**成绩**



**大数据系统开发实践项目**

**实验报告**

**题　　目： 大数据系统开发实践**

**学 院： 计算机学院**

**专业名称： 计算机科学与技术**

**小组成员： 刘秉致 1120220715**

**史立彬 1120220701**

**陈依凡 1120222908**

**武瑗恬 1120221592**

**董顺心 1120222190**

**任课教师： 郭贵锁**

目录

[小组成员 4](#_Toc178544242)

[搜索引擎实现的技术方案 4](#_Toc178544243)

[功能描述： 4](#_Toc178544244)

[技术选型： 4](#_Toc178544245)

[功能实现： 4](#_Toc178544246)

[1. 数据收集： 4](#_Toc178544247)

[2. 数据预处理 4](#_Toc178544248)

[3. 构建倒排索引： 5](#_Toc178544249)

[4. 索引存储： 5](#_Toc178544250)

[5. 用户查询处理： 5](#_Toc178544251)

[6. 排序和结果呈现： 5](#_Toc178544252)

[7. 实例分析 5](#_Toc178544253)

[工作计划： 5](#_Toc178544254)

[组织架构： 5](#_Toc178544255)

[软件功能： 5](#_Toc178544256)

[1. 建立搜索引擎库： 5](#_Toc178544257)

[2. 在索引数据库中检索关键内容： 6](#_Toc178544258)

[软件质量保证： 6](#_Toc178544259)

[1. 功能性测试： 6](#_Toc178544260)

[2. 性能测试： 6](#_Toc178544261)

[3. 软件的非功能性保证： 6](#_Toc178544262)

[文档的倒排索引与单台电脑多节点的伪分布式实现方案 6](#_Toc178544263)

[实验要求 6](#_Toc178544264)

[数据准备 7](#_Toc178544265)

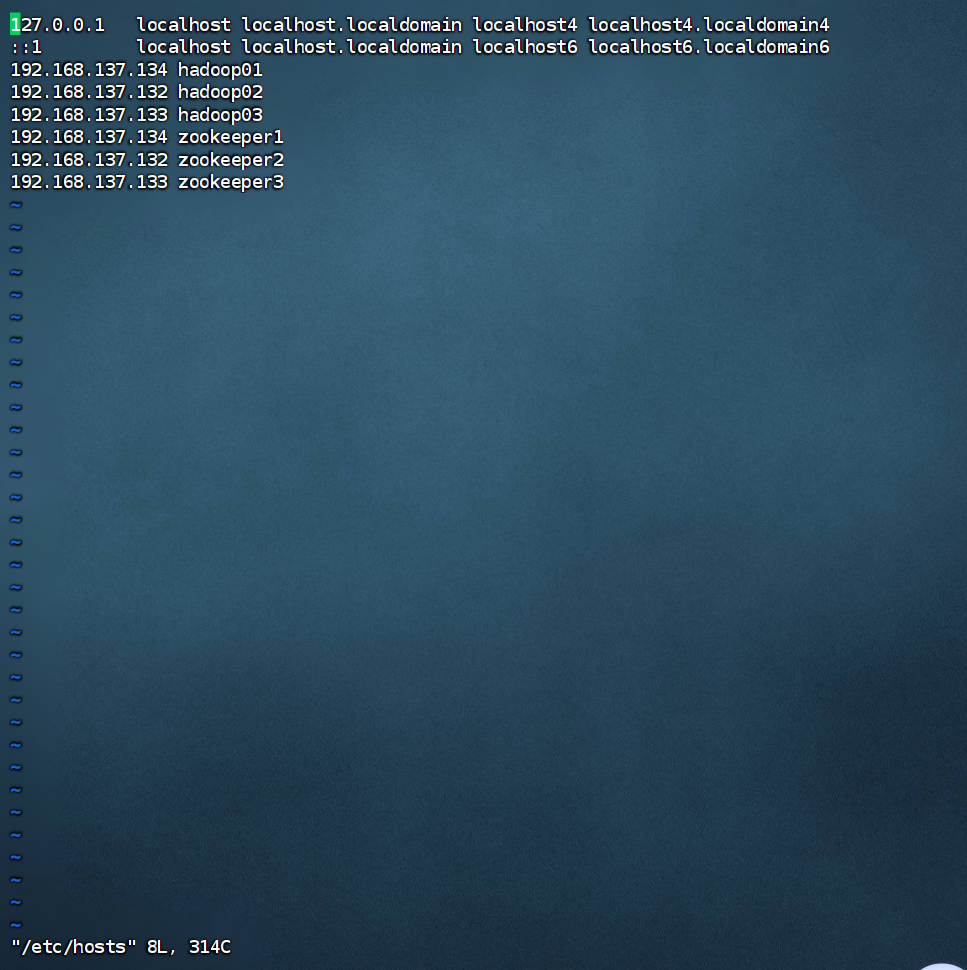
[Hadoop集群环境搭建与配置 7](#_Toc178544266)

[1. 软件环境 7](#_Toc178544267)

[2. 修改主机名 7](#_Toc178544268)

[3. 网络环境配置 7](#_Toc178544269)

[4. 修改每台电脑IP映射 7](#_Toc178544270)

[5. 配置ssh免密登录 8](#_Toc178544271)

[6. 关闭各节点防火墙 8](#_Toc178544272)

[7. 修改环境变量 9](#_Toc178544273)

[8. 在各节点上修改配置文件 9](#_Toc178544274)

[9. 启动Hadoop+zookeeper+hbase 11](#_Toc178544275)

[10. 实验中发现的坑 12](#_Toc178544276)

[11. 单机伪分布式展示 12](#_Toc178544277)

[算法及实现方案 13](#_Toc178544278)

[1. 拆分文件 13](#_Toc178544279)

[2. 倒排索引的原理与代码实现： 15](#_Toc178544280)

[运行结果与分析 19](#_Toc178544281)

[1. 分割源txt文件 19](#_Toc178544282)

[2. 上传文件 20](#_Toc178544283)

[3. 在HBase中建立空InvertedIndexTable 20](#_Toc178544284)

[4. 运行倒排索引代码 21](#_Toc178544285)

[5. 查看结果 23](#_Toc178544286)

[心得体会 23](#_Toc178544287)

# 小组成员

| 组员姓名 | 学号 | 教学班 |
| --- | --- | --- |
| 刘秉致 | 1120220715 | 07112201 |
| 陈依凡 | 1120222908 | 07112204 |
| 史立彬 | 1120220701 | 07112201 |
| 董顺心 | 1120222190 | 07112203 |
| 武瑗恬 | 1120221592 | 63012221 |

# 搜索引擎实现的技术方案

## 功能描述：

基于Hadoop生态圈，构建一个简单的分布式数据储存的搜索引擎，实现将数据集分布式的储存在集群中，由用户输入文本，依照输入的相关性对可能的结果进行排序，并返回结果。

## 技术选型：

在构建当前搜索引擎的数据集时，考虑到数据搜集的过程中涉及到大量类型多样的数据内容，如网页、文档和图片等，考虑直接使用Scrapy对需要获取的数据直接进行截取。Scrapy是基于Python开发的一个快速、高层次的屏幕抓取和web抓取框架，用于抓取web站点并从页面中提取结构化的数据。

