实验2 文档倒排索引

一(1)、带词频索引的文档倒排算法

map类

首先对输入键值对的文件来源作处理，如果是来自停用词文件则直接返回；之后将输入`value`对应的一行进行分词，对于每个词语，将词语作为`key`，文档名作为`value`并输出（类型均为`Text`）

```
public static class InvertedIndexMapper extends Mapper<Object, Text, Text, Text> {
    @Override
    public void map(Object key, Text value, Context context) throws IOException, InterruptedExcept
        // default RecordReader: LineRecordReader; key: line offset; value: line string
        FileSplit fileSplit = (FileSplit) context.getInputSplit();
        String fileName = fileSplit.getPath().getName();
        if (fileName.equals("cn_stopwords.txt"))  return;

        Text word = new Text();
        StringTokenizer itr = new StringTokenizer(value.toString());
        for (; itr.hasMoreTokens(); ) {
            word.set(itr.nextToken());
            context.write(word, new Text(fileName));  // 词语作为key，文档名作为value
        }
    }
}
```

reduce类

建立一个`HashMap`用于存放所有出现过该词语的文档名和次数，在此过程中统计词语在所有文档中出现的总次数，以及包含该词语的文档数，从而计算出平均出现次数；然后再将`HashMap`遍历一遍，将每个文档对应的词频加在后面，从而输出符合要求的结果（`key`和`value`的类型均为`Text`）

```java
public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, Interrupt
    Iterator<Text> it = values.iterator();
    HashMap<String, Integer> countMap = new HashMap<>();  // {文档名，词语出现次数}
    int sum = 0;  // 词语在所有文档中出现的总次数
    int num = 0;  // 包含该词语的文档数
    while (it.hasNext()) {
        sum++;
        String docName = it.next().toString();
        if (countMap.containsKey(docName)) {
            countMap.put(docName, countMap.get(docName) + 1);  // 词语在该文档中出现的次数加1
        }
        else {
            countMap.put(docName,1);  // 词语在该文档中第一次出现
            num++;  // 包含该词语的文档数加1
        }
    }

    StringBuilder all = new StringBuilder();
    float avg = (float) sum / num; // 平均出现次数
    String formatted = String.format("%.2f", avg); // 保留两位小数
    all.append(formatted).append(", ");

    for (Map.Entry<String, Integer> entry : countMap.entrySet()) {//遍历文档名和该文档内的词语出现次数
        String docName = entry.getKey();
        int count = entry.getValue();
        all.append(docName).append(":").append(count).append("; ");  // 每个文档的记录
    }

    // 删除最后一个多余的分号和空格
    if (!countMap.isEmpty()) {
        all.setLength(all.length() - 2);
    }

    String docName = "[" + key.toString() + "]";
    context.write(new Text(docName), new Text(all.toString()));
}
```

运行结果

文件 - /user/221220159stu/output-Exp2/...

Page 1 of 3050

```
[0]    3.00, 第三部-阿兹卡班的囚徒:2; 第四部-火焰杯:4
[1]    28.29, 第二部-密室:13; 第七部-死亡神器:21; 第六部-混血王子:28; 第一部-魔法石:23; 第三部-阿兹卡班的囚徒:13; 第四部-火焰杯:47; 第五部-凤凰社:53
[10]  7.00, 第七部-死亡神器:6; 第二部-密室:3; 第六部-混血王子:7; 第一部-魔法石:8; 第三部-阿兹卡班的囚徒:2; 第五部-凤凰社:16; 第四部-火焰杯:7
[1012]      1.00, 第六部-混血王子:1
[1056]      1.00, 第六部-混血王子:1
[11]  2.86, 第二部-密室:1; 第七部-死亡神器:4; 第六部-混血王子:2; 第一部-魔法石:3; 第三部-阿兹卡班的囚徒:1; 第五部-凤凰社:5; 第四部-火焰杯:4
[12]  6.00, 第七部-死亡神器:9; 第二部-密室:2; 第六部-混血王子:7; 第一部-魔法石:1; 第三部-阿兹卡班的囚徒:1; 第五部-凤凰社:21; 第四部-火焰杯:1
[125]       1.00, 第三部-阿兹卡班的囚徒:1
[13]  2.00, 第七部-死亡神器:6; 第二部-密室:1; 第六部-混血王子:1;
```

