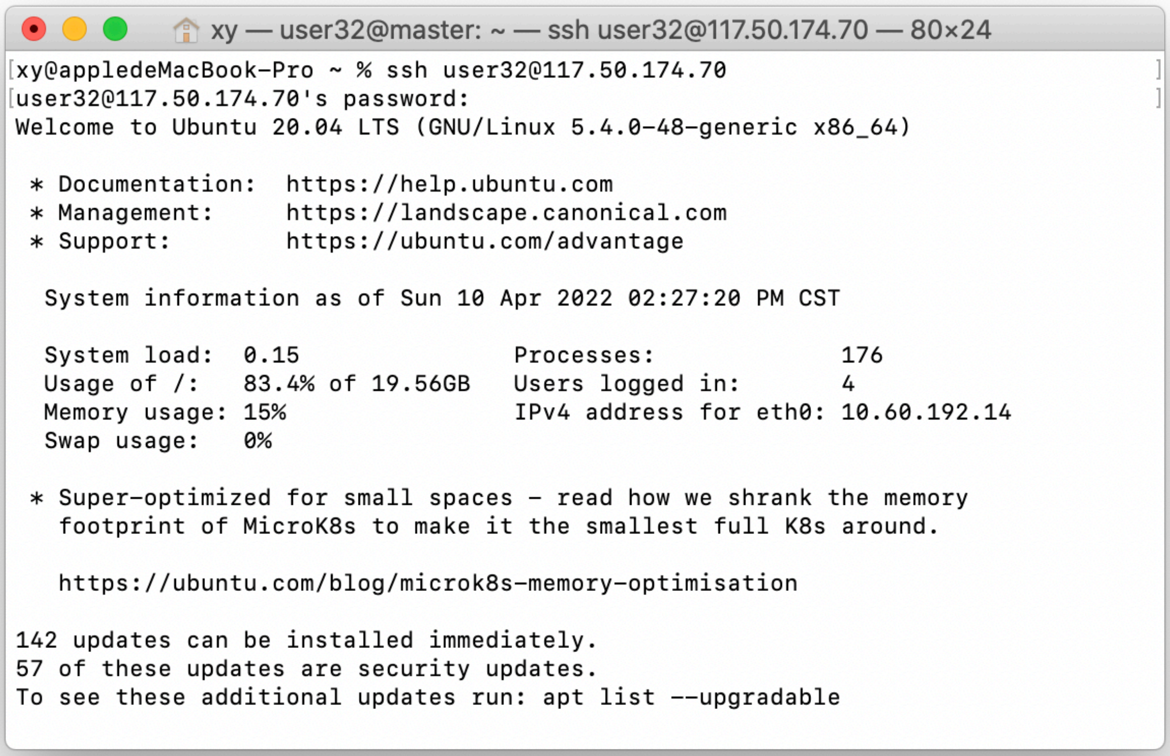
云计算-2051495-徐奕-作业4

零、登陆集群

> ssh user32@117.50.174.70



xy — user32@master: ~ — ssh user32@117.50.174.70 — 80×24

```
xy@appledeMacBook-Pro ~ % ssh user32@117.50.174.70
user32@117.50.174.70's password:
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-48-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun 10 Apr 2022 02:27:20 PM CST

  System load:  0.15               Processes:             176
  Usage of /:   83.4% of 19.56GB   Users logged in:       4
  Memory usage: 15%                IPv4 address for eth0: 10.60.192.14
  Swap usage:   0%

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

142 updates can be installed immediately.
57 of these updates are security updates.
To see these additional updates run: apt list --upgradable
```