由于涉及的数据量庞大，数据结构多样，同时有分布式储存的基本需求，考虑使用Hadoop生态圈下的HDFS文件储存系统实现文本储存功能，利用HBase数据库实现索引储存和查询。HDFS主要用于存储大量的数据，并支持高吞吐量的数据访问，适用于批处理。HBase则基于列式存储实现，提供了高度可扩展的表格存储和快速查询。

在获取大量所需数据后，考虑到需要对这些数据进行预处理，并且在实现搜索引擎的过程中，需要对大量数据进行操作以构建倒排索引，考虑采用Hadoop生态圈内的MapReduce框架来实现数据处理功能。MapReduce是一个分布式、并行处理的计算框架，可以并行的、快速的访问和处理HDFS文件系统中储存的文本数据。

索引构建完成并储存后，需要构建一个查询接口来接收并处理用户的查询请求，来返回具有一定相关性的查询结果，并将查询结果按照相关程度大小进行排序，考虑使用Apache Solr来构建搜索接口处理请求。Apache Solr基于Lucene库，是一个高性能、可伸缩的企业级搜索平台，具有丰富的功能集和多样的API，并支持分布式部署和低延迟响应。

## 功能实现：

### 数据收集：

使用Scrapy对目标数据源进行数据采集，获得需要的结构化数据。

### 数据预处理

对搜集到的数据进行清理和预处理，如去除HTML标签、去除停用词、分词、词干提取等操作。

将预处理后的数据储存在HDFS中备用。

### 构建倒排索引：

现代搜索引擎多采用倒排索引对数据进行组织。区别于正排索引“文档到单词”的映射结构，倒排索引是一个“单词到文档”的映射，用于快速检索文档。这里使用Hadoop框架下的MapReduce来构建倒排索引。

Map阶段：将文档拆分为单词列表，并将每个单词与文档ID关联。

Reduce阶段：将单词与文档ID列表关联起来，构建倒排索引。

### 索引存储：

使用HBase存储构建好的倒排索引以备使用。

### 用户查询处理：

接收用户的搜索查询（假设为一组文字描述），将查询描述分词、词干化等预处理。

使用倒排索引以查询描述中各查询词为索引快速定位包含查询词的文档。

### 排序和结果呈现：

根据查询词的匹配程度和其他相关性指标，使用TF-IDF（词频-逆文档）方法，对搜索结果进行排序，并将排序后的结果呈现给用户。

### 实例分析

首先，使用Scrapy,通过大量爬取网页，以获得原始数据。

获得原始数据后，我们将每个文件分解成单词，构建数据的单词集合，并使用TF-IDF分析工具计算每个单词在其出现的各个网页中的重要性。最后，生成包括每个单词、它所出现的各个文件及其对应重要性等信息的倒排索引文件。

当用户输入搜索内容后，搜索引擎会将搜索内容分解成单词，并查找它们在倒排索引文件中的位置。然后，这条索引及其关联的文档将被提取出来，并对于在同一文件中出现的单词，计算它们的重要性之和，按降序排列，呈现给用户。

## 工作计划：

上课（9.2-9.5）

搜索资料并学习Hadoop的使用。（9.5-9.12）

数据处理（数据集的拆分、数据清洗）。（9.12-9.19）

集群分布式实现倒排索引并编写搜索引擎技术文档。（9.19-9.25）

## 组织架构：

环境搭建与代码执行：刘秉致

文档整理编写：

## 软件功能：

### 建立搜索引擎库：

利用Hadoop框架，使用HDFS作为底层的存储和访问工具，结合Scrapy网络爬虫工具，采用广度优先搜索策略和以PageRank为权的选择策略，实现对用户查询相关的网页的爬取，并将其中的页面内容进行提取和相关信息的存储。

获取了保存在HDFS中的网页的文本信息后，可以利用MapReduce构建一个倒排索引。这个倒排索引可以存储在HBase中，形成一个快速、可扩展的索引数据库。

这个系统的工作流程如下：

利用MapReduce框架编写程序，将数据集进行处理，构建倒排索引。在倒排索引中，每个单词将与它出现在哪些网页、在这些网页中的位置以及单词在文件中的重要性等信息相关联。

构建的倒排索引可以存储在HBase中，作为一个高效的索引数据库。这样，用户查询时可以快速地定位相关的网页，并提供相关信息。

通过这个系统，用户可以进行高效的信息检索，而Hadoop、HDFS、MapReduce、HBase、Apache Solr等技术的结合为构建大规模、高性能的搜索引擎提供了可行的基础。

### 在索引数据库中检索关键内容：

用户输入搜索关键词并点击搜索按钮后，基于Apache Solr构建的系统的检索程序启动。接着，通过倒排索引，从网页索引数据库中检索包含用户搜索内容的所有相关网页。最后，系统对搜索结果进行排序处理，并将处理完毕的文档呈现给用户在客户端上浏览。

## 软件质量保证：

### 功能性测试：

确保软件按照规格说明书中的要求执行其基本功能。

验证每个功能是否按预期工作，包括输入验证、输出正确性等。

### 性能测试：

测试软件的极端负载阈值。

测试软件在正常和极端负载下的性能，包括响应时间、吞吐量和资源利用率等。

评估软件的性能是否满足预期的需求。

### 软件的非功能性保证：

测试搜索引擎是否满足快速和便捷的要求，以及能否对用户的打开和检索请求作出及时响应。

此外，搜索引擎的Web应用程序也需要满足简洁和美观的需求和符合人机交互逻辑的布局设计，实现人机交互友好。

# 文档的倒排索引与单台电脑多节点的伪分布式实现方案

## 实验要求

运用MapReduce为程序框架，对给定的文本文件构建一个倒排索引, 将倒排索引存储在HBase中。

## 数据准备

将压缩文件sentences.txt.zip以一万为单位拆分数据集，具体实现见下文。

## Hadoop集群环境搭建与配置

### 软件环境

VMware：VMware Workstation 16.0.0 build-16894299

CentOS：CentOS Linux release 7.9.2009 (core)

Java：jdk1.8.0 \_411

Hadoop：hadoop-3.4.0

ZooKeeper：zookeeper-3.9.2

HBase：hbase-3.0.0

### 修改主机名

在本地主机上构建三台虚拟机，虚拟机名修改为Master，Worker1，Worker2。

| 主机 | 角色 |
| --- | --- |
| Master | 名字节点、主节点 |
| Worker1 | 数据节点、工作节点 |
| Worker2 | 数据节电、工作节点 |

### 网络环境配置

将三台虚拟机的网络模式全部选为NAT模式，使用VM提供的本机虚拟网卡进行虚拟机的网络搭建。将三台虚拟机的IP分别配置为与虚拟网卡一致的主网段下的不同子网段。

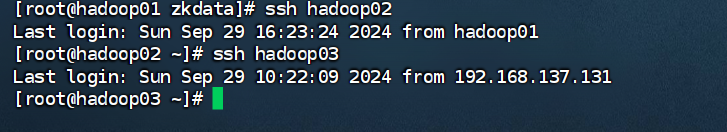
| 主机 | IP地址 |
| --- | --- |
| Master | 192.168.41.201 |
| Worker1 | 192.168.41.202 |
| Worker2 | 192.168.41.203 |