取消 下载

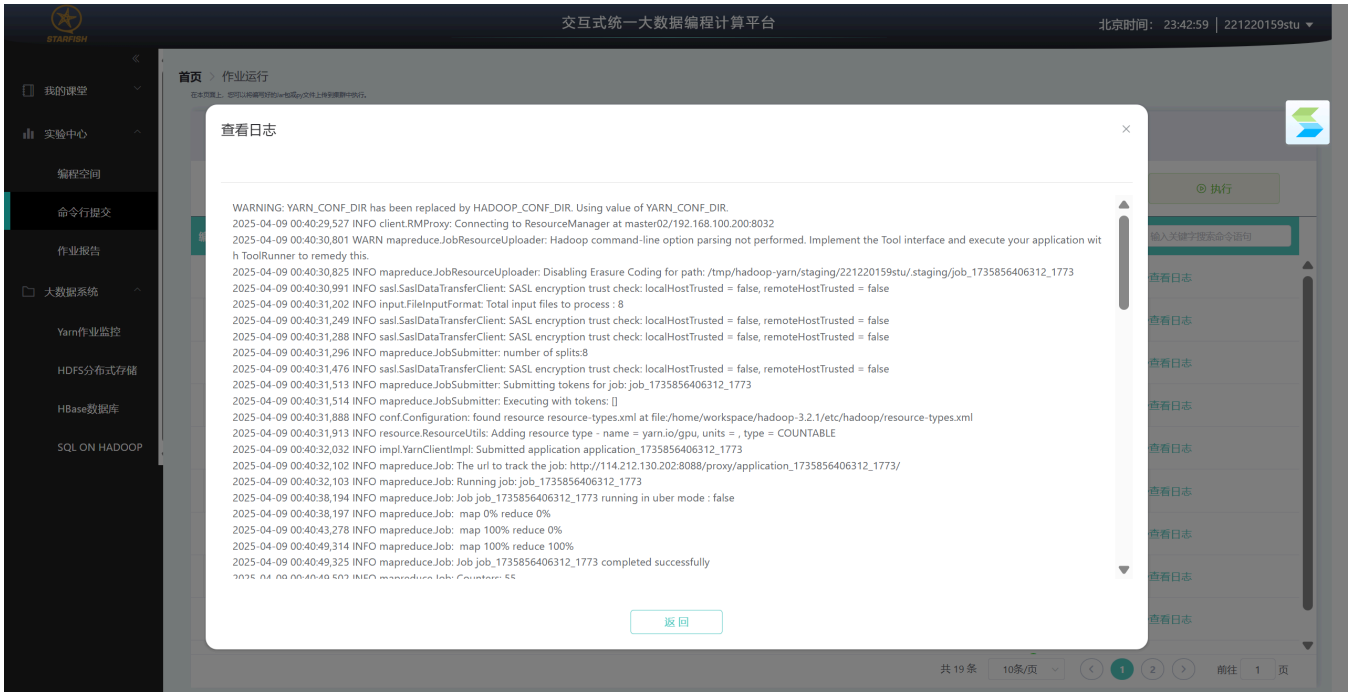
集群执行结果

输入命令`yarn jar BD2.jar InvertedIndexer /user/root/Exp2 output-Exp2/1-1`

输出结果存储在`/user/221220159stu/output-Exp2/1-1`

Show 20 entries

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB
application_1735856406312_1773	221220159stu	InvertedIndexer	MAPREDUCE	2025class03	0	Wed Apr 9 12:40:31 +0800 2025	Wed Apr 9 12:40:31 +0800 2025	Wed Apr 9 12:40:48 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A



