# 文档的倒排索引与二级索引实现方案

## 小组成员

| 组员姓名 | 学号 | 教学班 |
| --- | --- | --- |
| 刘博文 | 1120201883 | 07112004 |
| 张开元 | 1120202079 | 07112004 |
| 许康哲 | 1120201702 | 07112003 |
| 刘勇奇 | 1120201334 | 07152001 |

## Hadoop集群搭建

### Java环境

本次实验的Java版本是oracle jdk 1.8\_341

### Hadoop集群分布式安装

#### 集群规划

| 主机 | 角色 |
| --- | --- |
| node1 | NameNode DataNode ResourceManager NodeManager |
| node2 | SecondaryNameNode DataNode NodeManager |
| node3 | DataNode NodeManager |

#### 集群节点设置

1. 修改主机名
2. host映射
3. 关闭防火墙
4. 开启ssh免密登录

#### Hadoop安装

1. 解压Hadoop安装包
2. 修改以下配置文件：hadoop-env.sh、core-site.xml、hdfs-site.xml、mapred-site.xml、yarn-site.xml、workers。因为配置文件很多，要修改的内容也很多，故不再把这些配置文件的所有内容在此列出
3. 添加hadoop到环境变量
4. 首次启动之前格式化namenode

### HBase安装

#### ZooKeeper搭建

1. 下载并解压zookeeper：tar -xvf apache-zookeeper-3.7.1-bin.tar.gz
2. 配置

* cd /export/server/zookeeper/apache-zookeeper-3.7.1-bin/conf  
  cp zoo\_sample.cfg zoo.cfg  
  #修改zoo.cfg文件，修改或添加如下内容  
  vim zoo.cfg  
  dataDir=/opt/apache-zookeeper-3.7.1-bin/data  
  server.1=node1:2888:3888  
  server.2=node2:2888:3888  
  server.3=node3:2888:3888  
  #配置myid，设置当前服务器的编号  
  cd /export/server/zookeeper/apache-zookeeper-3.7.1-bin  
  mkdir data  
  cd data  
  echo 1 > myid  
  #将zookeeper文件夹远程拷贝到另外两台虚拟机中，并分别修改myid为2,3  
  scp -r apache-zookeeper-3.7.1-bin node2:/export/server/zookeeper  
  scp -r apache-zookeeper-3.7.1-bin node3:/export/server/zookeeper

#### HBase搭建

1. 下载并解压hbase：tar -xvf hbase-2.4.14-bin.tar.gz -C /opt/
2. 配置

* cd /export/server/hbase/hbase-2.4.14/conf  
  cp hbase-env.sh hbase-env.sh.bak  
  #修改hbase-env.sh  
  export JAVA\_HOME=/export/server/jdk1.8.0\_241  
  export HBASE\_MANAGES\_ZK=false //告诉hbase使用外部的zookeeper  
  #备份并修改conf下hbase-site.xml为如下内容  
   <property>  
   <name>hbase.rootdir</name>  
   <value>hdfs://node1:8020/hbase</value>  
   </property>  
    
   <property>  
   <name>hbase.cluster.distributed</name>  
   <value>true</value>  
   </property>  
    
   <property>  
   <name>hbase.zookeeper.quorum</name>  
   <value>node1:2181,node2:2181,node3:2181</value>  
   </property>  
    
   <property>  
   <name>hbase.master.info.port</name>  
   <value>60010</value>  
   </property>  
    
   <property>  
   <name>hbase.wal.provider</name>  
   <value>filesystem</value>  
   </property>  
  #配置conf路径下regionservers，添加如下内容  
  node1  
  node2  
  node3  
  #将hbase文件夹远程拷贝到另外两台虚拟机中  
  scp -r hbase-2.4.14/ node2:/export/server/hbase  
  scp -r hbase-2.4.14/ node3:/export/server/hbase

### 集群启动

1. 启动zookeeper：

* /export/server/zookeeper/apache-zookeeper-3.7.1-bin/bin/zkServer.sh start  
  /export/server/zookeeper/apache-zookeeper-3.7.1-bin/bin/zkServer.sh status