### 修改每台电脑IP映射

在本机的host和各虚拟机的host中修改IP映射。

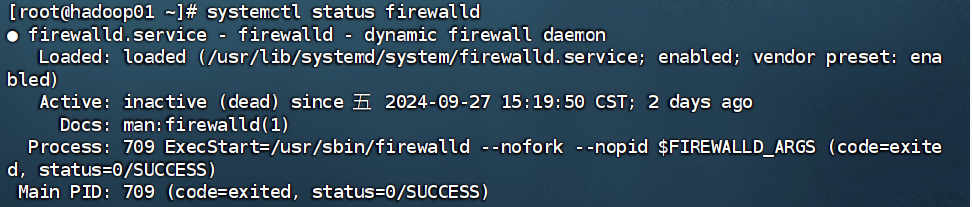
### 4eb6d5ae73784292adde6da43ef2b83配置ssh免密登录

下图为Master节点成功免密登录其他节点



### 关闭各节点防火墙

下图为主节点防火墙成功关闭后状态

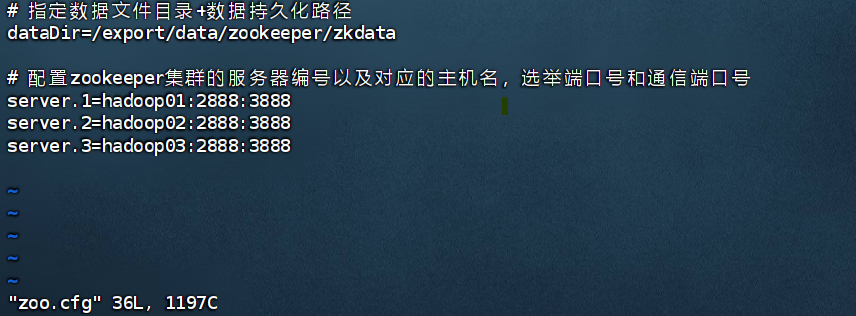


### 修改环境变量



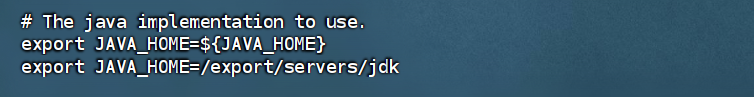
### 在各节点上修改配置文件

1. zookeeper:



1. HBase：

hbase-env.sh：



hbase-site.xml：

1. <property>
2. <name>hbase.rootdir</name>
3. <value>hdfs://master:9000/hbase</value>
4. </property>
5. <property>
6. <name>hbase.cluster.distributed</name>
7. <value>true</value>
8. </property>
9. <property>
10. <name>hbase.zookeeper.quorum</name>
11. <value>master,worker1,worker2</value>
12. </property>
13. <property>
14. <name>hbase.zookeeper.property.dataDir</name>
15. <value>/root/zookeeper-3.9.2/zkData</value>
16. </property>
17. <property>
18. <name>hbase.tmp.dir</name>
19. <value>$/root/hbase-3.0.0/tmp</value>
20. </property>
21. <property>
22. <name>hbase.unsafe.stream.capability.enforce</name>
23. <value>false</value>
24. </property>
25. <property>
26. <name>hbase.master.info.port</name>
27. <value>16010</value>
28. </property>
29. Hadoop：

core-site.xml:

1. <configuration>
2. <property>
3. <name>fs.defaultFS</name>
4. <value>hdfs://master:9000</value>
5. </property>
6. <property>
7. <name>hadoop.tmp.dir</name>
8. <value>/root/hadoopdata</value>
9. </property>
10. </configuration>

hdfs-site.xml:

1. <configuration>
2. <property>
3. <name>dfs.replication</name>
4. <value>2</value>
5. </property>
6. </configuration>

yarn-site.xml：

1. <configuration>
2. <property>
3. <name>yarn.resourcemanager.hostname</name>
4. <value>master</value>
5. </property>
6. <property>
7. <name>yarn.nodemanager.aux-services</name>
8. <value>mapreduce\_shuffle</value>
9. </property>
10. </configuration>

mapred-site.xml：

1. <configuration>
2. <property>
3. <name>mapreduce.framework.name</name>
4. <value>yarn</value>
5. </property>
6. </configuration>

### 启动Hadoop+zookeeper+hbase

编写bash脚本利用ssh远程操作实现一键启动zookeeper：

1. #!/bin/bash
2. case $1 in
3. "start"){
4. for i in master worker1 worker2
5. do
6. echo ---------- zookeeper $i 启动 ------------
7. ssh $i "/root/zookeeper-3.9.2/bin/zkServer.sh start"
8. done
9. };;
10. "stop"){
11. for i in master worker1 worker2
12. do
13. echo ---------- zookeeper $i 停止 ------------
14. ssh $i "/root/zookeeper-3.9.2/bin/zkServer.sh stop"
15. done
16. };;
17. "status"){
18. for i in master worker1 worker2
19. do
20. echo ---------- zookeeper $i 状态 ------------
21. ssh $i "/root/zookeeper-3.9.2/bin/zkServer.sh status"
22. done
23. };;
24. Esac

先启动zookeeper：

1. ./zk.sh start

接着启动hadoop，只需在master节点上运行一次即可。

1. start-all.sh

最后启动hbase:

1. start-hbase.sh

### 实验中发现的坑

1. 实验时如果Hadoop设置的活动端口与HBase设置的活动端口不一致，会导致HBase主节点无法正常启动。
2. 若环境变量配置错误可能会导致循环返回登录界面。此时应ctrl+alt+F3进入命令行界面。随后使用root用户登录，找到配置错误的文件并修改错误的配置，最后ctrl+alt+F1，返回登录界面即可正常登录。

### 单机伪分布式展示

以上是本次实验过程中所有的配置流程，我们搭建了单机伪分布式的环境，便于开发调试。从jps可以看出hadoop+zookeeper+hbase都以伪分布式的方式在运行。在实际的开发环境中，应该也是先在伪分布式上运行成功，然后再到真分布的生产环境。