一(2)、对词语的平均出现次数进行全局排序

map类

输入的一行数据里，格式是`[词语] 平均出现次数，所有文档词频`
现在使用正则表达将这三部分分开，把平均出现次数赋给`key`，剩余两部分合并赋给`value`，从而在`map`之后可以根据平均出现次数进行排序（`key`的类型为`FloatWritable`，`value`的类型为`Text`）

```
public static class SortMapper extends Mapper<Object, Text, FloatWritable, Text>
{
    @Override
    public void map(Object key, Text value, Context context) throws IOException, InterruptedExcept
    {
        // parts[0] is the term, parts[1] is the average, parts[2] are the files
        String[] parts = value.toString().split(",|\\s+", 3);
        FloatWritable average = new FloatWritable(Float.parseFloat(parts[1]));
        Text term = new Text(parts[0] + "," + parts[2]);
        context.write(average, term);
    }
}
```

reduce类

把三部分还原成一开始的格式，即`key`是[词语]，`value`是平均出现次数+所有文档词频，并输出（`key`和`value`的类型均为`Text`）

```
public static class SortReducer extends Reducer<FloatWritable, Text, Text, Text>
{
    @Override
    public void reduce(FloatWritable key, Iterable<Text> values, Context context) throws IOExcepti
    {
        for (Text value : values)
        {
            String[] parts = value.toString().split(",", 2);
            String new_key = parts[0];
            String new_value = key + ", " + parts[1];
            context.write(new Text(new_key), new Text(new_value));
        }
    }
}
```

DescendingFloatComparator类

`mapreduce`给`key`默认的排序是从小到大，但直观上我们更希望看到词频从大到小排序，因此可以自定义一个`float`的比较运算符，将已有的比较结果取反即可

```
public static class DescendingFloatComparator extends FloatWritable.Comparator
{
    @Override
    public int compare(byte[] b1, int s1, int l1, byte[] b2, int s2, int l2)
    {
        return -super.compare(b1, s1, l1, b2, s2, l2);
    }
    @Override
    public int compare(Object a, Object b)
    {
        return -super.compare((WritableComparable)a, (WritableComparable)b);
    }
}
```