1. 启动hadoop：

* start-dfs.sh  
  start-yarn.sh

1. 启动Hbase：

* /export/server/hbase/hbase-2.4.14/bin/start-hbase.sh

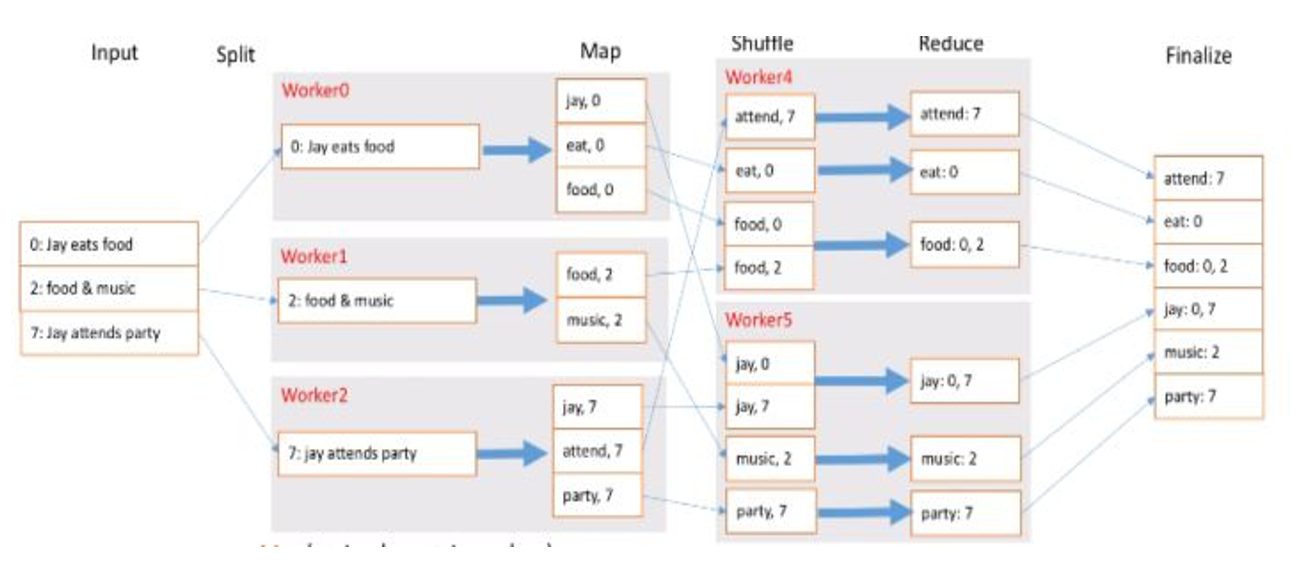
## 实现方案

### 倒排索引

InvertedIndex.java:

import java.util.\*;  
import java.io.IOException;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Set;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.FileSystem;  
import org.apache.hadoop.fs.Path;  
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.input.FileSplit;  
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;  
import org.apache.hadoop.util.GenericOptionsParser;  
  
import org.apache.hadoop.hbase.HBaseConfiguration;  
import org.apache.hadoop.hbase.client.Put;  
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;  
import org.apache.hadoop.hbase.mapreduce.TableMapReduceUtil;  
import org.apache.hadoop.hbase.mapreduce.TableReducer;  
import org.apache.hadoop.hbase.util.Bytes;  
  
public class InvertedIndex {  
 //偏移量\文本\单词\文件编号  
 static class MyMapper extends Mapper<LongWritable, Text, Text, Text> {  
 String fileName = null;  
 Text mk = new Text(); //map端输出的键  
 Text mv = new Text(); //map端输出的值  
  
 // 在setup中通过切片来获取文件名  
 @Override  
 protected void setup(Mapper<LongWritable, Text, Text, Text>.Context context)  
 throws IOException, InterruptedException {  
 FileSplit files = (FileSplit) context.getInputSplit();  
 fileName = files.getPath().getName();  
 }  
  
 @Override  
 //偏移量，字符串，暂存处理结果  
 protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, Text>.Context context)  
 throws IOException, InterruptedException {  
  
