)



- Map阶段

- 假设MR的输入文件“**Gone With The Wind**”有三个block; block1、block2、block3
- MR编程时，每个block对应一个分片split
- 每一个split对应一个map任务 (map task)
- 如图共3个map任务 (map1、map2、map3)；这3个任务的逻辑一样，所以以第一个map任务 (map1) 为例分析
- map1读取block1的数据；一次读取block1的一行数据;
 - 产生键值对(key/value)，作为map()的参数传入，调用map();
 - 假设当前所读行是第一行
 - 将当前所读行的行首相对于当前block开始处的字节偏移量作为key (0)
 - 当前行的内容作为value (Dear Bear River)
- map()内
 - (按需求，写业务代码)，将value当前行内容按空格切分，得到三个单词Dear | Bear | River
 - 将每个单词变成键值对，输出出去(Dear, 1) | (Bear, 1) | (River, 1); 最终结果写入map任务所在节点的本地磁盘中（内里还有细节，讲到shuffle时，再细细展开）
 - block的第一行的数据被处理完后，接着处理第二行；逻辑同上
 - 当map任务将当前block中所有的数据全部处理完后，此map任务即运行结束
- 其它的每一个map任务都是如上逻辑，不再赘述

- Reduce阶段

- reduce任务 (reduce task) 的个数由自己写的程序编程指定，main()内的 job.setNumReduceTasks(4)指定reduce任务是4个 (reduce1、reduce2、reduce3、reduce4)
- 每一个reduce任务的逻辑一下，所以以第一个reduce任务 (reduce1) 为例分析
- map1任务完成后，reduce1通过网络，连接到map1，将map1输出结果中属于reduce1的分区的数据，通过网络获取到reduce1端 (拷贝阶段)
- 同样也如此连接到map2、map3获取结果
- 最终reduce1端获得4个(Dear, 1)键值对；由于key键相同，它们分到同一组；
- 4个(Dear, 1)键值对，转换成[Dear, Iterable(1, 1, 1,)]，作为两个参数传入reduce()
- 在reduce()内部，计算Dear的总数为4，并将(Dear, 4)作为键值对输出
- 每个reduce任务最终输出文件（内里还有细节，讲到shuffle时，再细细展开），文件写入到HDFS

2.2 MR中key的作用 (5分钟)

- **MapReduce编程中，key有特殊的作用**

- ①数据中，若要针对某个值进行分组、聚合时，需将此值作为MR中的reduce的输入的key
- 如当前的词频统计例子，按单词进行分组，每组中对出现次数做聚合（计算总和）；所以需要将每个单词作为reduce输入的key，MapReduce框架自动按照单词分组，进而求出每组即每个单词的总次数

```
word → num
dear → 1
bear → 1
river → 1
river → 1
dear → 1
...
select word, count(*) from novel group by word;
```

- ②另外，key还具有可排序的特性，因为MR中的key类需要实现WritableComparable接口；而此接口又继承Comparable接口（可查看源码）
- MR编程时，要充分利用以上两点；结合实际业务需求，设置合适的key

```
50   @Stringable
51   @InterfaceAudience.Public
52   @InterfaceStability.Stable
53   public class Text extends BinaryComparable
54     implements WritableComparable<BinaryComparable> {
55
56
71   @InterfaceAudience.Public
72   @InterfaceStability.Stable
73   public interface WritableComparable<T> extends Writable, Comparable<T> {
74   }
75 }
```

2.3 创建MAVEN工程

所有编程操作，在hadoop集群某节点的IDEA中完成

- 使用IDEA创建maven工程
- pom文件参考提供的pom.xml，主要用到的dependencies有

```
<properties>
    <cdh.version>2.6.0-cdh5.14.2</cdh.version>
</properties>

<repositories>
    <repository>
        <id>cloudera</id>
        <url>https://repository.cloudera.com/artifactory/cloudera-
repos/</url>
    </repository>
</repositories>

<dependencies>

    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
        <version>2.6.0-mr1-cdh5.14.2</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>${cdh.version}</version>
    </dependency>
```

```

<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-hdfs</artifactId>
    <version>${cdh.version}</version>
</dependency>

<dependency>
    <groupId>org.apache.hadoop</groupId>
    <artifactId>hadoop-mapreduce-client-core</artifactId>
    <version>${cdh.version}</version>
</dependency>
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>RELEASE</version>
    <scope>test</scope>
</dependency>

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.38</version>
    <scope>compile</scope>
</dependency>
</dependencies>

```

2.4 MR参考代码（15分钟）

- 创建包com.kaikeba.hadoop.wordcount
- 在包中创建自定义mapper类、自定义reducer类、包含main类

2.4.1 Mapper代码

```

package com.kaikeba.hadoop.wordcount;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

/**
 * 类Mapper<LongWritable, Text, Text, IntWritable>的四个泛型分别表示
 * map方法的输入的键的类型kin、值的类型vin; 输出的键的类型kout、输出的值的类型vout
 * kin指的是当前所读行行首相对于split分片开头的字节偏移量，所以是long类型，对应序列化类型
 * LongWritable
 * vin指的是当前所读行，类型是String，对应序列化类型Text
 * kout根据需求，输出键指的是单词，类型是String，对应序列化类型是Text
 * vout根据需求，输出值指的是单词的个数，1，类型是int，对应序列化类型是IntWritable

```

```

/*
 */
public class wordCountMap extends Mapper<LongWritable, Text, Text, IntWritable>
{

    /**
     * 处理分片split中的每一行的数据；针对每行数据，会调用一次map方法
     * 在一次map方法调用时，从一行数据中，获得一个个单词word，再将每个单词word变成键值对形式
     *(word, 1)输出出去
     * 输出的值最终写到本地磁盘中
     * @param key 当前所读行行首相对于split分片开头的字节偏移量
     * @param value 当前所读行
     * @param context
     * @throws IOException
     * @throws InterruptedException
     */
    public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
        //当前行的示例数据(单词间空格分割): Dear Bear River
        //取得当前行的数据
        String line = value.toString();
        //按照\t进行分割，得到当前行所有单词
        String[] words = line.split("\t");

        for (String word : words) {
            //将每个单词word变成键值对形式(word, 1)输出出去
            //同样，输出前，要将kout, vout包装成对应的可序列化类型，如String对应Text, int
            对应IntWritable
            context.write(new Text(word), new IntWritable(1));
        }
    }
}

```

2.4.2 Reducer代码

```

package com.kaikeba.hadoop.wordcount;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;

/**
 *
 * Reducer<Text, IntWritable, Text, IntWritable>的四个泛型分别表示
 * reduce方法的输入的键的类型kin、输入值的类型vin；输出的键的类型kout、输出的值的类型vout
 * 注意：因为map的输出作为reduce的输入，所以此处的kin、vin类型分别与map的输出的键类型、值类型
 * 相同
 * kout根据需求，输出键指的是单词，类型是String，对应序列化类型是Text
 * vout根据需求，输出值指的是每个单词的总个数，类型是int，对应序列化类型是IntWritable
 *
 */
public class wordCountReduce extends Reducer<Text, IntWritable, Text,
IntWritable> {
    /**
     */

```

```

    *
    * key相同的一组kv对，会调用一次reduce方法
    * 如reduce task汇聚了众多的键值对，有key是hello的键值对，也有key是spark的键值对，如
    下
    * (hello, 1)
    * (hello, 1)
    * (hello, 1)
    * (hello, 1)
    * ...
    * (spark, 1)
    * (spark, 1)
    * (spark, 1)
    *
    * 其中，key是hello的键值对被分成一组；merge成[hello, Iterable(1,1,1,1)]，调用一次
    reduce方法
    * 同样，key是spark的键值对被分成一组；merge成[spark, Iterable(1,1,1)]，再调用一次
    reduce方法
    *
    * @param key 当前组的key
    * @param values 当前组中，所有value组成的可迭代集和
    * @param context reduce上下文环境对象
    * @throws IOException
    * @throws InterruptedException
    */
    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context) throws IOException,
InterruptedException {
    //定义变量，用于累计当前单词出现的次数
    int sum = 0;

    for (IntWritable count : values) {
        //从count中获得值，累加到sum中
        sum += count.get();
    }

    //将单词、单词次数，分别作为键值对，输出
    context.write(key, new IntWritable(sum)); // 输出最终结果
}
}

```

2.4.3 Main程序入口

```

package com.kaikeba.hadoop.wordcount;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;

/**
 *
 * MapReduce程序入口
 * 注意：