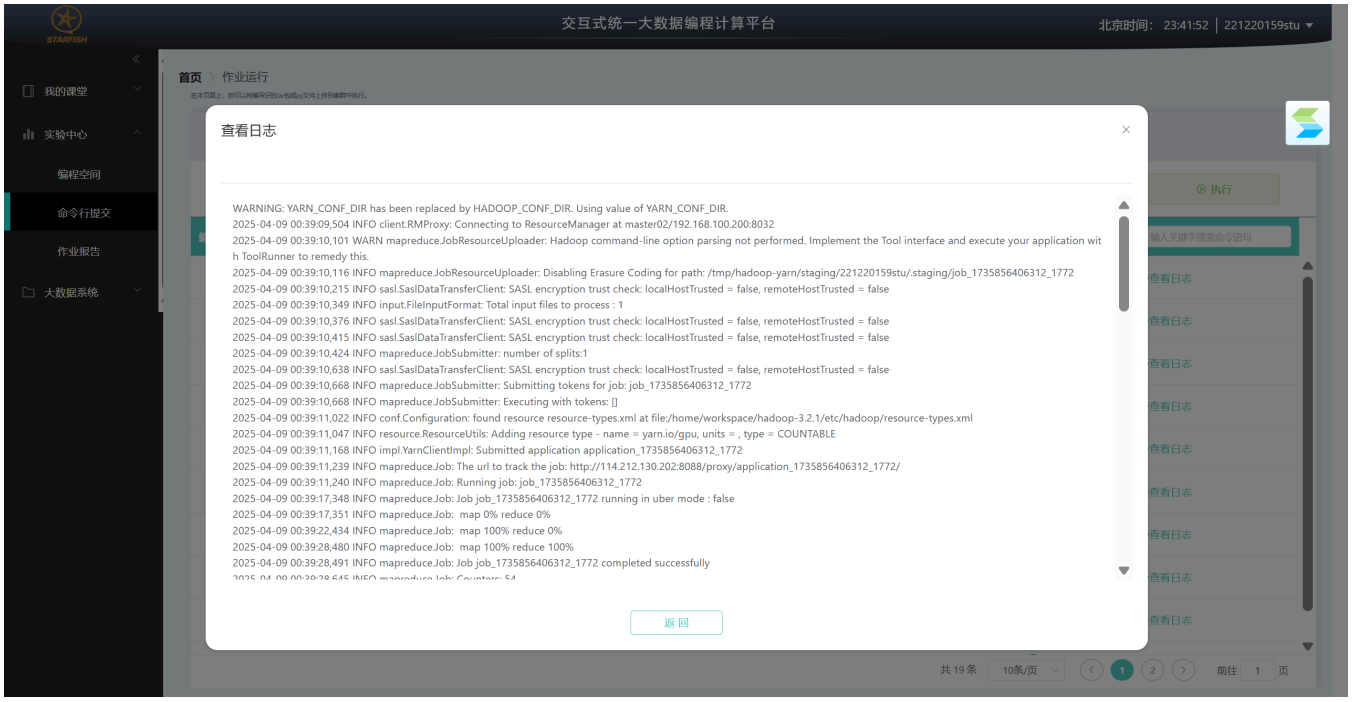
运行结果

集群执行结果

输入命令 `yarn jar BD2.jar Sort /user/221220159stu/output-Exp2/1-1 output-Exp2/1-2`

输出结果存储在 `/user/221220159stu/output-Exp2/1-2`

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU VCores	Allocated Memory MB
application_1735856406312_1772	221220159stu	Sort	MAPREDUCE	2025class03	0	Wed Apr 9 12:39:11 +0800 2025	Wed Apr 9 12:39:11 +0800 2025	Wed Apr 9 12:39:26 +0800 2025	FINISHED	SUCCEEDED	N/A	N/A	N/A



二、计算TF-IDF

在`main`函数中增加一个变量存储语料库里的文档数目

```
Path inputPath = new Path(args[0]);
FileSystem fs = inputPath.getFileSystem(conf);
FileStatus[] fileStatuses = fs.listStatus(inputPath);
int fileCount = fileStatuses.length - 1; //去除停用词文档
conf.setInt("fileCount", fileCount);
fs.close();
```

map类

和`InvertedIndexer`基本一致，将输入`value`对应的一行进行分词，对于每个词语，将词语作为`key`，文档名作为`value`并输出（类型均为`Text`）

```
public static class TF_IDFMapper extends Mapper<Object, Text, Text, Text> {
    @Override
    public void map(Object key, Text value, Context context) throws IOException, InterruptedExcept
        // default RecordReader: LineRecordReader; key: line offset; value: line string
        FileSplit fileSplit = (FileSplit) context.getInputSplit();
        String fileName = fileSplit.getPath().getName();
        if (fileName.equals("cn_stopwords.txt"))  return;
        Text word = new Text();
        StringTokenizer itr = new StringTokenizer(value.toString());
        for (; itr.hasMoreTokens(); ) {
            word.set(itr.nextToken());
            context.write(word, new Text(fileName));
        }
    }
}
```

reduce类

同样建立一个`HashMap`用于存放所有出现过该词语的文档名和次数，在此过程中统计包含该词的文档数；然后再将`HashMap`遍历一遍，为每个文档计算TF-IDF，并输出结果（`key`和`value`的类型均为`Text`）
在此过程中还出了bug，后来发现是`/`默认是整除，需要加上`(double)`才能得到非整数结果。

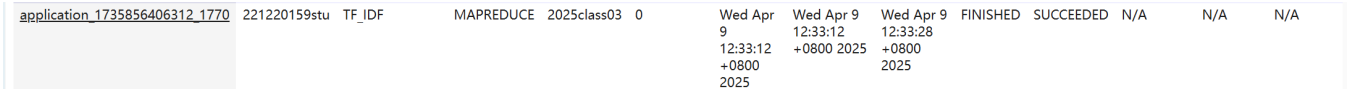
```java
public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, Interrupt
    Iterator<Text> it = values.iterator();
    HashMap<String, Integer> countMap = new HashMap<>();  // {文档名，词语出现次数}
    int num = 0;  // 包含该词语的文档数
    while (it.hasNext()) {
        String docName = it.next().toString();
        if (countMap.containsKey(docName)) {
            countMap.put(docName, countMap.get(docName) + 1);
        }
        else {
            countMap.put(docName,1);
            num++;
        }
    }

    Configuration conf = context.getConfiguration();
    int fileCount = conf.getInt("fileCount", 0);
    double IDF = Math.log((double)fileCount / (num + 1)) / Math.log(2);

    // 为每个文档计算TF-IDF
    for (Map.Entry<String, Integer> entry : countMap.entrySet()) {
        String docName = entry.getKey();
        int TF = entry.getValue();
        double TF_IDF = TF * IDF;
        Text value = new Text(String.format("%.2f, ", TF_IDF));
        context.write(new Text(docName + "[" + key.toString() + "]"), value);
    }
}
```