## 算法及实现方案

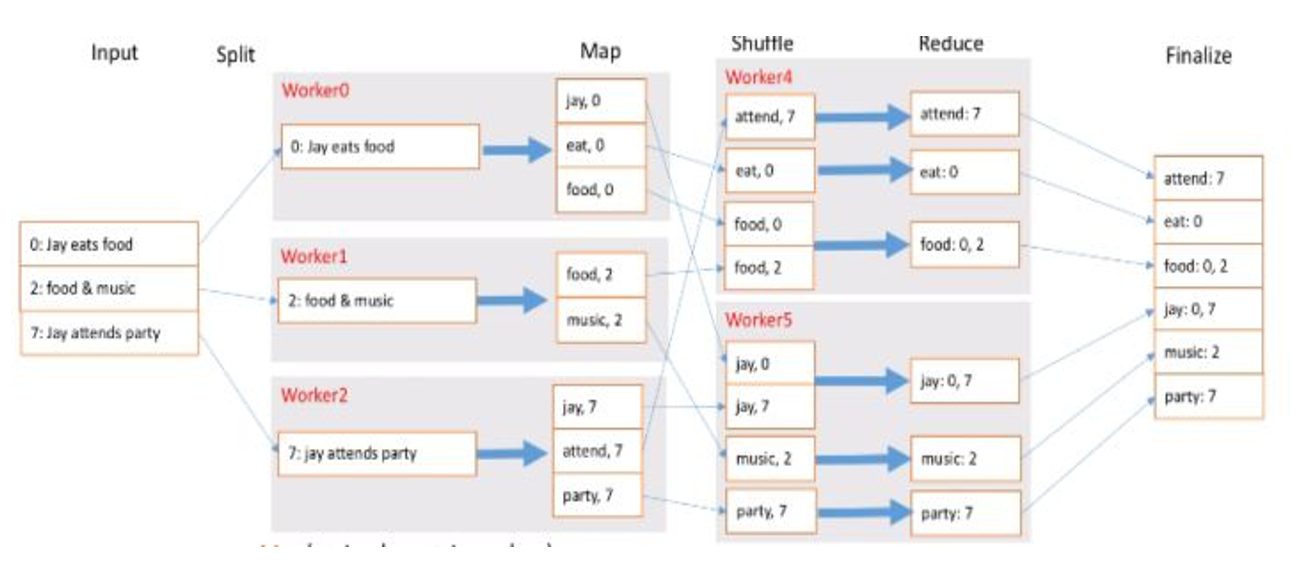
### 拆分文件

从实验要求可以得到，每个文件应该包含10000个句子，关注到每一行即为一个句子，因此只需要对文件按行进行拆分。我们使用java进行编程，编写了分割程序进行处理。以下是该部分的代码，处理完成后共940个文件：

1. //FileSplitter.java
2. import java.io.BufferedReader;
3. import java.io.BufferedWriter;
4. import java.io.FileReader;
5. import java.io.FileWriter;
6. import java.io.IOException;
7. import java.io.File;
8. public class FileSplitter {
9. public static void main(String[] args) {
10. // 源文件路径
11. String sourceFilePath = "sentences.txt";
12. // 目标文件路径前缀
13. File dataDirectory = new File("/root/SplitedData");//指定目标文件夹
14. if(!dataDirectory.exists()){
15. dataDirectory.mkdir();
16. }
17. String targetFilePrefix = "/root/SplitedData/split\_";
18. // 每个文件的最大行数
19. int maxLinesPerFile = 10000;
20. //为当前filereader给定buffer
21. try{
22. BufferedReader reader = new BufferedReader(new FileReader(sourceFilePath));
23. //行缓存
24. String line;
25. //计数器
26. int lineCount = 0;
27. int fileCount = 0;
28. BufferedWriter writer = new BufferedWriter(new FileWriter(targetFilePrefix + fileCount + ".txt"));
29. while ((line = reader.readLine()) != null) {
30. writer.write(line);
31. writer.newLine();
32. lineCount++;
33. if (lineCount >= maxLinesPerFile) {
34. writer.close();
35. lineCount = 0;
36. fileCount++;
37. writer = new BufferedWriter(new FileWriter(targetFilePrefix + fileCount + ".txt"));
38. }
39. }
40. writer.close();
41. reader.close();
42. }catch(IOException e){
43. e.printStackTrace();
44. }
45. System.out.println("文件分割完成");
46. }
47. }

### 倒排索引的原理与代码实现：

mapreduce过程示意图：



#### map阶段

当输入为”0 Jay eats food”时，map函数以空格为界，将该句中所有的单词分离，第一个单词就是该行文档的编号，所以从第二个单词开始，单词为key，编号为value进行输出。

#### reduce阶段

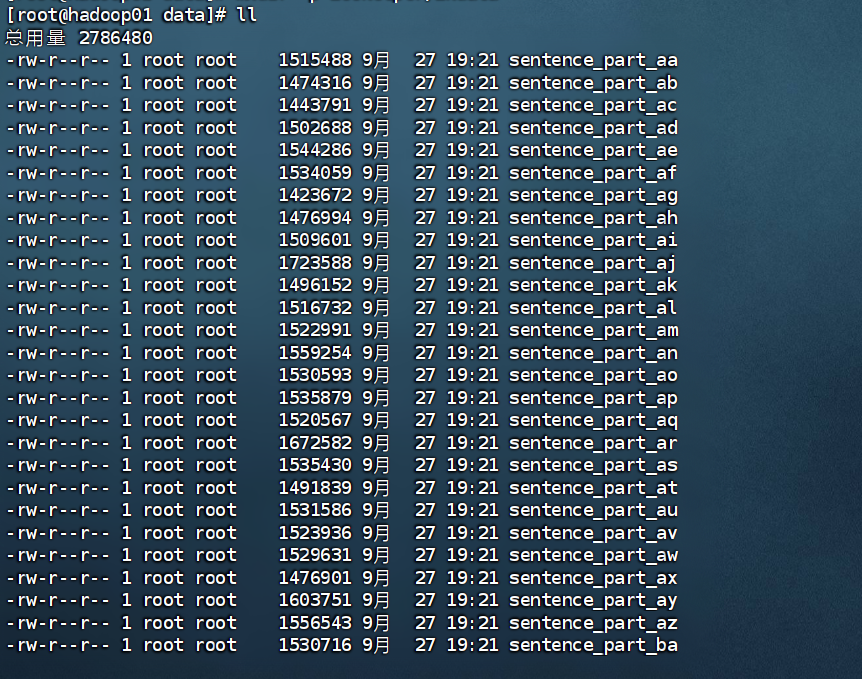
reduce阶段将key相同的键值对进行累加合并，key不变，value值以分号为分割符放在一起，最终写入Hbase。