```

```
* 导包时，不要导错了；  
* 另外，map\reduce相关的类，使用mapreduce包下的，是新API，如  
org.apache.hadoop.mapreduce.Job;  
*/  
public class WordCountMain {  
    //若在IDEA中本地执行MR程序，需要将mapred-site.xml中的mapreduce.framework.name值修  
改成local  
    //参数 c:/test/README.txt c:/test/wc  
    public static void main(String[] args) throws IOException,  
        ClassNotFoundException, InterruptedException {  
  
        //判断一下，输入参数是否是两个，分别表示输入路径、输出路径  
        if (args.length != 2 || args == null) {  
            System.out.println("please input Path!");  
            System.exit(0);  
        }  
  
        Configuration configuration = new Configuration();  
        //configuration.set("mapreduce.framework.name","local");  
  
        //告诉程序，要运行的jar包在哪  
  
        //configuration.set("mapreduce.job.jar","/home/hadoop/IdeaProjects/Hadoop/targe  
t/com.kaikeba.hadoop-1.0-SNAPSHOT.jar");  
  
        //调用getInstance方法，生成job实例  
        Job job = Job.getInstance(configuration,  
WordCountMain.class.getSimpleName());  
  
        //设置job的jar包，如果参数指定的类包含在一个jar包中，则此jar包作为job的jar包；参  
数class跟主类在一个工程即可；一般设置成主类  
//      job.setJarByClass(WordCountMain.class);  
        job.setJarByClass(WordCountMain.class);  
  
        //通过job设置输入/输出格式  
        //MR的默认输入格式是TextInputFormat，输出格式是TextOutputFormat；所以下两行可以  
注释掉  
//      job.setInputFormatClass(TextInputFormat.class);  
//      job.setOutputFormatClass(TextOutputFormat.class);  
  
        //设置输入/输出路径  
        FileInputFormat.setInputPaths(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
  
        //设置处理Map阶段的自定义的类  
        job.setMapperClass(WordCountMap.class);  
        //设置map combine类，减少网路传出量  
        job.setCombinerClass(WordCountReduce.class);  
        //设置处理Reduce阶段的自定义的类  
        job.setReducerClass(WordCountReduce.class);  
        //注意：如果map、reduce的输出的kv对类型一致，直接设置reduce的输出的kv对就行；如果  
不一样，需要分别设置map、reduce的输出的kv类型  
        //注意：此处设置的map输出的key/value类型，一定要与自定义map类输出的kv对类型一致；否  
则程序运行报错  
//      job.setMapOutputKeyClass(Text.class);  
//      job.setMapOutputValueClass(IntWritable.class);
```

```

        //设置reduce task最终输出key/value的类型
        //注意：此处设置的reduce输出的key/value类型，一定要与自定义reduce类输出的kv对类型
一致；否则程序运行报错
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        // 提交作业
        job.waitForCompletion(true);

    }

}

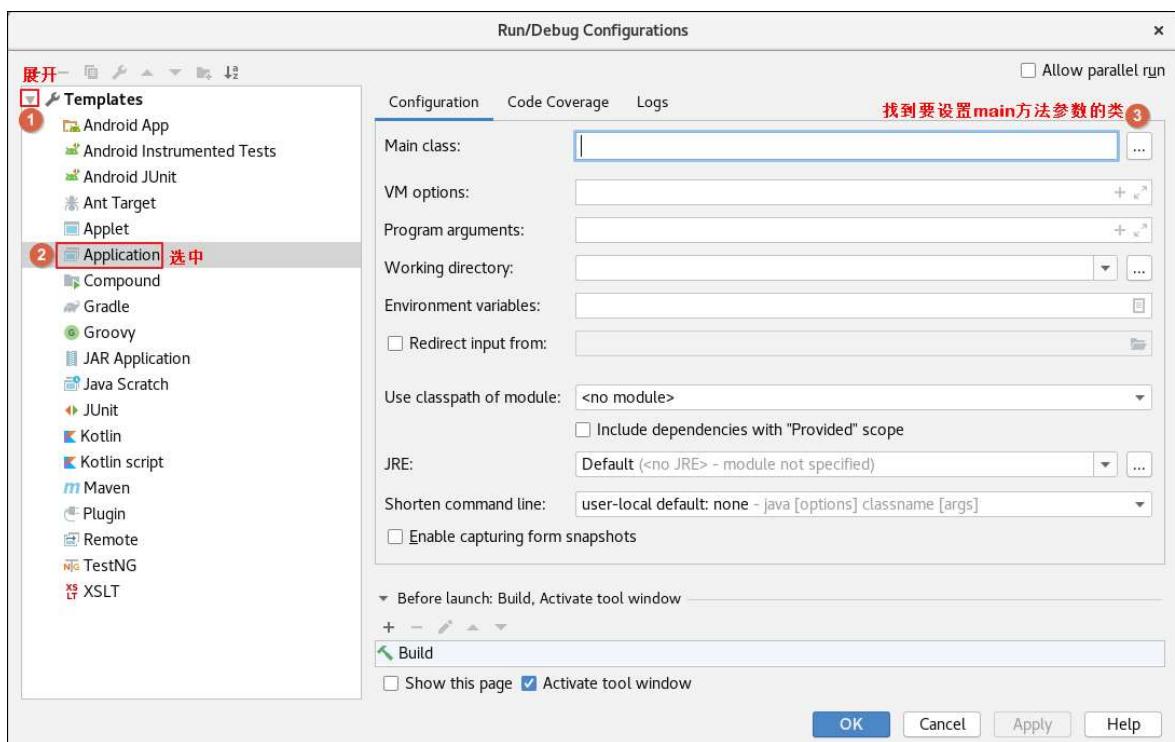
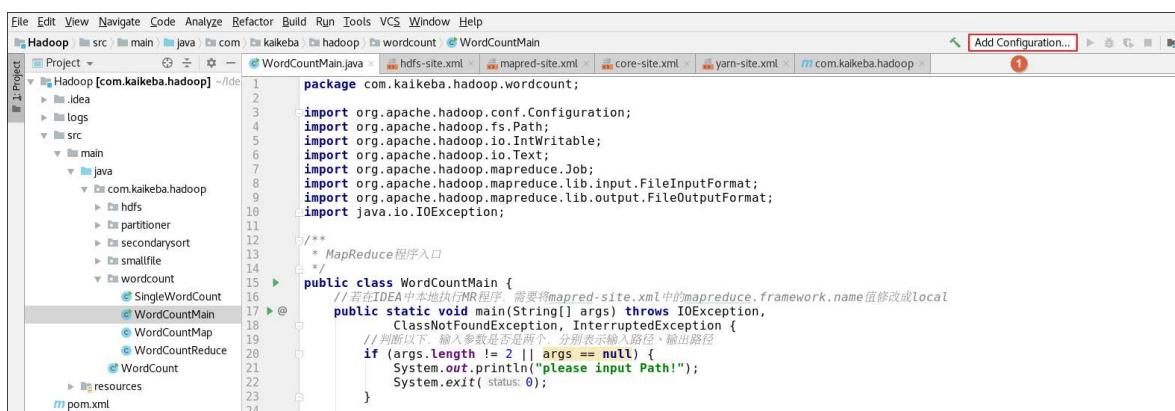
```