 String[] datas = value.toString().split(" ");  
 String line = datas[0];  
 int line\_num = Integer.parseInt(line);  
 //文件编号  
 int file\_num = line\_num / 100000 + 1;  
 //map存储单词和出现次数  
 Map<String, Integer> map = new HashMap<String, Integer>();  
 //更新单词出现次数  
 for (int i = 1; i < datas.length; i++) {  
 String word = datas[i];  
 if (map.containsKey(word))  
 map.put(word, map.get(word) + 1);  
 else  
 map.put(word, 1);  
 }  
 //获取所有单词  
 Set<String> keySet = map.keySet();  
 for (String s : keySet) {  
 mk.set(s);  
 // 拼接信息，这里本来写入了文件名、行号和出现次数  
 //但考虑到hbase写入内容的大小上限，仅写入单词出现的文件编号  
 // mv.set(fileName + ":" + line + "," + map.get(s));  
 mv.set(Integer.toString(file\_num));  
 context.write(mk, mv);  
 }  
  
 }  
  
 }  
  
  
 static class MyReducer extends TableReducer<Text, Text, ImmutableBytesWritable> {  
 Text rv = new Text();  
  
 @Override  
 protected void reduce(Text key, Iterable<Text> values, Context context)  
 throws IOException, InterruptedException {  
 // 用StringBuilde来进行拼接  
 StringBuilder s = new StringBuilder();  
 //存储该单次出现的所有文件的编号  
 List<String> mylist = new ArrayList<>();  
 for (Text v : values) {  
 String str1 = v.toString();  
 if (!mylist.contains(str1)) {  
 mylist.add(str1);  
 }  
 }  
 for (String i : mylist) {  
 s.append(i).append(",");  
 }  
 // 去掉最后一个符号  
 rv.set(s.substring(0, s.length() - 1));  
  
 //设置单次为Row Key 构建Put对象  
 Put put = new Put(key.toString().getBytes());  
 //指定插入的列族、列名和值  
 put.addColumn("col\_family".getBytes(), "info".getBytes(), rv.toString().getBytes());  
 context.write(null, put);  
 }  
  
 }  
  
 public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {  
  
 Configuration conf = new Configuration();  
 conf = HBaseConfiguration.create(conf);  
  
 String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();  
  
 Job job = Job.getInstance(conf);  
  
 job.setJarByClass(InvertedIndex.class);  
  
 job.setMapperClass(MyMapper.class);  
  
 job.setReducerClass(MyReducer.class);  
  
 job.setMapOutputKeyClass(Text.class);  
 job.setMapOutputValueClass(Text.class);  
  
 job.setOutputKeyClass(Text.class);  
 job.setOutputValueClass(Text.class);  
 // 设置HBase表输出：表名，reducer类  
 TableMapReduceUtil.initTableReducerJob("test\_table", MyReducer.class, job);  
  
 FileInputFormat.addInputPath(job, new Path(otherArgs[0]));  
  
 job.waitForCompletion(true);  
 }  
  
}

mapreduce过程示意图：



#### map阶段

假设输入为"0 Jay eats food"，map函数会将该句的所有单词按照空格分离，那么第一个“单词”显然就是该行文档标题的编号，所以从第二个单词开始，分别以这些单词为key，以编号为value作为输出