一、倒排索引

- 第一题：倒排索引（50分）

　　倒排索引是构造搜索引擎的核心部件。它是一个纯文本的HDFS大数据文件。里面记录着每个索引词出现的位置：所在的文件，偏移量是多少。

　　示例程序 扫描input目录中的每个输入文件：

　　　　　　提取word，记录word在这个文件中出现的次数。

把<word，文件名，出现次数>写入HDFS文件。同一个word，出现在多少个文件中，倒排索引就有多少条记录。

要求：修改示例程序，生成的倒排索引文件中，每个索引词一行，格式：
索引词，tab键，文件1：文件1中出现的次数；文件2：文件2中出现的次数……

file1

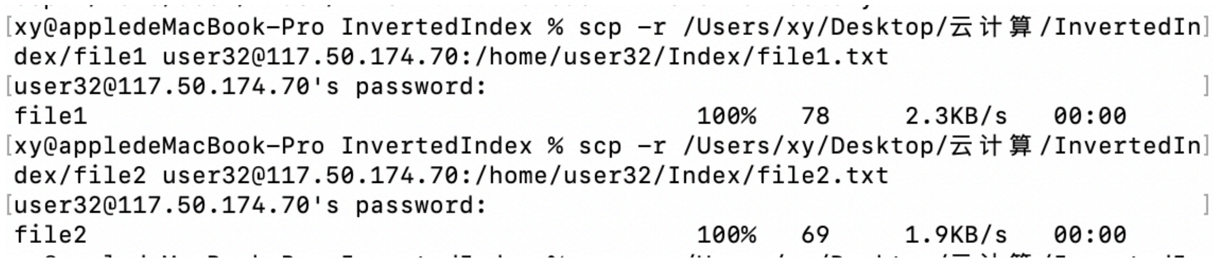
```
Hello World Bye World
Hello World Bye World
Hello World Bye World
Hello World
```

file2

```
Hello Hadoop Goodbye Hadoop
Hello Hadoop Goodbye Hadoop
Hello Hadoop
```

导入云主机

> location:/home/user32/Index



```
[xy@appledeMacBook-Pro InvertedIndex % scp -r /Users/xy/Desktop/云计算/InvertedIn]
dex/file1 user32@117.50.174.70:/home/user32/Index/file1.txt
[user32@117.50.174.70's password:                                              ]
file1                                         100%   78     2.3KB/s   00:00
[xy@appledeMacBook-Pro InvertedIndex % scp -r /Users/xy/Desktop/云计算/InvertedIn]
dex/file2 user32@117.50.174.70:/home/user32/Index/file2.txt
[user32@117.50.174.70's password:                                              ]
file2                                         100%   69     1.9KB/s   00:00
```

class InvertedIndex主要构成：

Mapper

```
/*1.实现Mapper端的逻辑
   * KEYIN：文件中读取的偏移量-->LongWritable(固定的)
   * VALUEIN：文件中实际读取的内容-->Text
   * KEYOUT:Mapper处理完成后传递给Reducer中的KEYIN的数据类型-->不固定，根据需求来
   * VALUEOUT：Mapper端处理完成后传递给Reducer中的VALUEIN的数据类型-->不固定，根据需求来
   */

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, Text>{
    //修改一：统一map和reduce的key-value类型
    private final static Text one = new Text();
    private Text word = new Text();
    private String fileName = null;

    /*
     * 需要实现Mapper端的处理逻辑
     * 将每个词拆开，key设为文件名加词，value设成空值。
     *  好处：可以在reduce之前先进行一次计数，减少后续数据处理压力
     * key:是文件中数据的偏移量，数据类型是由泛型中定义得来的KEYIN
     * value:是文件中实际的内容，数据类型是泛型中定义得来的VALUEIN
     * context：将处理过后产生的KV，写成文件输出
     */

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {

      if(fileName == null)
      {
         FileSplit fileSplit = (FileSplit)context.getInputSplit();
         fileName = fileSplit.getPath().getName();
      }
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken() + '\t' + fileName);
        context.write(word, one);
      }

    }
  }
```