程序运行有两种方式，分别是windows本地运行、集群运行，依次演示

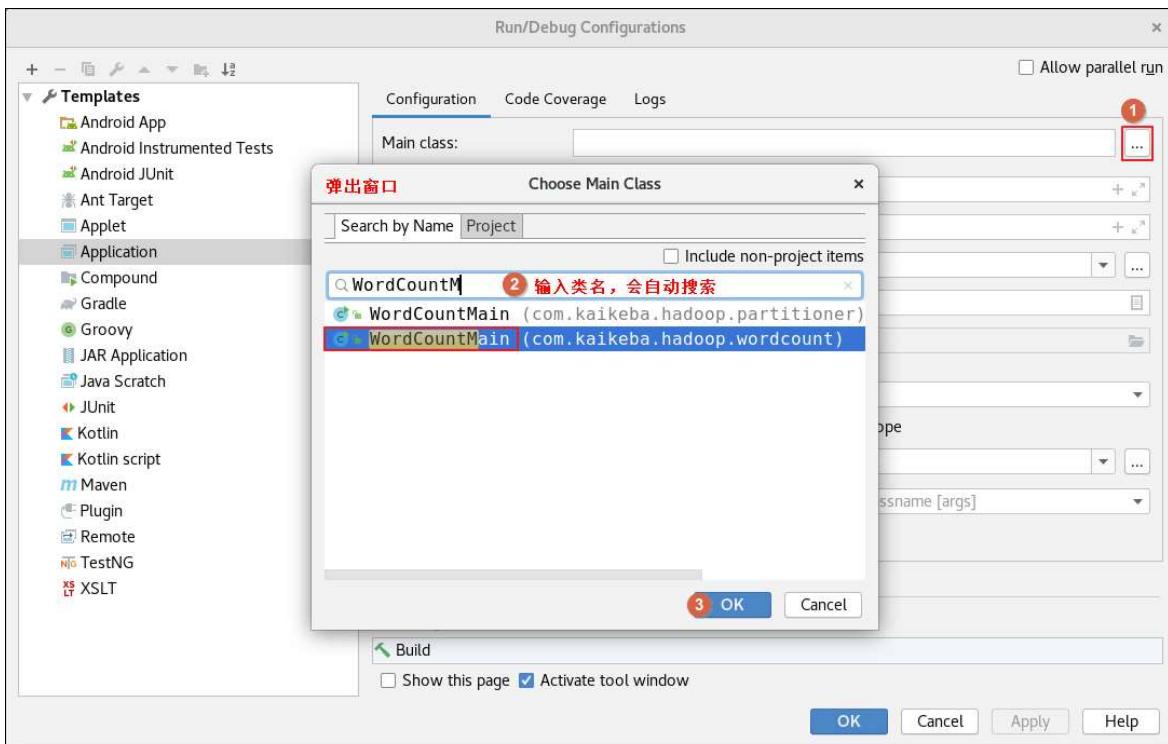
2.5 本地运行（5分钟）

在windows的IDEA中运行

2.5.1 初次运行WordCountMain，先设置main方法参数，根据图示操作即可



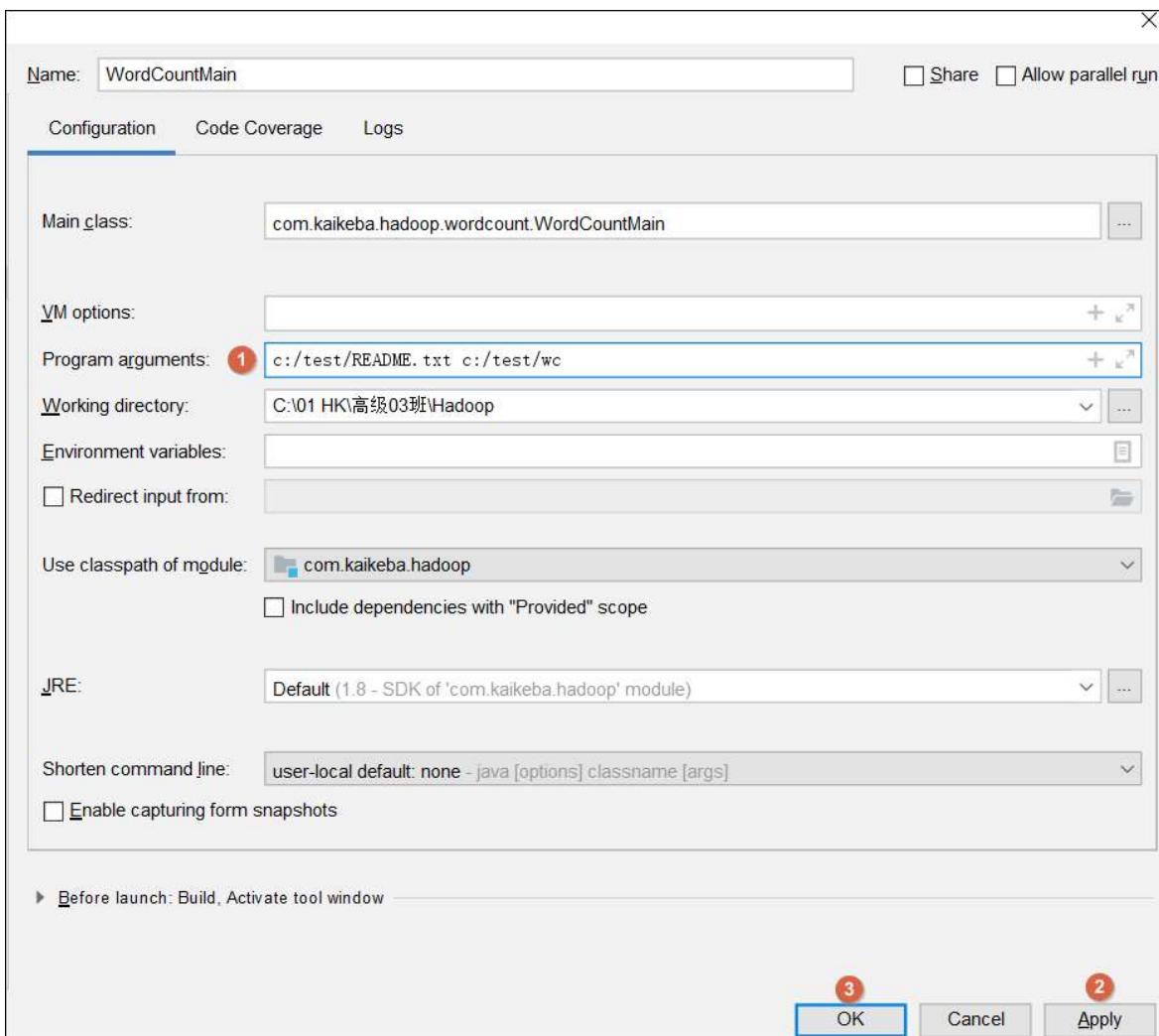
- 弹出窗口中，设置包含main方法的类



- 设置main方法在参数

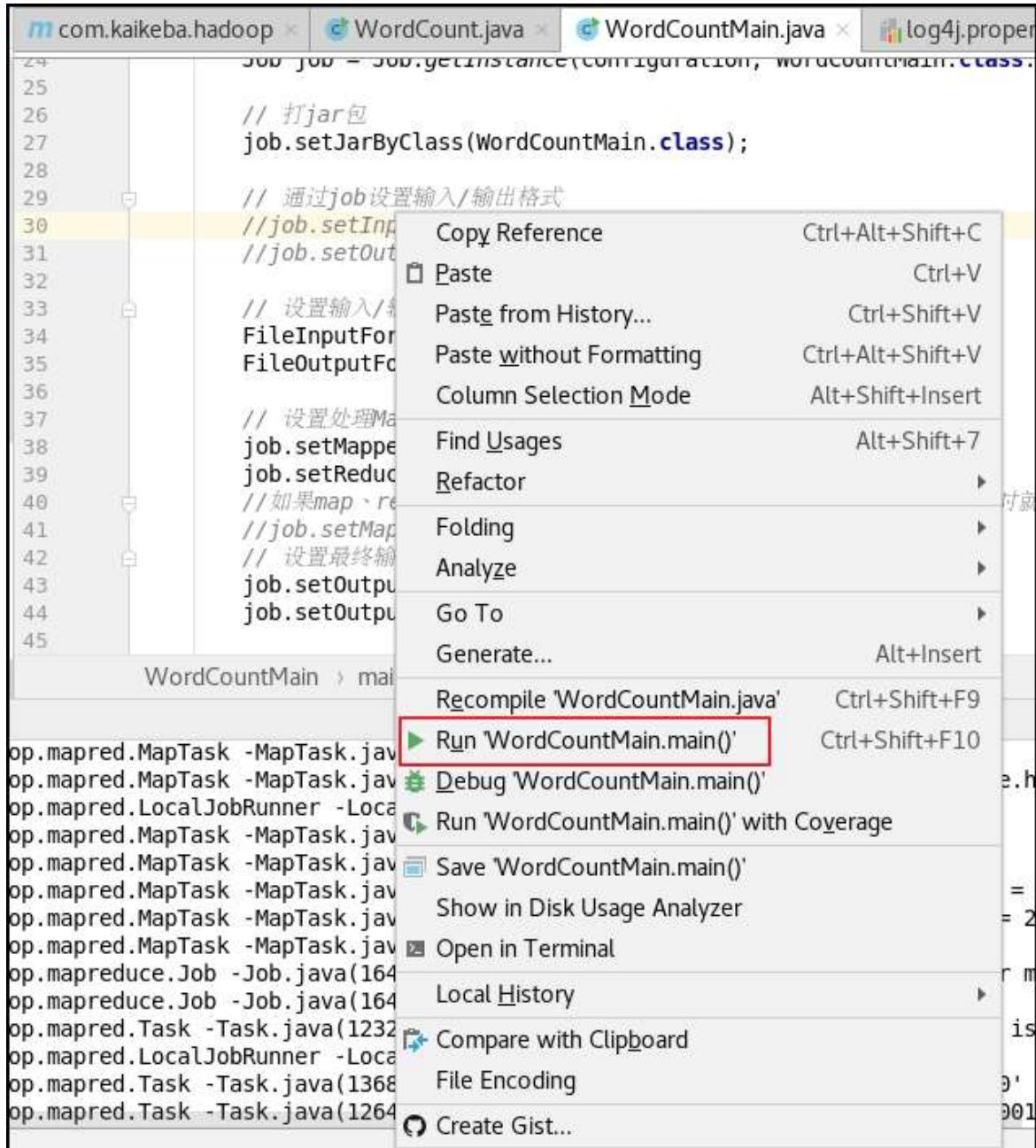
注意：两个参数间有一个英文空格，表示两个参数

```
c:/test/README.txt c:/test/wc
```