#### reduce阶段

reduce阶段将相同key的键值对合并，合并的具体方式是key不变，value的值以分号为分隔符放到一起，并最终将其写入HBase中

#### 结果

最终得到一个Rowkey为单词，列为该单词出现在了哪些编号的文档中的HBase表

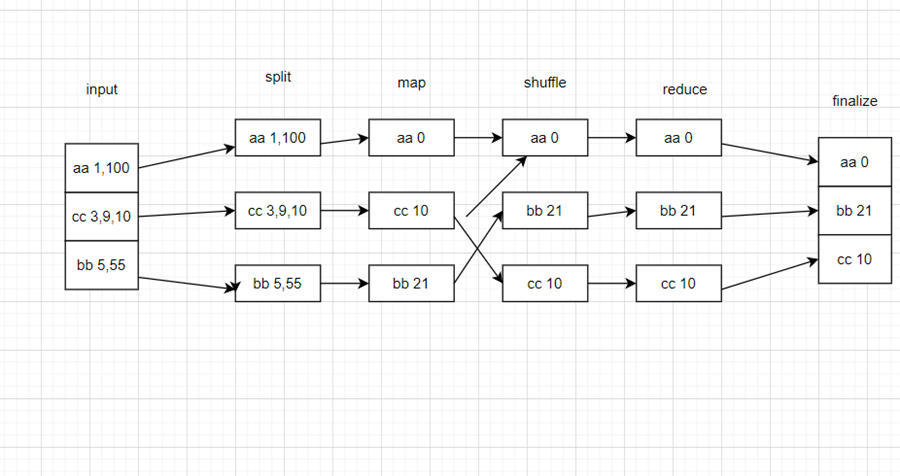
### 二级索引

在实现了倒排索引之后，原本总共1.34GB的数据集在被计算之后变成了1.93GB大小的倒排索引文件，这么大的一个倒排索引文件不方便直接读到内存中，所以需要再对该倒排索引文件建立一个二级索引，并将该索引直接加载到内存中，加快读取速度。

SecondaryIndex.java

import java.io.IOException;
  
  
import org.apache.hadoop.conf.Configuration;
  
import org.apache.hadoop.fs.Path;
  
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;
  
  
public class SecondaryIndex {
  
  
 private static class SecondaryIndexMapper extends Mapper<LongWritable, Text,Text,Text> {
  
  
 Text mapKeyOut = new Text();
  
 Text mapValueOut = new Text();
  
  
 @Override
  
 protected void map(LongWritable key, Text value, Mapper<LongWritable, Text, Text, Text>.Context context)
  
 throws IOException, InterruptedException {
  
 //将一行数据以空格为分隔符读入datas
  
 String[] datas = value.toString().split(" ");
  
 //datas中第一个元素即是该行的最前面的单词
  
 String word = datas[0];
  
  
 //输出key设置为目标单词，输出value设置为行数
  
 mapKeyOut.set(word);
  
 mapValueOut.set(key.toString());
  
 context.write(mapKeyOut,mapValueOut);
  
 }
  
 }
  
  
 private static class SecondaryIndexReducer
  
 extends Reducer<Text, Text, Text,Text> {
  
 @Override
  
 protected void reduce(Text key, Iterable<Text> values, Reducer<Text, Text, Text, Text>.Context context) throws IOException, InterruptedException {
  
 super.reduce(key, values, context);
  
 }
  
 }
  
  
 public static void main(String[] args)
  
 throws IOException, ClassNotFoundException, InterruptedException{
  
 //获取conf
  
 Configuration conf = new Configuration();
  
 //创建job对象
  
 Job job = Job.getInstance(conf, "SecondaryIndex");
  
 //指定Jar包主类
  
 job.setJarByClass(SecondaryIndex.class);
  
 //设置mapper类
  
 job.setMapperClass(SecondaryIndexMapper.class);
  
 //设置Combiner类，一般设定为Reducer类
  
 job.setCombinerClass(SecondaryIndexReducer.class);
  
 //设置Reducer类
  
 job.setReducerClass(SecondaryIndexReducer.class);
  
 //设置输出key类型
  
 job.setOutputKeyClass(Text.class);
  
 //设置输出value类型
  
 job.setOutputValueClass(Text.class);
  
 //设置文件输入路径
  
 FileInputFormat.addInputPath(job, new Path(args[0]));
  
 //设置文件输出路径
  
 FileOutputFormat.setOutputPath(job, new Path(args[1]));
  
 //提交作业
  
 job.waitForCompletion(true);
  
 }
  
  
}

二级索引的代码实现相比倒排索引实现简单很多，mapreduce过程：



#### map阶段

假设读入一行内容为："at 10086;10087"，那么map阶段会将该行的单词作为输出的key值，在此例子中是单词"at"；除此之外，map阶段输出的value则是该行在输入文件的字节偏移量，在这里我们假设是10010，那么map阶段最后的输出为："at 10010"