Reducer

IntSumReducer

```
/*
   * Reducer相当于对Mapper端处理过后的数据进行一个实际的处理业务,在此处统计拥有同一个key的键值对个数，然后统一形式输出
     (此题：<词汇/t文件名,one> -> <词汇,文件名：个数>)
   * KEYIN-->Mapper处理过后输出key的数据类型，由Mapper的泛型中第三个参数决定
   * VALUE-->Mapper处理过后输出value的数据类型，由Mapper的泛型中第四个参数决定
   * KEYOUT-->Reducer端处理完数据之后要写出key的数据类
      (此题：词)
   * VALUEOUT-->Reducer处理完数据之后，要写出value的 数据类
      (此题：文件名和个数)
   */

  public static class IntSumReducer
       extends Reducer<Text,Text,Text,Text> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<Text> values,
                       Context context
                       ) throws IOException, InterruptedException {
     String keyString =key.toString();
     String[] keySplited =keyString.split("\t");
```

```
      //把词汇和文件名用'/t'分开
      int sum = 0;
      for (Text val : values) {
        sum += 1;
      }
      result.set(sum);
      String resultString=keySplited[1]+':'+result;
      context.write(new Text(keySplited[0]) ,new Text(resultString));
    }
  }
```

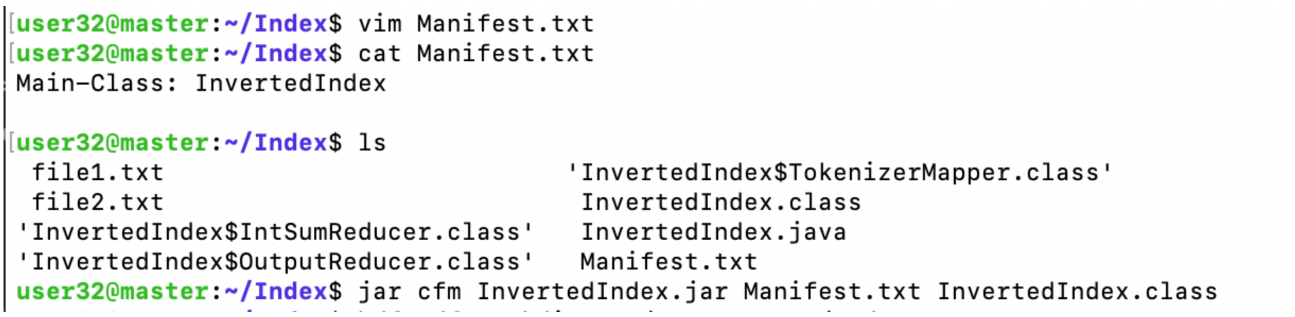
OutputReducer

```
 //对相同key的键值对 合并value
  public static class OutputReducer
       extends Reducer<Text,Text,Text,Text> {
    public void reduce(Text key, Iterable<Text> values,
                       Context context
                       ) throws IOException, InterruptedException {
      String result = new String();
      boolean isFirst = true;
      for (Text val : values)
      {
          if(isFirst==false)
            result = result+';';
          isFirst = false;
          result = result + val.toString();
      }
        context.write(key, new Text(result));
    }
  }
```

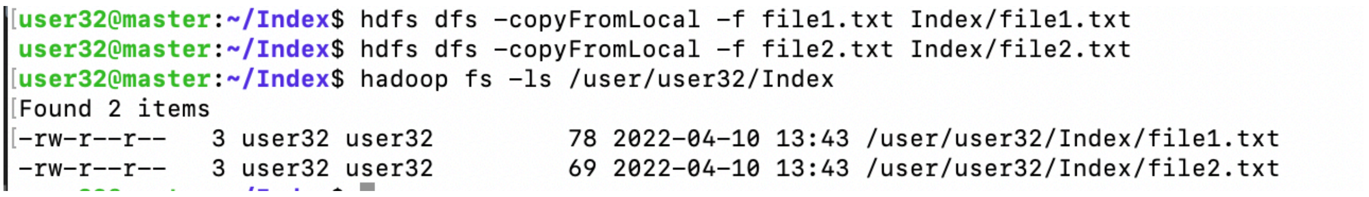
Main

```
public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
    if (otherArgs.length < 2) {
      System.err.println("Usage: wordcount <in> [<in>...] <out>");
      System.exit(2);
    }
    Job job = Job.getInstance(conf, "word count");

    job.setJarByClass(InvertedIndex.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(OutputReducer.class);
    job.setMapOutputKeyClass(Text.class);
    job.setMapOutputValueClass(Text.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    for (int i = 0; i < otherArgs.length - 1; ++i) {
      FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
    }
    FileOutputFormat.setOutputPath(job,
      new Path(otherArgs[otherArgs.length - 1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
}////最终实现倒序索引
```