#### 倒排索引的代码实现：

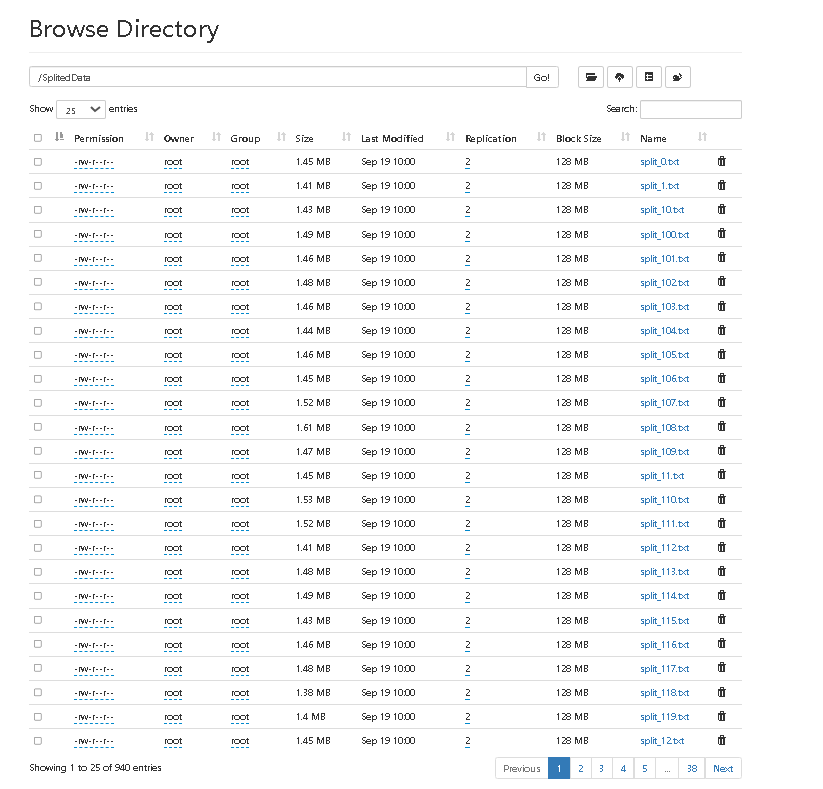
1. //MapReduce.java
2. //java原生类
3. import java.io.IOException;//异常处理
4. import java.util.StringTokenizer;//处理字符串
5. //Hadoop类
6. import org.apache.hadoop.conf.Configuration;//Hadoop配置
7. import org.apache.hadoop.fs.Path;//Hadoop路径
8. import org.apache.hadoop.hbase.HBaseConfiguration;//HBase配置
9. import org.apache.hadoop.hbase.client.Put;//写入HBase
10. import org.apache.hadoop.hbase.io.ImmutableBytesWritable;//HBase用可变字节数组
11. import org.apache.hadoop.hbase.mapreduce.TableOutputFormat;//HBase输出格式类
12. import org.apache.hadoop.hbase.util.Bytes;//HBase字节工具，处理字节数组
13. import org.apache.hadoop.mapreduce.Job;//指向一个MapReduce任务
14. import org.apache.hadoop.mapreduce.Mapper;//Map父类
15. import org.apache.hadoop.mapreduce.Reducer;//Reduce父类
16. import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;//MapReduce输入格式类
17. import org.apache.hadoop.util.GenericOptionsParser;//Hadoop输入参数解析
18. import org.apache.hadoop.mapreduce.lib.input.FileSplit;//MapReduce输入文件切片类
19. import org.apache.hadoop.io.Text;//Hadoop文本类
20. public class MapReduce {
21. //map阶段
22. public static class Map extends Mapper<Object, Text, Text, Text> {//继承自Mapper类，指明四个泛型参数(KEY\_IN,VALUE\_IN,KEY\_OUT,vALUE\_OUT)
24. private Text keyInfo = new Text();//输出键内容
25. private Text valueInfo = new Text();//输出值内容
26. private FileSplit split;//输入文件的切片
27. public void map(Object key, Text value, Context context)
28. throws IOException, InterruptedException {
29. split = (FileSplit) context.getInputSplit();//使用Hadoop提供的FileSplit类获取文件切片
30. // 检查 split 是否为 null
31. if (split == null) {
32. throw new IOException("FileSplit is null. Unable to process the input file.");
33. }
35. StringTokenizer itr = new StringTokenizer(value.toString());
36. String fileName = split.getPath().getName();//获取文件名
37. while (itr.hasMoreTokens()) {//按行map，采用默认分割符分割即可，键为单词+文件名，值为1
38. keyInfo.set(itr.nextToken() + ":" + fileName);
39. valueInfo.set("1");
40. context.write(keyInfo, valueInfo);
41. }
42. }
43. }
44. //combine阶段
45. public static class Combine extends Reducer<Text, Text, Text, Text> {//继承自Reducer类，指明四个泛型参数(KEY\_IN,VALUE\_IN,KEY\_OUT,vALUE\_OUT)
46. private Text info = new Text();//输出值内容
47. public void reduce(Text key, Iterable<Text> values, Context context)
48. throws IOException, InterruptedException {
49. int sum = 0;//计数器
51. //开始循环，遍历所有与键关联的值，累加计数器
52. for (Text value : values) {
53. sum += Integer.parseInt(value.toString());
54. }
55. //拆分键，将单词与文件名分开，单词为键，文件名：计数器为值，写入上下文
56. int splitIndex = key.toString().indexOf(":");//在单词：文件名中定位“：”
57. info.set(key.toString().substring(splitIndex + 1) + ":" + sum);//取key.toString()的文件名子串，与sum拼接
58. key.set(key.toString().substring(0, splitIndex));//取key.toString()的单词子串，作为键
59. context.write(key, info);
60. }
61. }
62. public static class Reduce extends Reducer<Text, Text, ImmutableBytesWritable, Put> {//继承自Reducer类，指明四个泛型参数（KEY\_IN,VALUE\_IN,KEY\_OUT,VALUE\_OUT)
63. public void reduce(Text key,Iterable<Text>values,Context context)
64. throws IOException, InterruptedException {
65. //reduce这一步直接将数据写入HBase建好的表格中
66. //考虑建立一个文件列表fileList，将所有文件名和计数器进行拼接，以“；”分割
67. // 初始化 fileList
68. String fileList = " ";
70. if(values==null){
71. throw new IOException("values is null. Unable to process the input file.");
72. }
73. for (Text value : values) {
74. if(value!=null){
75. fileList += value.toString() + ";";
76. }
77. else{
78. throw new IOException("value is null. Unable to process the input file.");
79. }
80. }
81. //建立一个Put实例，用于写入HBase
82. Put put = new Put(Bytes.toBytes(key.toString()));//写入键值为单词
83. put.addColumn(Bytes.toBytes("fileInfo"), Bytes.toBytes("fileList"), Bytes.toBytes(fileList));//写入新的一列，列族为fileInfo,列名为fileList
84. context.write(new ImmutableBytesWritable(key.getBytes()), put);//将新的键值对写入上下文
85. }
86. }
87. //主方法
88. public static void main(String[] args)
89. throws Exception {
90. Configuration conf = HBaseConfiguration.create();//HBaseConfiguration.create() 方法会加载 HBase 和 Hadoop 的相关配置文件，为之后的任务使用
91. String[] ioArgs = new String[]{"SplitedData", "output"};//给定默认的输入路径和输出路径。这只是一个初始设置，真正的输入输出路径会从用户输入的命令行参数中解析
93. //GenericOptionsParser用于解析传递给程序的命令行参数，并将它们应用到配置 conf 中。
94. //getRemainingArgs()方法返回未被解析的命令行参数，即输入和输出路径。otherArgs 是一个包含实际传入的输入和输出路径的数组
95. String[] otherArgs = new GenericOptionsParser(conf, ioArgs).getRemainingArgs();
97. //验证是否指定自定义输入和输出目录
98. if (otherArgs.length != 2) {
99. System.err.println("Usage: Inverted Index <input path> <output path>");
100. System.exit(2);
101. }
102. //创建一个MapReduce的job实例，进行配置并启动
103. Job job = new Job(conf, "Inverted Index");//配置文件为conf 名称叫“Inverted Index”
104. job.setJarByClass(MapReduce.class);//设置当前类来打包jar文件
105. job.setMapperClass(Map.class);//map类
106. job.setCombinerClass(Combine.class);//combine类
107. job.setReducerClass(Reduce.class);//reduce类
108. job.setMapOutputKeyClass(Text.class);//设置map阶段输出键类
109. job.setMapOutputValueClass(Text.class);//设置map阶段输出值类
111. job.setOutputKeyClass(ImmutableBytesWritable.class);//设置reduce后输出键类
112. job.setOutputValueClass(Put.class);//设置reduce后输出值类
113. //向hbase中写入
114. job.setOutputFormatClass(TableOutputFormat.class);//指定输出格式为HBase表格格式，表示输出将写入HBase
115. job.getConfiguration().set(TableOutputFormat.OUTPUT\_TABLE, "InvertedIndexTable");//指定输出表名为“InvertedIndexTable”
116. FileInputFormat.addInputPath(job, new Path(otherArgs[0]));//设置输入路径为用户输入的第一个参数
118. System.exit(job.waitForCompletion(true) ? 0 : 1);
120. }
121. }