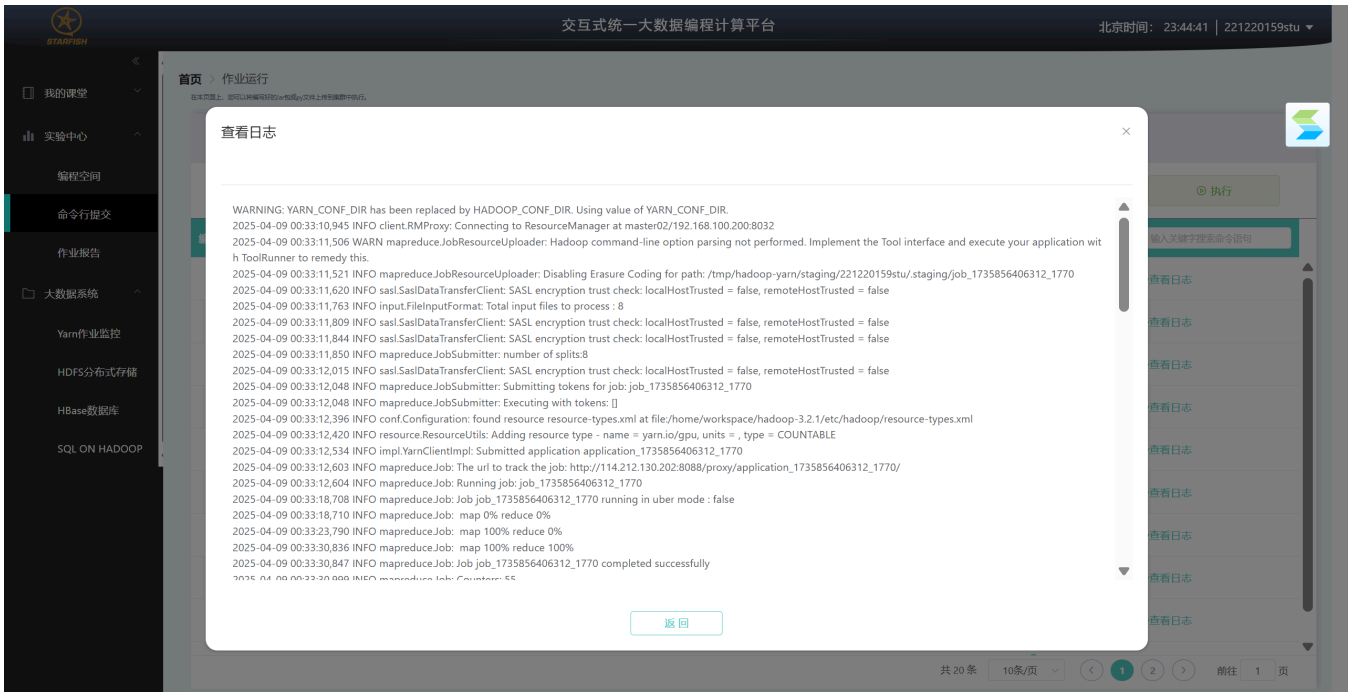
运行结果

集群执行结果

输入命令 `yarn jar BD2.jar TF_IDF /user/root/Exp2 output-Exp2/2-2`

输出结果存储在 `/user/221220159stu/output-Exp2/2-2`





三、去除停用词，输出排序后的结果

仿照`WordCount2.0`，输入时需新加一个参数表示停用词文档地址，
在`main`函数里新加一句`job.addCacheFile(new Path(args[2]).toUri());`