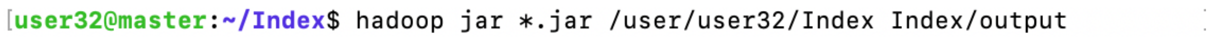
编译运行



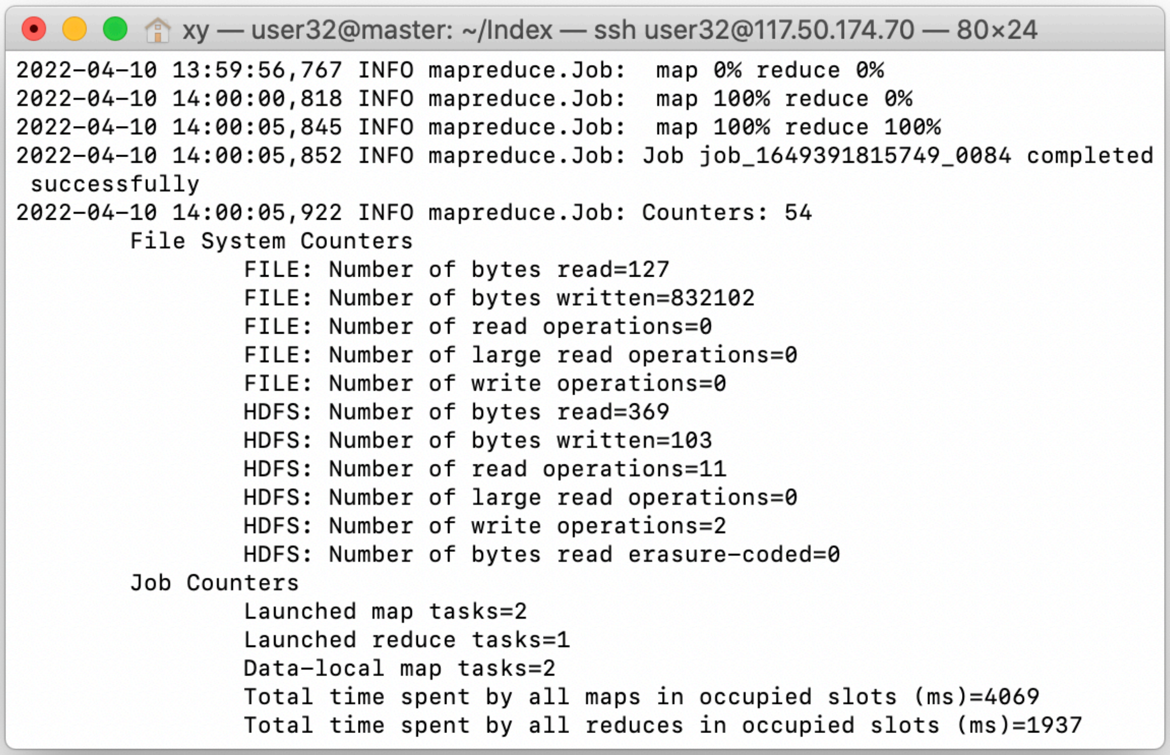
```
user32@master:~/Index$ vim Manifest.txt
user32@master:~/Index$ cat Manifest.txt
Main-Class: InvertedIndex

user32@master:~/Index$ ls
 file1.txt                            'InvertedIndex$TokenizerMapper.class'
 file2.txt                             InvertedIndex.class
'InvertedIndex$IntSumReducer.class'    InvertedIndex.java
'InvertedIndex$OutputReducer.class'    Manifest.txt
user32@master:~/Index$ jar cfm InvertedIndex.jar Manifest.txt InvertedIndex.class
```



```
user32@master:~/Index$ hdfs dfs -copyFromLocal -f file1.txt Index/file1.txt
user32@master:~/Index$ hdfs dfs -copyFromLocal -f file2.txt Index/file2.txt
user32@master:~/Index$ hadoop fs -ls /user/user32/Index
Found 2 items
-rw-r--r--   3 user32 user32         78 2022-04-10 13:43 /user/user32/Index/file1.txt
-rw-r--r--   3 user32 user32         69 2022-04-10 13:43 /user/user32/Index/file2.txt
```