2.5.3 本地运行程序

- 在WordCountMain代码上，点击鼠标右键，运行程序



2.5.4 查看结果

①成功标识文件

②结果文件

此电脑 > Windows (C:) > test > wc			
名称	修改日期	类型	大小
.SUCCESS.crc	2019/10/14 12:06	CRC 文件	1 KB
.part-r-00000.crc	2019/10/14 12:06	CRC 文件	1 KB
SUCCESS ①	2019/10/14 12:06	文件	0 KB
part-r-00000 ②	2019/10/14 12:06	文件	2 KB

2.6 集群运行 (5分钟)

- 用maven插件打jar包；①点击Maven，②双击package打包

```

16 // 若在IDEA中本地执行MR程序，需要将mapred-site.xml中的mapreduce.framework.name修改为yarn
17 public static void main(String[] args) throws IOException,
18     ClassNotFoundException, InterruptedException {
19     // 判断以下，输入参数是否是两个，分别表示输入路径、输出路径
20     if (args.length != 2 || args == null) {
21         System.out.println("please input Path!");
22         System.exit( status: 0);
23     }
24
25     Configuration configuration = new Configuration();
26     //configuration.set("mapreduce.job.jar", "/home/hadoop/project");
27
28     // 调用getInstance方法，生成job实例
29     Job job = Job.getInstance(configuration, WordCountMain.class);
30
31     // 设置jar包，参数是包含main方法的类
32     job.setJarByClass(WordCountMain.class);
33
34     // 通过job设置输入/输出格式
35     // MR的默认输入格式是TextInputFormat，所以下两行可以注释掉
36     // job.setInputFormatClass(TextInputFormat.class);
37     // job.setOutputFormatClass(TextOutputFormat.class);
38
39     // 设置输入/输出路径
40     FileInputFormat.setInputPaths(job, new Path(args[0]));
41     FileOutputFormat.setOutputPath(job, new Path(args[1]));
42
43     // 设置处理Map阶段的自定义的类
44     job.setMapperClass(WordCountMap.class);
45     // 设置map combine类，减少网路传出量
46     // job.setCombinerClass(WordCountReduce.class);

```

- 控制台打印结果；①表示打包成功；②是生成的jar所在路径

```

[Hadoop] pom.xml
Project Run: com.kaikeba.hadoop [package]
[WARNING] otherwise try to manually exclude artifacts based on
[WARNING] mvn dependency:tree -Ddetail=true and the above output.
[WARNING] See http://maven.apache.org/plugins/maven-shade-plugin/
[INFO] Minimized 27811 → 13782 (4%)
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing C:\01 HK\高级03班\Hadoop\target\com.kaikeba.hadoop-1.0-SNAPSHOT.jar → C:\01 HK\高级03班\Hadoop\target\com.kaikeba.hadoop-1.0-SNAPSHOT-shaded.jar
[INFO] Dependency-reduced POM written at: C:\01 HK\高级03班\Hadoop\dependency-reduced-pom.xml
[INFO] Dependency-reduced POM written at: C:\01 HK\高级03班\Hadoop\dependency-reduced-pom.xml
[INFO] -----
[INFO] BUILD SUCCESS ①
[INFO] -----
[INFO] Total time: 18.582 s
[INFO] Finished at: 2019-10-14T12:17:46+08:00
[INFO] Final Memory: 43M/1269M
[INFO] -----
Process finished with exit code 0

```

- 将jar包上传到node01用户主目录/home/hadoop下
- 用hadoop jar命令运行mr程序

```

[hadoop@node01 ~]$ cd
[hadoop@node01 ~]$ hadoop jar com.kaikeba.hadoop-1.0-SNAPSHOT.jar
com.kaikeba.hadoop.wordcount.WordCountMain /README.txt /wordcount01

```

说明：

com.kaikeba.hadoop-1.0-SNAPSHOT.jar是jar包名

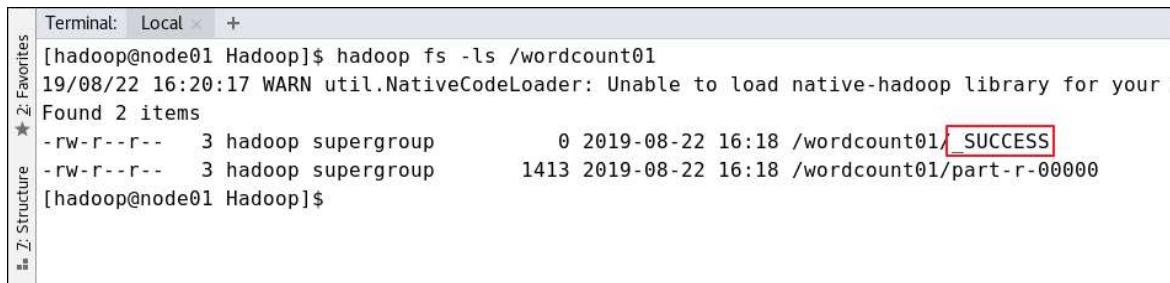
com.kaikeba.hadoop.wordcount.WordCountMain是包含main方法的类的全限定名

/NOTICE.txt和/wordcount是main方法的两个参数，表示输入路径、输出路径

```
[hadoop@node01 Desktop]$ hadoop jar com.kaikeba.hadoop-1.0-SNAPSHOT.jar com.kaikeba.hadoop.wordcount.WordCountMain /README.txt /wordcount02
```

- 确认结果

```
[hadoop@node01 ~]$ hadoop fs -ls /wordcount01
```



```
Terminal: Local +  
[hadoop@node01 Hadoop]$ hadoop fs -ls /wordcount01  
19/08/22 16:20:17 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin Java classes instead.  
Found 2 items  
★ -rw-r--r-- 3 hadoop supergroup 0 2019-08-22 16:18 /wordcount01/_SUCCESS  
★ -rw-r--r-- 3 hadoop supergroup 1413 2019-08-22 16:18 /wordcount01/part-r-00000  
[hadoop@node01 Hadoop]$
```

2.7 总结

- MR分为两个阶段：map阶段、reduce阶段
- MR输入的文件有几个block，就会生成几个map任务
- MR的reduce任务的个数，由程序中编程指定：job.setNumReduceTasks(4)
- map任务
 - map任务中map()一次读取block的一行数据，以kv对的形式输入map()
 - map()的输出作为reduce()的输入
- reduce任务
 - reduce任务通过网络将各执行完成的map任务输出结果中，属于自己的数据取过来
 - key相同的键值对作为一组，调用一次reduce()
 - reduce任务生成一个结果文件
 - 文件写入HDFS

3. WEB UI查看结果（5分钟）

3.1 Yarn

node01是resourcemanager所在节点主机名，根据自己的实际情况修改主机名

浏览器访问url地址：<http://node01:8088>

The screenshot shows the Hadoop 'All Applications' interface. On the left, there's a sidebar with 'Cluster Metrics' (Apps Submitted: 3, Apps Pending: 0, Apps Running: 0, Apps Completed: 3), 'Cluster Nodes Metrics' (Active Nodes: 3, Decommissioning Nodes: 0, Decommissioned Nodes: 0, Lost Nodes: 0), and a 'User Metrics for dr.who' section. The main area displays a table of running applications:

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores
application_1570781031851_0003	hadoop	distcp	MAPREDUCE	root.hadoop	Fri Oct 11 23:03:06 +0800 2019	Fri Oct 11 23:03:18 +0800 2019	FINISHED	SUCCEEDED	N/A	N/A
application_1570781031851_0002	hadoop	hadoop-archives-2.6.0-cdh5.14.2.jar	MAPREDUCE	root.hadoop	Fri Oct 11 22:59:36 +0800 2019	Fri Oct 11 22:59:43 +0800 2019	FINISHED	SUCCEEDED	N/A	N/A

3.2 HDFS结果

浏览器输入URL: <http://node01:50070>

①点击下拉框；②浏览文件系统；③输入根目录，查看hdfs根路径中的内容

The screenshot shows the HDFS browser interface. The top navigation bar includes 'Hadoop', 'Overview', 'Datanodes', 'Snapshot', 'Startup Progress', 'Utilities' (with a red notification badge), and 'Browse the file system' (with a red notification badge). The main area is titled 'Browse Directory' and shows a table of files under the root directory '/':

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	hadoop	supergroup	1.33 KB	Thu Aug 22 15:17:16 +0800 2019	3	128 MB	README.txt
drwx-----	hadoop	supergroup	0 B	Thu Aug 22 16:17:38 +0800 2019	0	0 B	tmp
drwxr-xr-x	hadoop	supergroup	0 B	Thu Aug 22 15:19:39 +0800 2019	0	0 B	wordcount
drwxr-xr-x	hadoop	supergroup	0 B	Thu Aug 22 16:18:03 +0800 2019	0	0 B	wordcount01
drwxr-xr-x	hadoop	supergroup	0 B	Thu Aug 22 16:34:40 +0800 2019	0	0 B	wordcount02

At the bottom, it says 'Hadoop, 2017.'

4. MapReduce编程：数据清洗（10分钟）

mapreduce在企业中，可以用于对海量数据的数据清洗；当然，随着新一代大数据框架的出现，也可以使用spark、flink等框架，做数据清洗

4.1 需求

- 现有一批日志文件，日志来源于用户使用搜狗搜索引擎搜索新闻，并点击查看搜索结果过程；
- 但是，日志中有一些记录损坏，现需要使用MapReduce来将这些**损坏记录**（如记录中少字段、多字段）从日志文件中删除，此过程就是传说中的**数据清洗**。
- 并且在清洗时，要**统计损坏的记录数**。

4.2 数据结构

- 日志格式：每行记录有6个字段；分别表示时间datetime、用户ID userid、新闻搜索关键字searchkw、当前记录在返回列表中的序号reorder、用户点击链接的顺序cliorder、点击的URL连接cliurl

1	20111230000005	57375476989ee12893c0c3811607bcf	奇艺高清	1	1	http://www.qiyi.com/
2	20111230000005	66c5b7774e310a2278249b26bc83a	凡人修仙传	3	1	http://www.booksky.org/BookDetail.aspx?BookID=1050804&Level=1
3	20111230000007	b797920521c78de70ac38e3713f524b50	本本联盟	1	1	http://www.bblianmeng.com/
4	20111230000008	6961d0c97fe93701fc90d861d096cd9	华南师范大学图书馆	1	1	http://lib.scnu.edu.cn/
5	20111230000008	f2f5a21c764aebed1e8afcc2871e086f	在线代理	2	1	http://proxie.cn/
6	20111230000009	96994a0480e71edcaeef67b20d816b7	伟大导演	1	1	http://movie.douban.com/review/1128960/
7	20111230000009	698956eb07815439fe5f46e9a4503997	youku	1	1	http://www.youku.com/
8	20111230000009	599cd26984f72ee68b2b66bebeffcaed	安徽合肥365房产网	1	1	http://hf.house365.com/
9	20111230000010	f577230df7b6c532837cd16ab731f874	哈萨克网址大全	1	1	http://www.kz321.com/
0	20111230000010	f59f88780dd0659f5fc0acc7ca4949f2	IQ数码	1	1	http://www.iqshuma.com/
1	20111230000010	f4ba3f337efb1c469fcdb34feff9fb	推荐待机时间长的手机	1	1	http://mobile.zol.com.cn/148/1487938.html
2	20111230000010	3d1acc7235374d531de1ca885df5e711	满江红	1	1	http://baike.baidu.com/view/6500.htm
3	20111230000010	dbce4101683913365648eba6a85b6273	光标下载	1	1	http://zhidao.baidu.com/question/38626533
4	20111230000011	58e7d0caec23bbcb4da7bcc4d37f008	张国立的电视剧	2	1	http://tv.sogou.com/vertical/2xc3k6wbuk24inphzli35zy.html?p=40230600

关于reorder、cliorder说明：

The screenshot shows a search results page for '凡人修仙传'. The results are ordered by reorder (1, 2, 3) and then by cliorder (3, 1, 2). The first result (reorder 1, cliorder 3) is a game advertisement for '凡人修仙传' on cs.sogou.com. The second result (reorder 2, cliorder 1) is another game advertisement for '凡人飞仙H5游戏' on h5.leyoww.cn. The third result (reorder 3, cliorder 2) is a free online reading link for the full text of '凡人修仙传' on 147xs.com.

4.3 逻辑分析

- MapReduce程序一般分为map阶段，将任务分而治之；
- reduce阶段，将map阶段的结果进行聚合；
- 而有些mapreduce应用不需要数据聚合的操作，也就是说**不需要reduce阶段**。即编程时，不需要编写自定义的reducer类；在main()中调用job.setNumReduceTasks(0)设置
- 而本例的数据清洗就是属于此种情况
- 统计损坏的记录数，可使用**自定义计数器**的方式进行

```

Run: WordCountMain
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -JobClient.java(1492) - map 100% reduce 100%
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -JobClient.java(1547) -Job complete: job_local1181430430_0001
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(566) Counters: 17 系统内置计数器
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(568) - File System Counters
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - FILE: Number of bytes read=4865
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - FILE: Number of bytes written=343086
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - FILE: Number of read operations=0
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - FILE: Number of large read operations=0
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - FILE: Number of write operations=0
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(568) - Map-Reduce Framework
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Map input records=31
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Map output records=197
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Map output bytes=2145
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Input split bytes=89
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Combine input records=197
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Combine output records=132
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Reduce input groups=132
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Reduce shuffle bytes=0
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Reduce input records=132
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Reduce output records=132
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Spilled Records=264
2019-10-14 12:33:36 [INFO] - org.apache.hadoop.mapred.JobClient -Counters.java(570) - Total committed heap usage (bytes)=514850816

```

- map方法的逻辑：取得每一行数据，与每条记录的固定格式比对，是否符合？
 - 若符合，则是完好的记录；
 - 否则是损坏的记录。并对自定义计数器累加

4.4 MR代码

若要集群运行，需先将sogou.50w.utf8上传到HDFS根目录

```
[hadoop@node01 soft]$ pwd
/kkb/soft
[hadoop@node01 soft]$ hadoop fs -put sogou.50w.utf8 /
```

运行等操作，与上边类似；不再赘述

4.4.1 Mapper类

具体逻辑，可详见代码注释

注意：实际工作中，写良好的代码注释也是基本的职业素养

```

package com.kaikeba.hadoop.dataclean;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

/**
 *
 * 现对sogou日志数据，做数据清洗；将不符合格式要求的数据删除
 * 每行记录有6个字段：
 * 分别表示时间datetime、用户ID userid、新闻搜索关键字searchkw、当前记录在返回列表中的序
 * 号reorder、用户点击链接的顺序cliorder、点击的URL连接cliurl
 */