在`Mapper`的`setup`阶段读取停用词文档，并将停用词全都加到一个集合`patternsToSkip`里

```
private Set<String> patternsToSkip = new HashSet<String>();
private BufferedReader fis;

public void setup(Context context) throws IOException, InterruptedException {
    //String fileName = "/home/nightheron/dataset/cn_stopwords.txt";
    Configuration conf = context.getConfiguration();
    URI[] patternsURIs = Job.getInstance(conf).getCacheFiles();
    for (URI patternsURI : patternsURIs) {
        Path patternsPath = new Path(patternsURI.getPath());
        String fileName = patternsPath.getName().toString();
        fis = new BufferedReader(new FileReader(fileName));
        String pattern = null;
        while ((pattern = fis.readLine()) != null) {
            patternsToSkip.add(pattern);
        }
    }
}
```

之后在`map`函数里，只有词语不在停用词集合里才写入

```
if (!patternsToSkip.contains(word.toString())) {
    context.write(word, new Text(fileName));
}
```

`Reducer`与之前一致
运行完成后，还需要再通过之前完成的`Sort`进行分类。

运行结果

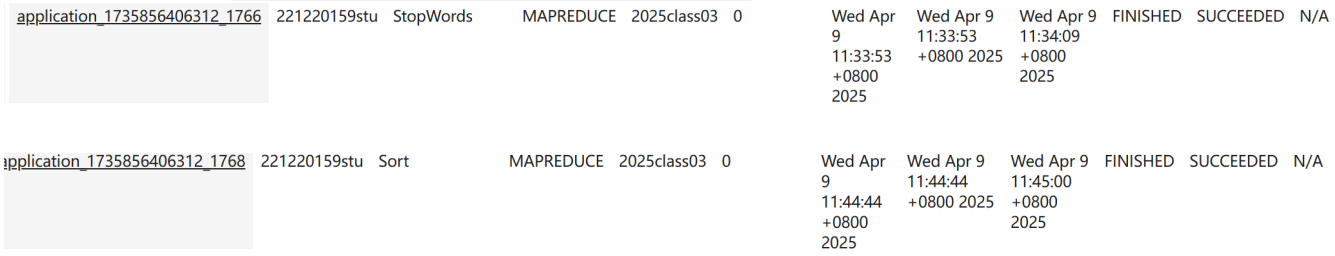
集群执行结果

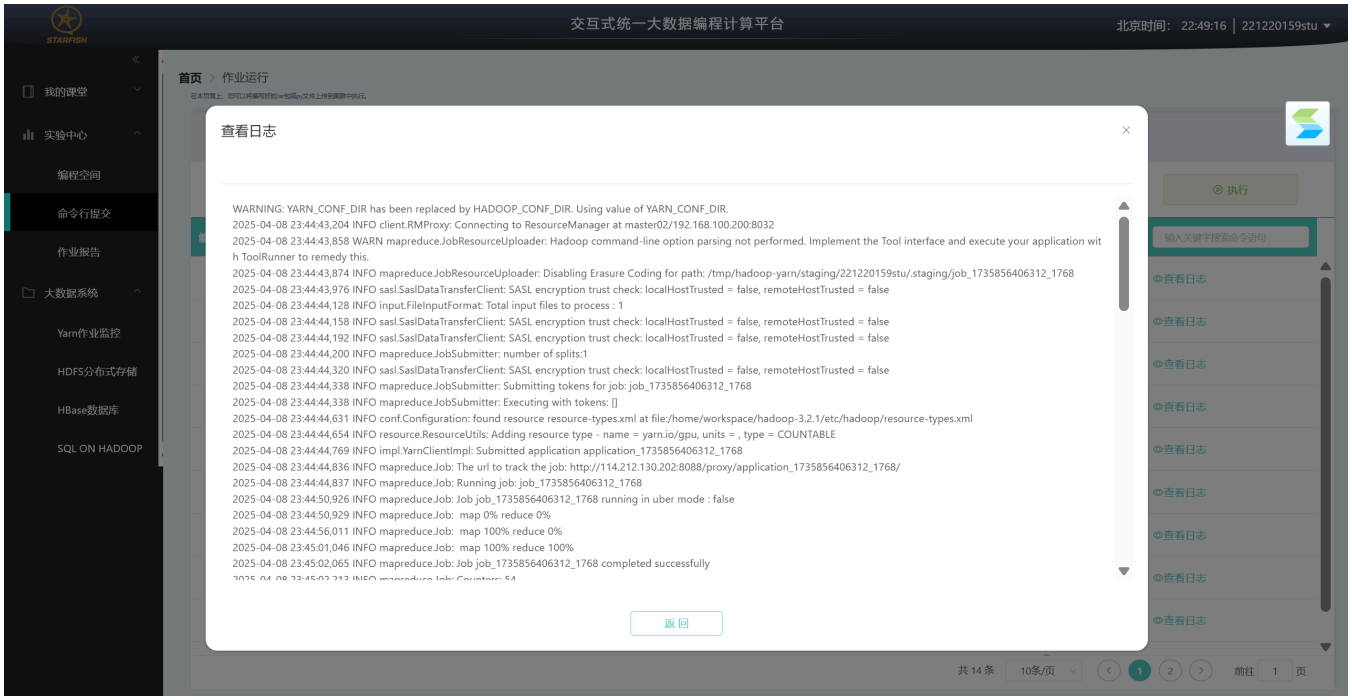
依次输入命令

① `yarn jar BD2.jar StopWords /user/root/Exp2 output-Exp2/3-1 hdfs://hcdsj/user/root/Exp2/cn_stopwor`

② `yarn jar BD2.jar Sort /user/221220159stu/output-Exp2/3-1 output-Exp2/3-2`

输出结果存储在 `/user/221220159stu/output-Exp2/3-2`




交互式统一大数据编程计算平台
北京时间： 22:49:16 | 221220159stu
我的课堂
实验中心
编程空间
命令行提交
作业报告
大数据系统
Yarn作业监控
HDFS分布式存储
HBase数据库
SQL ON HADOOP
首页 作业运行
执行
查看日志
WARNING: YARN_CONF_DIR has been replaced by HADOOP_CONF_DIR. Using value of YARN_CONF_DIR.
2025-04-08 23:44:43,204 INFO client.RMProxy: Connecting to ResourceManager at master02/192.168.100.200:8032
2025-04-08 23:44:43,858 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-04-08 23:44:43,874 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/221220159stu/.staging/job_1735856406312_1768