## 运行结果与分析

### 分割源txt文件

* 

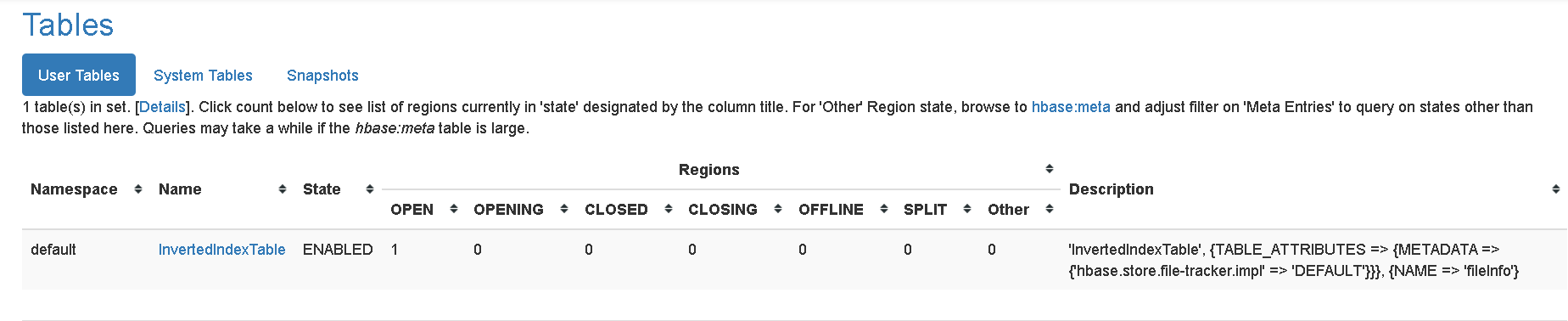
### 上传文件

* 

### 在HBase中建立空InvertedIndexTable

1. create 'InvertedIndexTable', 'fileInfo'