```

```
* 日志样本:  
* 20111230111308 0bf5778fc7ba35e657ee88b25984c6e9          nba直播 4      1  
* http://www.hoopchina.com/tv  
*  
*/  
public class DataClean {  
    /**  
     *  
     * 基本上大部分MR程序的main方法逻辑，大同小异；将其他MR程序的main方法代码拷贝过来，稍做修改即可  
     * 实际开发中，也会有很多的复制、粘贴、修改  
     *  
     * 注意：若要IDEA中，本地运行MR程序，需要将resources/mapred-site.xml中的  
     mapreduce.framework.name属性值，设置成local  
     * @param args  
     */  
    public static void main(String[] args) throws IOException,  
ClassNotFoundException, InterruptedException {  
  
        //判断一下，输入参数是否是两个，分别表示输入路径、输出路径  
        if (args.length != 2 || args == null) {  
            System.out.println("please input Path!");  
            System.exit(0);  
        }  
  
        Configuration configuration = new Configuration();  
  
        //调用getInstance方法，生成job实例  
        Job job = Job.getInstance(configuration,  
DataClean.class.getSimpleName());  
  
        //设置jar包，参数是包含main方法的类  
        job.setJarByClass(DataClean.class);  
  
        //设置输入/输出路径  
        FileInputFormat.setInputPaths(job, new Path(args[0]));  
        FileOutputFormat.setOutputPath(job, new Path(args[1]));  
  
        //设置处理Map阶段的自定义的类  
        job.setMapperClass(DatacleanMapper.class);  
  
        //注意：此处设置的map输出的key/value类型，一定要与自定义map类输出的kv对类型一致；否则程序运行报错  
        job.setMapOutputKeyClass(Text.class);  
        job.setMapOutputValueClass(NullWritable.class);  
  
        //注意：因为不需要reduce聚合阶段，所以，需要显示设置reduce task个数是0  
        job.setNumReduceTasks(0);  
  
        // 提交作业  
        job.waitForCompletion(true);  
    }  
  
    /**  
     *  
     * 自定义mapper类  
     * 注意：若自定义的mapper类，与main方法在同一个类中，需要将自定义mapper类，声明成  
     static的  
    */
```

```

*/
public static class DataCleanMapper extends Mapper<LongWritable, Text, Text,
NullWritable> {
    //为了提高程序的效率，避免创建大量短周期的对象，触发频繁GC；此处生成一个对象，共用
    NullWritable nullvalue = NullWritable.get();

    @Override
    protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
        //自定义计数器，用于记录残缺记录数
        Counter counter = context.getCounter("DataCleaning",
        "damagedRecord");

        //获得当前行数据
        //样例数据：20111230111645 169796ae819ae8b32668662bb99b6c2d 塘
承高速公路规划线路图 1 1
http://auto.ifeng.com/roll/20111212/729164.shtml
        String line = value.toString();

        //将行数据按照记录中，字段分隔符切分
        String[] fields = line.split("\t");

        //判断字段数组长度，是否为6
        if(fields.length != 6) {
            //若不是，则不输出，并递增自定义计数器
            counter.increment(1L);
        } else {
            //若是6，则原样输出
            context.write(value, nullvalue);
        }
    }
}
}

```

4.4.2 运行结果

仅以本地运行演示

- 运行参数

"C:\01 HK\高级03班\MapReduce\20191014-MR-第一次\sogou.50w.utf8" C:\test\dataclean



- ①reduce 0%， job就已经successfully，表示此MR程序没有reduce阶段

- ②DataCleaning是自定义计数器组名；damagedRecord是自定义的计数器；值为6，表示有6条损坏记录

```

Run: DataClean
2019-08-23 11:07:12 [ INFO] - org.apache.hadoop.mapred.LocalJobRunner - LocalJobRunner.java(591) -map
2019-08-23 11:07:12 [ INFO] - org.apache.hadoop.mapred.Task -Task.java(1158) -Task 'attempt_local1047857088_0001_m_000000_0' done.
2019-08-23 11:07:12 [ INFO] - org.apache.hadoop.mapred.LocalJobRunner -LocalJobRunner.java(249) -Finishing task: attempt_local1047857088_0001_m_000000_0
2019-08-23 11:07:12 [ INFO] - org.apache.hadoop.mapred.LocalJobRunner -LocalJobRunner.java(456) -map task executor complete.
2019-08-23 11:07:12 [ INFO] - org.apache.hadoop.mapreduce.Job -Job.java(1367) [map 100% reduce 0%]
2019-08-23 11:07:12 [ INFO] - org.apache.hadoop.mapreduce.Job -Job.java(1378) -Job job_local1047857088_0001 [completed successfully]
2019-08-23 11:07:12 [ INFO] - org.apache.hadoop.mapreduce.Job -Job.java(1385) -Counters: 21

File System Counters
FILE: Number of bytes read=203
FILE: Number of bytes written=301761
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=57288205
HDFS: Number of bytes written=57287664
HDFS: Number of read operations=1
HDFS: Number of large read operations=0
HDFS: Number of write operations=3

Map-Reduce Framework
Map input records=500000
Map output records=499994
Input split bytes=98
Spilled Records=0
Failed Shuffles=0
Merged Map outputs=0
GC time elapsed (ms)=32
Total committed heap usage (bytes)=271581184

2 DataCleaning 自定义计数器 (组名、计数器名)
damagedRecord=6

```

- 图中part-m-00000中的m表示，此文件是由map任务生成

此电脑 > Windows (C:) > test > dataclean				
	名称	修改日期	类型	大小
	._SUCCESS.crc	2019/10/14 12:39	CRC 文件	1 KB
	.part-m-00000.crc	2019/10/14 12:39	CRC 文件	257 KB
计算	.part-m-00001.crc	2019/10/14 12:39	CRC 文件	182 KB
	._SUCCESS	2019/10/14 12:39	文件	0 KB
	part-m-00000	2019/10/14 12:39	文件	32,768 KB
	part-m-00001	2019/10/14 12:39	文件	23,178 KB

4.5 总结

- MR可用于数据清洗；另外，也可以使用Spark、Flink等框架做数据清洗
- 可使用自定义计数器记录符合特定条件的记录数，用于统计

5. MapReduce编程：用户搜索次数 (10分钟)

5.1 需求

- 使用MR编程，统计sogou日志数据中，每个用户搜索的次数；结果写入HDFS

5.2 数据结构

- 数据来自“MapReduce编程：数据清洗”中的输出结果
- 仍然是sogou日志数据；不再赘述

1 20111230000005 -57375476999ea12893c0c3811607bcf	奇艺高清	1	1	http://www.qiyi.com/
2 20111230000005 665bb7774e31d0a22270249b26bc83a	凡人修仙传	3	1	http://www.booksky.org/BookDetail.aspx?BookID=1050804&Level=1
3 20111230000007 b97920521c78de70ac38e3713f524b50	本木联盟	1	1	http://www.bblianmeng.com/
4 20111230000008 6961d0c97fe93701fc9c0d861d096cd9	华南师范大学图书馆	1	1	http://lib.scnu.edu.cn/
5 20111230000008 f2f5a21c764aeabd1e8afcc2871e086f	在线代理	2	1	http://proxyie.cm/
6 20111230000009 96994a0480e71edcaef67b20d816b7	伟大导演	1	1	http://movie.douban.com/review/1128960/
7 20111230000009 698956eb07815439fe5f46e9a2053997	youku	1	1	http://www.youku.com/
8 20111230000009 599cd26984f72ee6b2b6bebefccfaed	安徽合肥365房产网	1	1	http://hf.house365.com/
9 20111230000010 f577230df7b6c532837cd16ab731f874	哈萨克网址大全	1	1	http://www.kz321.com/
0 20111230000010 -285f88780dd659ffcaacc7cc4949f2	IQ数码	1	1	http://www.iqshuma.com/
1 20111230000010 f4ba3f337efb1cc469fcdb34feff9fb	推荐待机时间长的手机	1	1	http://mobile.zol.com.cn/148/1487938.html
2 20111230000010 -3d1acc7235374d531de1ca085df5e711	满江红	1	1	http://baike.baidu.com/view/6500.htm
3 20111230000010 dbce4010683913365648ebaba85b6273	光标下载	1	1	http://zhidao.baidu.com/question/38626533
4 20111230000011 58e740caec23bcba4da7bbcc437f008	张国立的电视剧	2	1	http://tv.sogou.com/vertical/2xc3t6wbuk24inphz135zy.html?p=40230600

5.3 逻辑分析

- 还记得之前提到的MR中key的作用吗？
- MR编程时，若要针对某个值对数据进行分组、聚合时，如当前的词频统计例子，需要将每个单词作为reduce输入的key，从而按照单词分组，进而求出每组即每个单词的总次数
- 那么此例也是类似的。