2025-04-08 23:44:43,976 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2025-04-08 23:44:44,128 INFO input.FileInputFormat: Total input files to process : 1
2025-04-08 23:44:44,158 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2025-04-08 23:44:44,192 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2025-04-08 23:44:44,200 INFO mapreduce.JobSubmitter: number of splits:1
2025-04-08 23:44:44,320 INFO sasl.SaslDataTransferClient: SASL encryption trust check: localHostTrusted = false, remoteHostTrusted = false
2025-04-08 23:44:44,338 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1735856406312_1768
2025-04-08 23:44:44,338 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-04-08 23:44:44,631 INFO conf.Configuration: found resource resource-types.xml at file:/home/workspace/hadoop-3.2.1/etc/hadoop/resource-types.xml
2025-04-08 23:44:44,654 INFO resource.ResourceUtils: Adding resource type - name = yarn.io/gpu, units = , type = COUNTABLE
2025-04-08 23:44:44,769 INFO impl.YarnClientImpl: Submitted application application_1735856406312_1768
2025-04-08 23:44:44,836 INFO mapreduce.Job: The url to track the job: http://114.212.130.202:8088/proxy/application_1735856406312_1768/
2025-04-08 23:44:44,837 INFO mapreduce.Job: Running job: job_1735856406312_1768
2025-04-08 23:44:50,926 INFO mapreduce.Job: Job job_1735856406312_1768 running in uber mode : false
2025-04-08 23:44:50,929 INFO mapreduce.Job: map 0% reduce 0%
2025-04-08 23:44:56,011 INFO mapreduce.Job: map 100% reduce 0%
2025-04-08 23:45:01,046 INFO mapreduce.Job: map 100% reduce 100%
2025-04-08 23:45:02,065 INFO mapreduce.Job: Job job_1735856406312_1768 completed successfully
返回
查看日志
共 14 条
10条/页
前往 1 页