#### reduce阶段

因为在map阶段实际上我们已经完成了对数据的处理，所以reduce阶段直接将map阶段后的输出写入即可

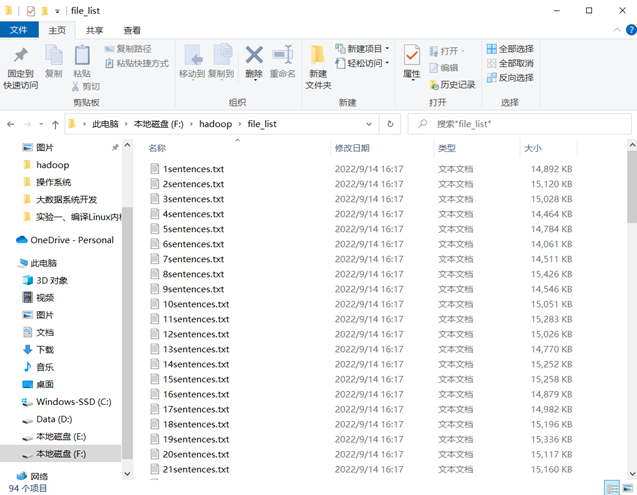
#### 结果

最终得到的文件是对倒排索引的一个二级索引，每一行是单词以及该单词在倒排索引文件的字节偏移量，通过该偏移量可以迅速的在倒排索引文件中找到该单词。

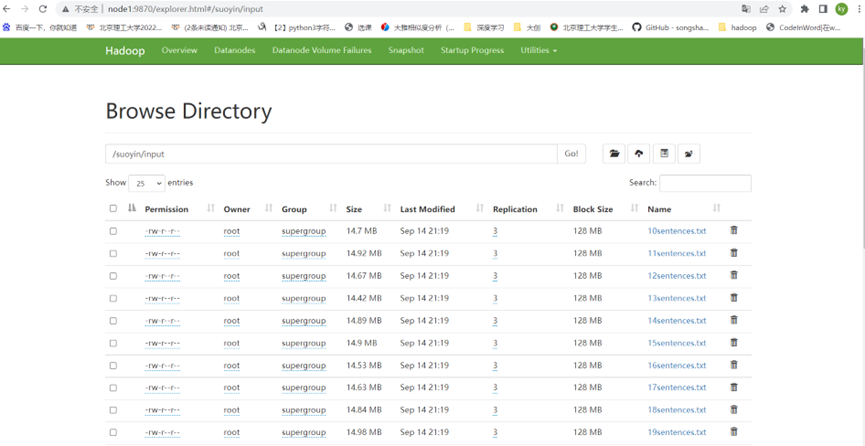
## 部署运行

### 倒排索引

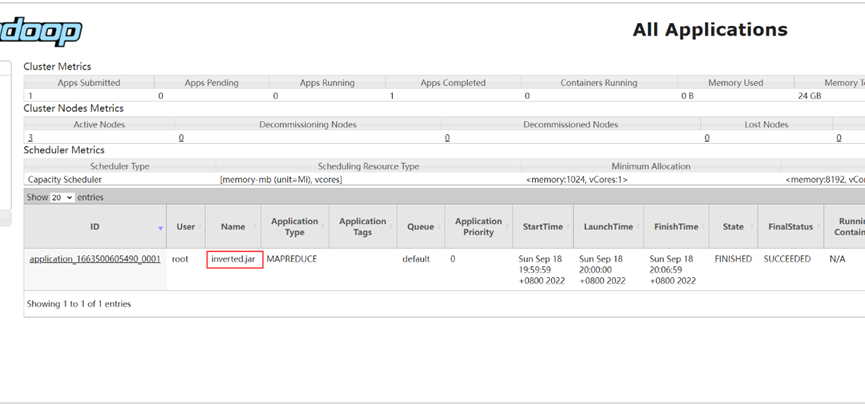
1. 分割源txt文件

* 

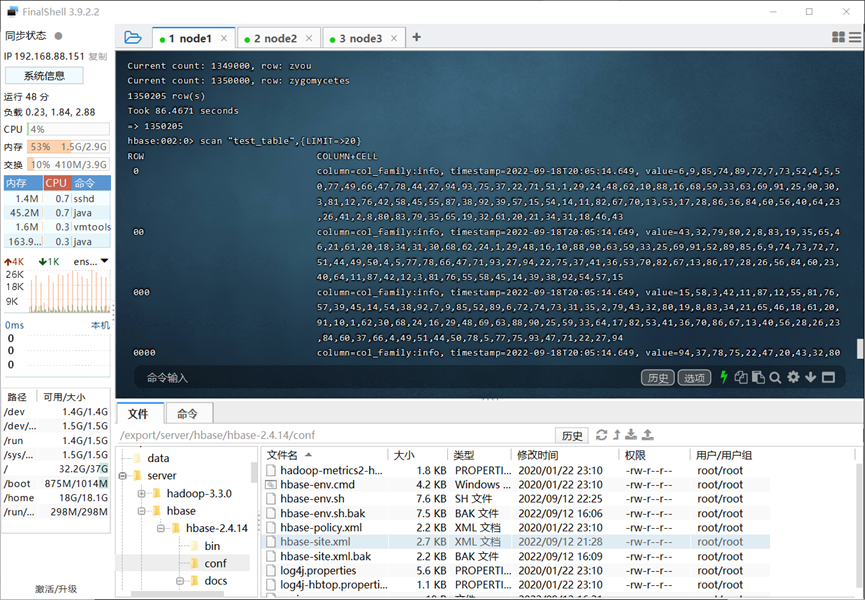
1. 上传文件

* 