使用HBase Web UI查看建立好的表格：

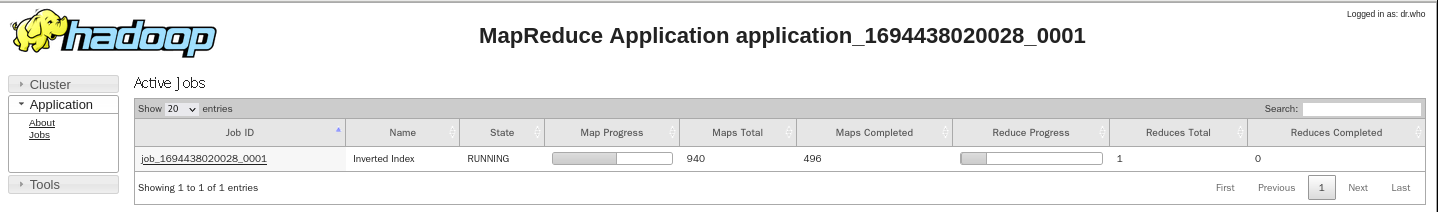


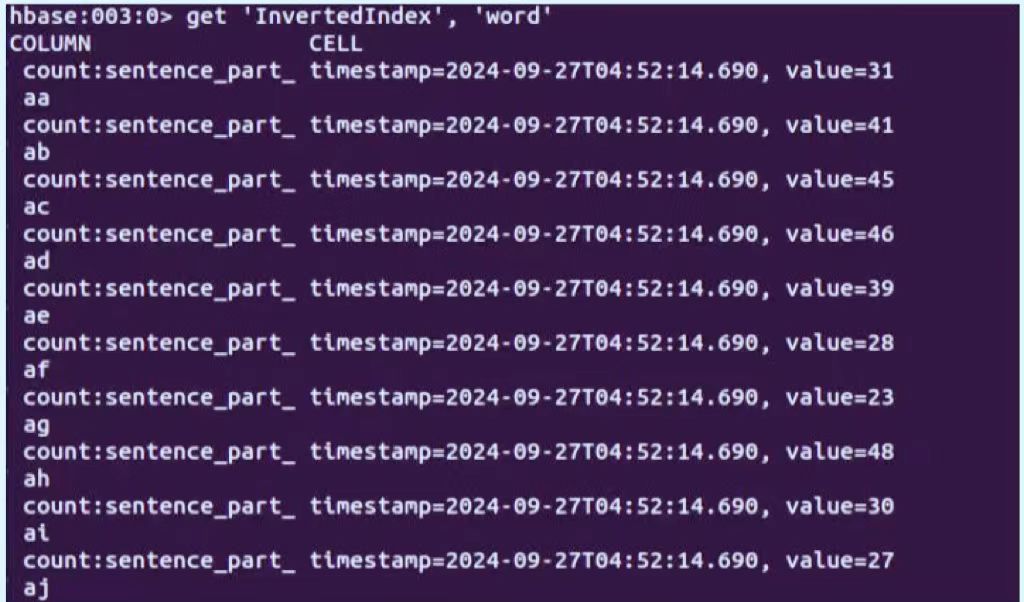
### 运行倒排索引代码

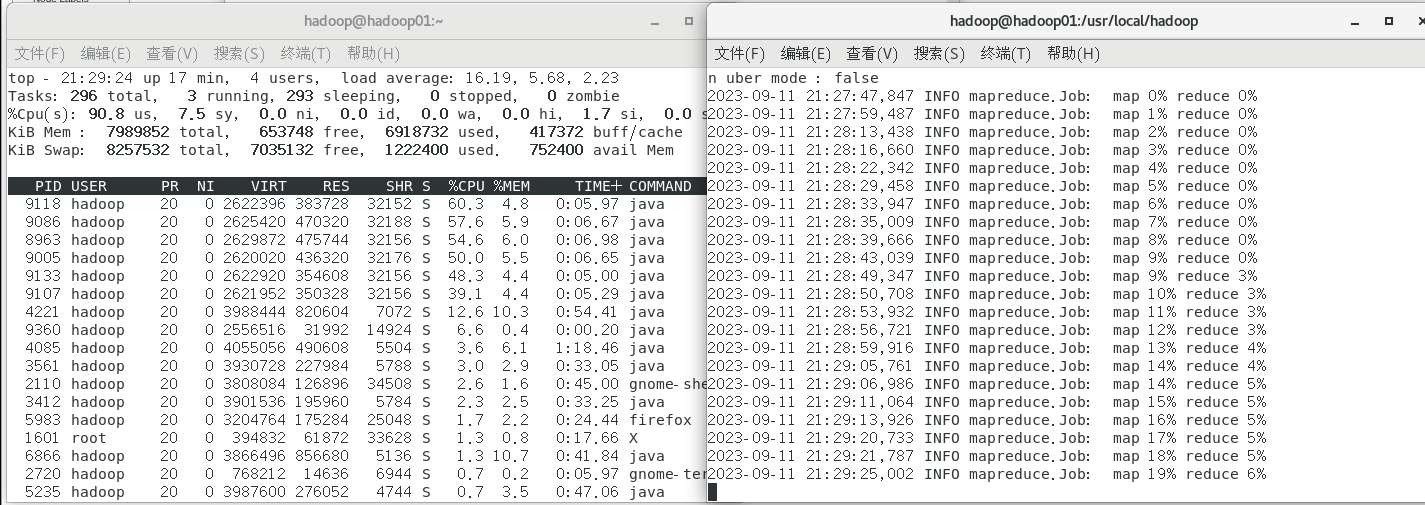
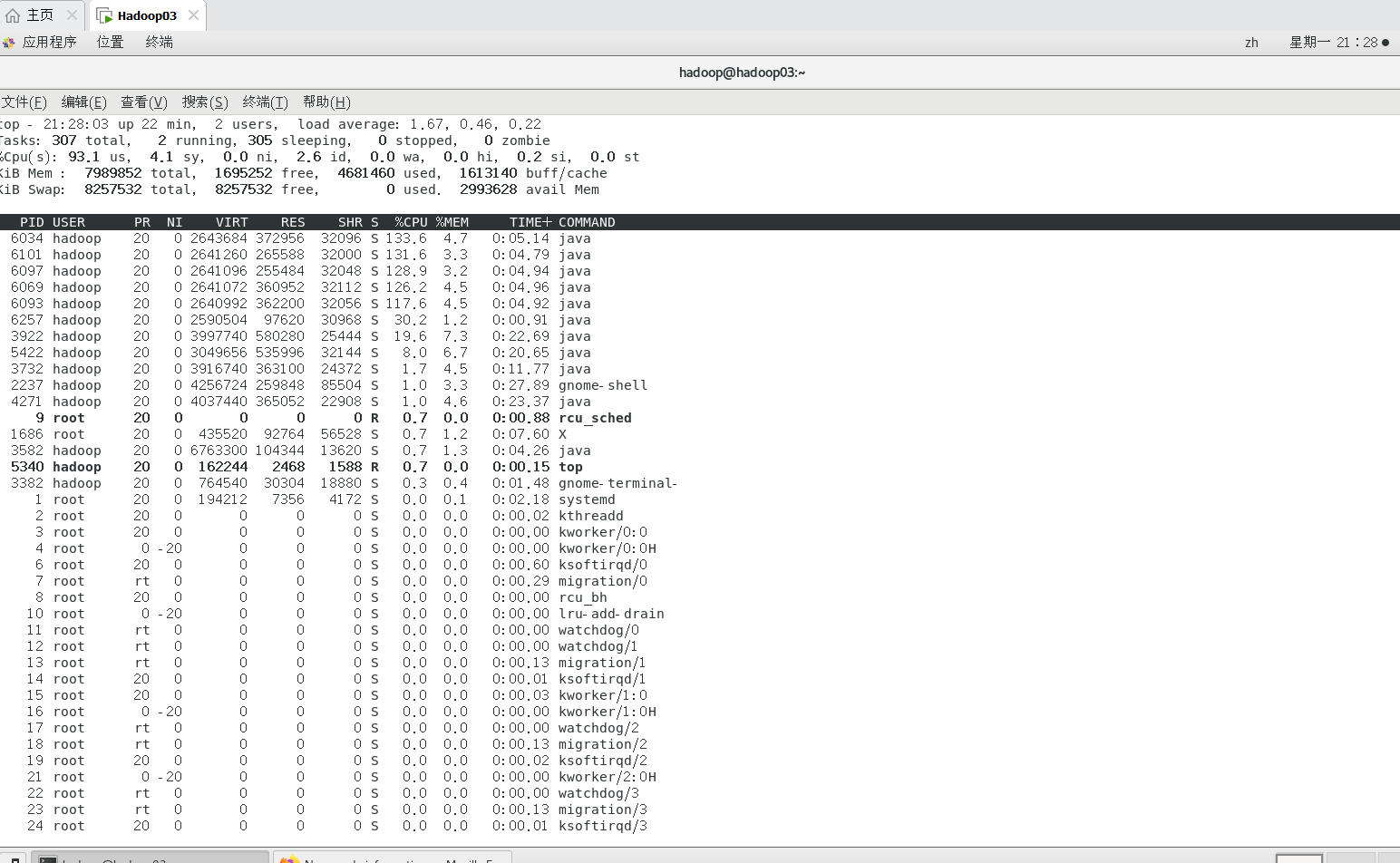
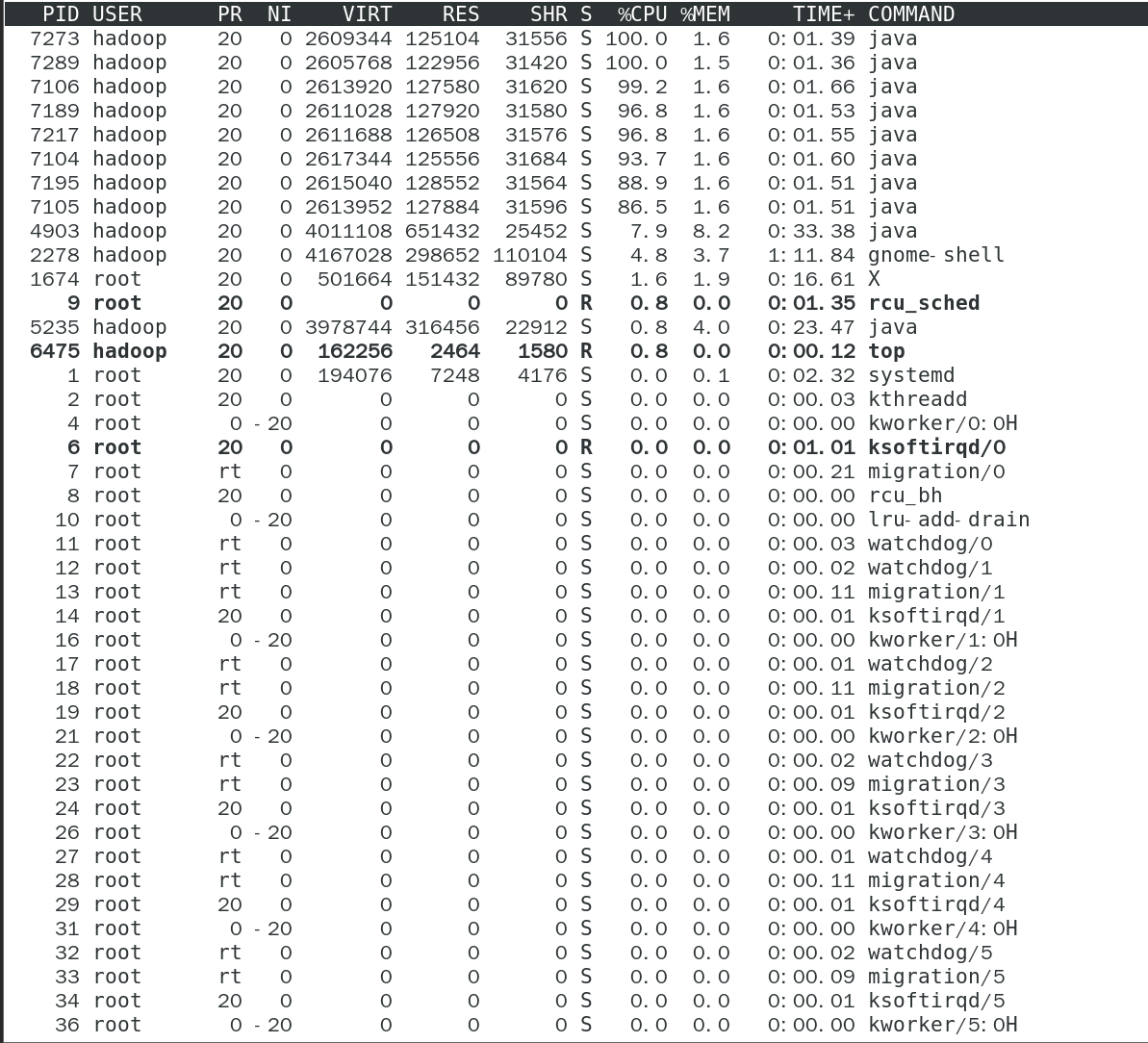
我们将MapReduce.java源文件打包成jar文件运行。