- 统计每个用户的搜索次数，将userid放到reduce输入的key的位置；
- 对userid进行分组
- 进而统计每个用户的搜索次数

5.4 MR代码

给MR程序在IDEA中设置参数，运行等操作，与上边类似；不再赘述

此处MR程序的输入文件是“MapReduce编程：数据清洗”中的输出结果文件

```
package com.kaikeba.hadoop.searchcount;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;

/**
 *
 * 本MR示例，用于统计每个用户搜索并查看URL链接的次数
 */
public class UserSearchCount {

    /**
     *
     * @param args C:\test\datacleanresult c:\test\usersearch
     * @throws IOException
     * @throws ClassNotFoundException
     * @throws InterruptedException
     */
    public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {

        //判断一下，输入参数是否是两个，分别表示输入路径、输出路径
        if (args.length != 2 || args == null) {
            System.out.println("please input Path!");
            System.exit(0);
        }

        Configuration configuration = new Configuration();

        //configuration.set("mapreduce.job.jar","/home/hadoop/IdeaProjects/Hadoop/targe
        t/com.kaikeba.hadoop-1.0-SNAPSHOT.jar");

        //调用getInstance方法，生成job实例
        Job job = Job.getInstance(configuration,
UserSearchCount.class.getSimpleName());

        //设置jar包，参数是包含main方法的类
    }
}
```

```

        job.setJarByClass(UserSearchCount.class);

        //通过job设置输入/输出格式
        //MR的默认输入格式是TextInputFormat，输出格式是TextOutputFormat；所以下两行可以
注释掉
//        job.setInputFormatClass(TextInputFormat.class);
//        job.setOutputFormatClass(TextOutputFormat.class);

        //设置输入/输出路径
FileInputFormat.setInputPaths(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

        //设置处理Map阶段的自定义的类
job.setMapperClass(SearchCountMapper.class);
//设置map combine类，减少网路传出量
//job.setCombinerClass(WordCountReduce.class);
//设置处理Reduce阶段的自定义的类
job.setReducerClass(SearchCountReducer.class);

        //如果map、reduce的输出的kv对类型一致，直接设置reduce的输出的kv对就行；如果不一样，需要分别设置map，reduce的输出的kv类型
        //注意：此处设置的map输出的key/value类型，一定要与自定义map类输出的kv对类型一致；否则程序运行报错
//        job.setMapOutputKeyClass(Text.class);
//        job.setMapOutputValueClass(IntWritable.class);

        //设置reduce task最终输出key/value的类型
        //注意：此处设置的reduce输出的key/value类型，一定要与自定义reduce类输出的kv对类型一致；否则程序运行报错
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);

        //提交作业
job.waitForCompletion(true);
}

public static class SearchCountMapper extends Mapper<LongWritable, Text,
Text, IntWritable> {

        //定义公用的对象，减少GC压力
Text userIdKOut = new Text();
IntWritable vout = new IntWritable(1);

@Override
protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {

        //获得当前行的数据
        //样例数据：20111230111645 169796ae819ae8b32668662bb99b6c2d 塘
承高速公路规划线路图 1 1
http://auto.ifeng.com/roll/20111212/729164.shtml
String line = value.toString();

        //切分，获得各字段组成的数组
String[] fields = line.split("\t");

        //因为要统计每个user搜索并查看URL的次数，所以将userid放到输出key的位置
        //注意：MR编程中，根据业务需求设计key是很重要的能力
}
}

```

```

        String userid = fields[1];

        //设置输出的key的值
        userIdKOut.set(userid);
        //输出结果
        context.write(userIdKOut, vOut);
    }
}

public static class SearchCountReducer extends Reducer<Text, IntWritable,
Text, IntWritable> {

    //定义公用的对象，减少GC压力
    IntWritable totalNumVOut = new IntWritable();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values, Context
context) throws IOException, InterruptedException {
        int sum = 0;

        for(IntWritable value: values) {
            sum += value.get();
        }

        //设置当前user搜索并查看总次数
        totalNumVOut.set(sum);
        context.write(key, totalNumVOut);
    }
}
}

```

5.4.1 结果

- 运行参数

```
C:\test\datacleanresult c:\test\usersearch
```

- 运行结果

此电脑 > Windows (C:) > test > usersearch					
	名称	修改日期	类型	大小	
	._SUCCESS.crc	2019/10/14 12:50	CRC 文件	1 KB	
	.part-r-00000.crc	2019/10/14 12:50	CRC 文件	40 KB	
	_SUCCESS	2019/10/14 12:50	文件	0 KB	
	part-r-00000	2019/10/14 12:50	文件	5,100 KB	

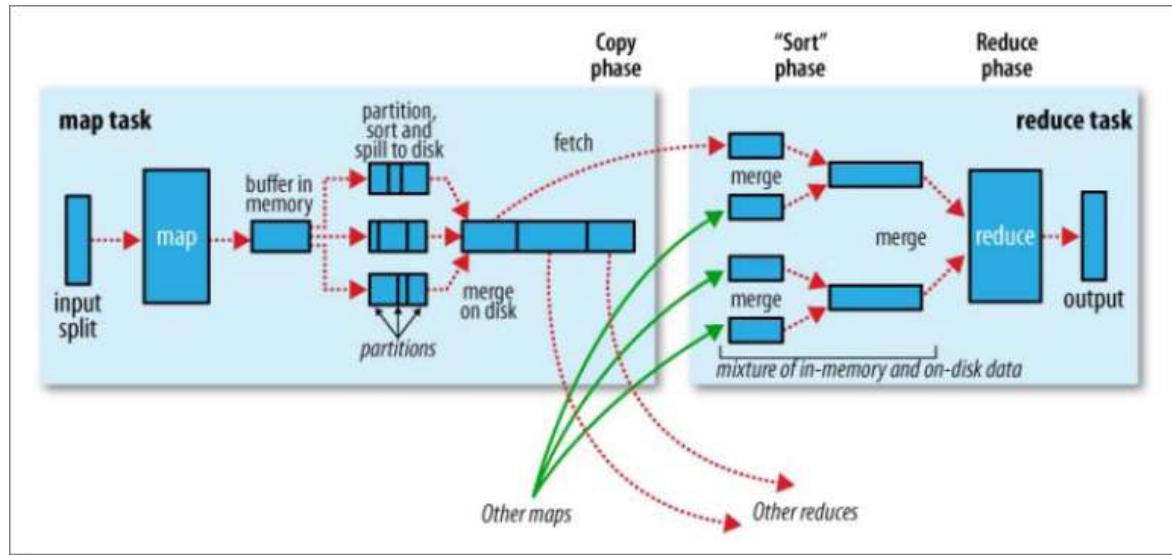
5.5 总结

- 结合本例子的需求，设计MR程序；因为要统计每个用户的搜索次数，所以最终userid作为reduce的输出的key
- MR编程能够根据业务需求设计合适的key是一个很重要的能力；而这是需要建立在自己地MR框架原理有清晰认识的基础之上的

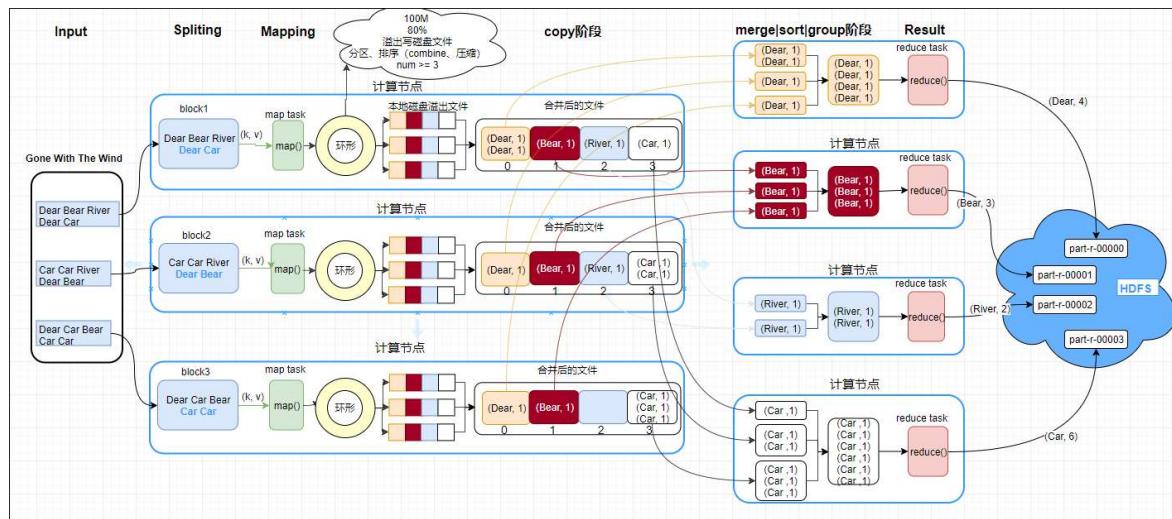
6. Shuffle (重点 20分钟)