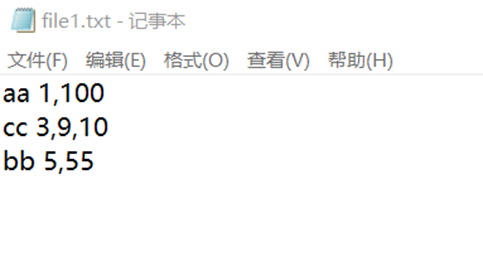
1. 运行倒排索引代码

* hadoop jar inverted.jar InvertedIndex /suoyin/input
* 

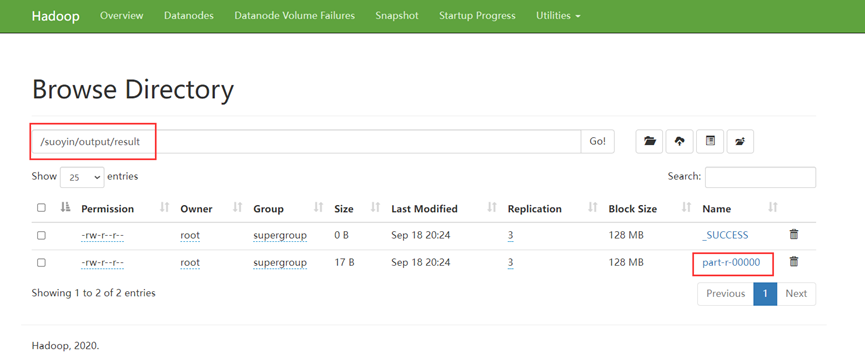
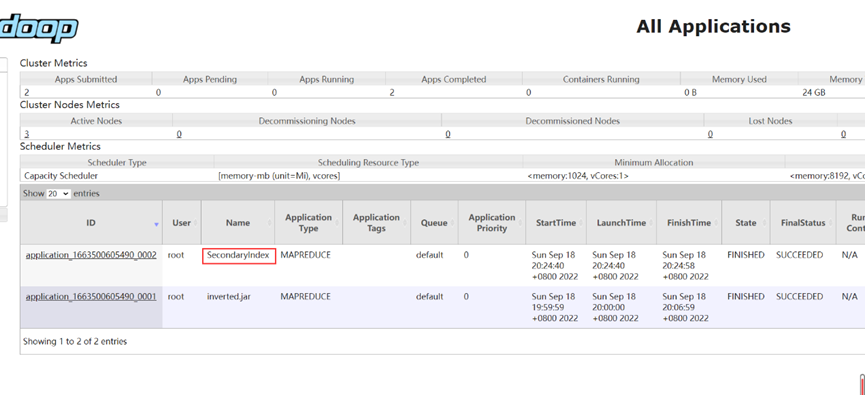
1. 查看结果

* 

### 二级索引

这里以一个简易的file1.txt作为输入文件，测试二级索引的功能，其内容如下：

1. 运行二级索引代码

* hadoop jar second.jar SecondaryIndex /suoyin/second\_input /suoyin/out/result
* 

1. 查看结果

* 