hadoop jar ./MapReduce/MapReduceJob.jar MapReduce



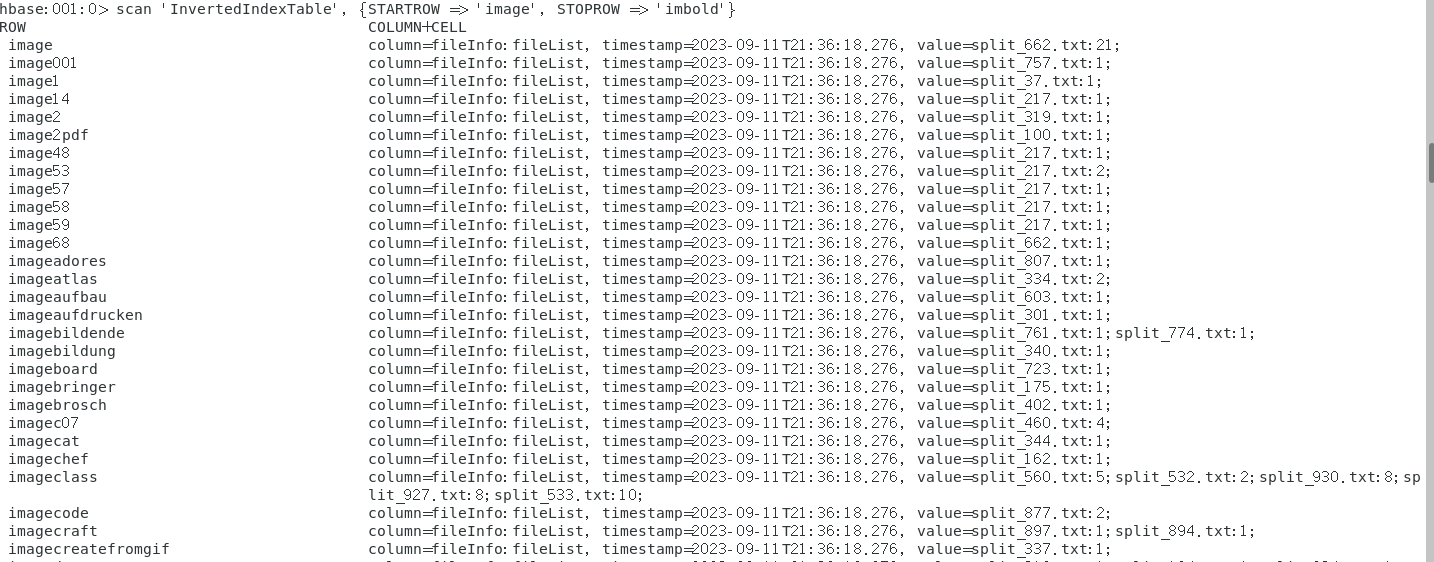


* 节点Hadoop01运行倒排索引的过程进度：
* 节点Hadoop02运行倒排索引的过程：
* 
* 节点Hadoop03运行倒排索引的过程：
* 
* 可以看出三台虚拟机都在运行MapReduce

### 查看结果

由于本次实验是按照文件划分的，所以900万多的句子每一行都会有序号，导致hbase中很大一部分是 “行号-文件名”形式。为了体现出倒排索引，运行以下的代码，将单词image

到单词'imbold'的倒排索引展示出来。

scan 'InvertedIndexTable', {STARTROW => 'image', STOPROW => 'imbold'}

可以看出确实实现了倒排索引。

# 心得体会

刘秉致：通过本次实验，我初步认识和了解到了大数据系统开发的基本知识，尤其深入的了解了Hadoop软件框架的运作原理和HBase数据库的基本应用。本次实验中，出现了诸多问题，环境配置无效、缺少依赖文件的情况时有发生，但通过我自己的查询和学习，一点点克服了困难，实现了实验，这次试验培养了我的自我解决问题和查询信息的能力，更让我对计算机网络和分布式原理有了一定认识。

史立彬：本次实验让我将课上学到的大数据系统开发的基础知识运用于实践之中，使我对Hadoop和HBase的架构有了更深入的认识。我在实验中遇到了诸多问题，尤其是配置环境的问题耗费了我较长的时间。我通过csdn上的经验以及chatgpt给出的回答成功地解决了这些问题，这锻炼了我提出问题和查阅问题的能力，对我今后的学习有一定的益处。

陈依凡：通过本次实验，我对大数据系统开发有了更为深入的理解和实践体验，特别是在 Hadoop 框架的工作机制和 HBase 数据库的实际应用上，我积累了宝贵的经验。在实验过程中，虽然遇到了不少挑战，诸如环境配置失败、依赖文件缺失等问题，但这些困难也为我提供了提升自我能力的机会。通过不断查阅文档、论坛资料以及其他技术社区的讨论，我逐步解决了这些难题。此次实验不仅强化了我解决问题的能力，还让我对计算机系统的分布式架构和网络通信有了更加全面的理解。

董顺心：通过本次实验，我对大数据系统开发的基本知识有了更深入的认识，特别是在Hadoop框架和HBase数据库的运作方面。在实验过程中遇到了许多问题，如环境配置报错，配置文件存在问题，Java无法运行等，这些问题让我花费了不少时间去解决。然而，通过查询技术社区的相关资料寻求帮助，我逐一克服了这些困难。这次实验不仅提升了我解决问题的能力，还加深了我对计算机网络和分布式原理的理解。为分布式系统的开发工作积累了实践经验。