- shuffle主要指的是map端的输出作为reduce端输入的过程

6.1 shuffle简图



6.2 shuffle细节图



- 分区用到了分区器，默认分区器是HashPartitioner

源码：

```

public class HashPartitioner<K2, V2> implements Partitioner<K2, V2> {

    public void configure(JobConf job) {}

    /** Use {@link object#hashCode()} to partition. */
    public int getPartition(K2 key, V2 value,
                           int numReduceTasks) {
        return (key.hashCode() & Integer.MAX_VALUE) % numReduceTasks;
    }
}

```

6.3 map端

- 每个map任务都有一个对应的环形内存缓冲区；输出是kv对，先写入到环形缓冲区（默认大小100M），当内容占据80%缓冲区空间后，由一个后台线程将缓冲区中的数据溢出写到一个磁盘文件
- 在溢出写的过程中，map任务可以继续向环形缓冲区写入数据；但是若写入速度大于溢出写的速度，最终造成100m占满后，map任务会暂停向环形缓冲区中写数据的过程；只执行溢出写的过程；直到环形缓冲区的数据全部溢出写到磁盘，才恢复向缓冲区写入
- 后台线程溢写磁盘过程，有以下几个步骤：
 - 先对每个溢写的kv对做分区；分区的个数由MR程序的reduce任务数决定；默认使用HashPartitioner计算当前kv对属于哪个分区；计算公式：`(key.hashCode() & Integer.MAX_VALUE) % numReduceTasks`
 - 每个分区中，根据kv对的key做内存中排序；
 - 若设置了map端本地聚合combiner，则对每个分区中，排好序的数据做combine操作；
 - 若设置了对map输出压缩的功能，会对溢写数据压缩
- 随着不断的向环形缓冲区中写入数据，会多次触发溢写（每当环形缓冲区写满100m），本地磁盘最终会生成多个溢出文件
- 合并溢写文件：在map task完成之前，所有溢出文件会被合并成一个大的溢出文件；且是已分区、已排序的输出文件
- 小细节：
 - 在合并溢写文件时，如果至少有3个溢写文件，并且设置了map端combine的话，会在合并的过程中触发combine操作；
 - 但是若只有2个或1个溢写文件，则不触发combine操作（因为combine操作，本质上是一个reduce，需要启动JVM虚拟机，有一定的开销）

6.4 reduce端

- reduce task会在每个map task运行完成后，通过HTTP获得map task输出中，属于自己的分区数据（许多kv对）
- 如果map输出数据比较小，先保存在reduce的jvm内存中，否则直接写入reduce磁盘
- 一旦内存缓冲区达到阈值（默认0.66）或map输出数的阈值（默认1000），则触发归并merge，结果写到本地磁盘
- 若MR编程指定了combine，在归并过程中会执行combine操作
- 随着溢出写的文件的增多，后台线程会将它们合并大的、排好序的文件
- reduce task将所有map task复制完后，将合并磁盘上所有的溢出文件
- 默认一次合并10个
- 最后一批合并，部分数据来自内存，部分来自磁盘上的文件
- 进入“归并、排序、分组阶段”
- 每组数据调用一次reduce方法
- 参考文件《**reduce端merge 排序 分组.txt**》

6.5 总结

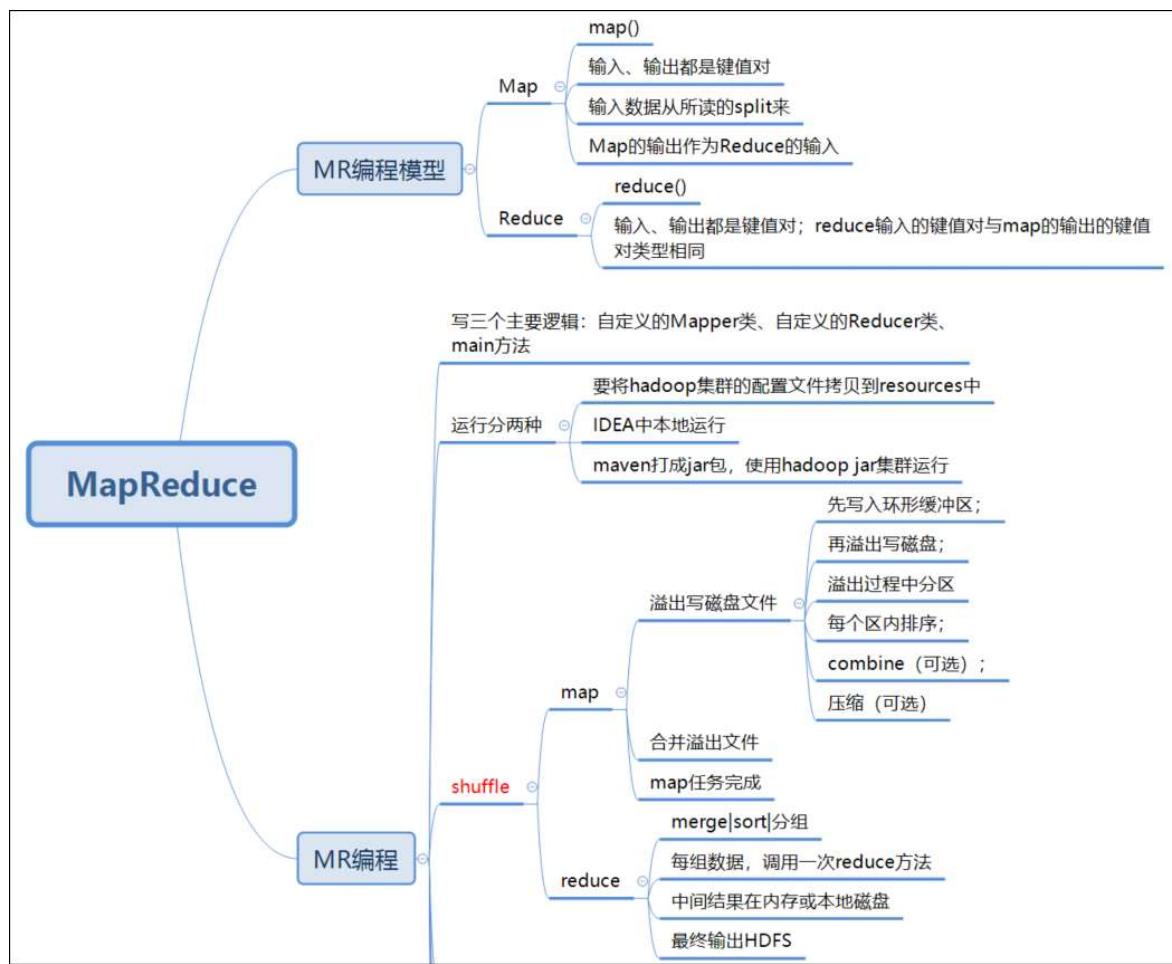
- map端
 - map()输出结果先写入环形缓冲区
 - 缓冲区100M；写满80M后，开始溢出写磁盘文件
 - 此过程中，会进行分区、排序、combine（可选）、压缩（可选）
 - map任务完成前，会将多个小的溢出文件，合并成一个大的溢出文件（已分区、排序）
- reduce端
 - 拷贝阶段：reduce任务通过http将map任务属于自己的分区数据拉取过来
 - 开始merge及溢出写磁盘文件

- 所有map任务的分区全部拷贝过来后，进行阶段合并、排序、分组阶段
- 每组数据调用一次reduce()
- 结果写入HDFS

五、拓展点、未来计划、行业趋势（5分钟）

- 扩展阅读：
 - 《Hadoop权威指南 第4版》7.3小节 Shuffle和排序

六、总结（5分钟）



七、作业

八、互动问答

九、题库 - 本堂课知识点

- 描述MR的shuffle全流程（面试）

2. 搭建MAVEN工程，统计词频，并提交集群运行，查看结果
3. 利用搜狗数据，找出所有独立的uid并写入HDFS
4. 利用搜狗数据，找出所有独立的uid出现次数，并写入HDFS，并要求使用Map端的Combine操作
5. 谈谈什么是数据倾斜，什么情况会造成数据倾斜？（面试）
6. 对MR数据倾斜，如何解决？（面试）