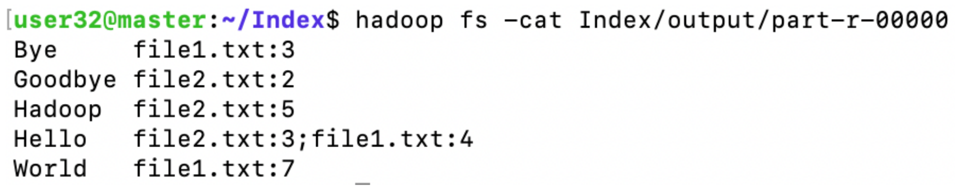
```
user32@master:~/Index$ hadoop jar *.jar /user/user32/Index Index/output
```



xy — user32@master: ~/Index — ssh user32@117.50.174.70 — 80×24

```
2022-04-10 13:59:56,767 INFO mapreduce.Job:  map 0% reduce 0%
2022-04-10 14:00:00,818 INFO mapreduce.Job:  map 100% reduce 0%
2022-04-10 14:00:05,845 INFO mapreduce.Job:  map 100% reduce 100%
2022-04-10 14:00:05,852 INFO mapreduce.Job: Job job_1649391815749_0084 completed
 successfully
2022-04-10 14:00:05,922 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=127
                FILE: Number of bytes written=832102
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=369
                HDFS: Number of bytes written=103
                HDFS: Number of read operations=11
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=2
                Launched reduce tasks=1
                Data-local map tasks=2
                Total time spent by all maps in occupied slots (ms)=4069
                Total time spent by all reduces in occupied slots (ms)=1937
```

查看结果



```
user32@master:~/Index$ hadoop fs -cat Index/output/part-r-00000
Bye     file1.txt:3
Goodbye file2.txt:2
Hadoop  file2.txt:5
Hello   file2.txt:3;file1.txt:4
World   file1.txt:7
```

二、 大数组排序

• 第2题：编程排序256M个整数组成的大数组（50分）。

要求，外部编程，随机生成256M个整数。传入HDFS文件系统。编写MapReduce程序，对数组进行排序。

提示：

每个reduce任务生成一个输出文件。这个输出文件，按key排序。

如果，系统中有多个reduce任务。简单合并输出文件，里面的记录是没有按照key排序的。需要再编一个MapReduce程序。

总之，想要排序的结果，最后一级reduce，只能有一个reduce任务。

注意点：在map阶段需要将输入的文本转换成整数

256mIntArray.txt

大数组数据txt location

> /common/Big_Array/256mIntArray.txt

SortMapper.java

```java
package Sort;

import java.io.IOException;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class SortMapper extends MapReduceBase implements Mapper<LongWritable,
Text,IntWritable, NullWritable> {
        private static IntWritable data= new IntWritable();
        public void map(LongWritable key, Text value, OutputCollector<IntWritable,
NullWritable> output, Reporter reporter) throws IOException {
//String[] SingleCountryData = valueString.split(",");
//output.collect(new Text(SingleCountryData[7]), one);
//if(SingleCountryData[7].equals("United States"))
                String valueString = value.toString();
                data.set(Integer.parseInt(valueString));
                output.collect(data, NullWritable.get());
        }
}
```

SortReducer.java

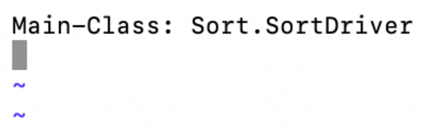
```java
package Sort;
import java.io.IOException;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import java.util.*;
import org.apache.hadoop.mapred.*;
public class SortReducer extends MapReduceBase implements Reducer<IntWritable,
NullWritable,IntWritable, NullWritable> {
        public void reduce(IntWritable t_key, Iterator<NullWritable> values,
OutputCollector<IntWritable,NullWritable> output, Reporter reporter) throws IOException
{
                //Text key = t_key;
                //int frequencyForCountry = 0;
                while(values.hasNext())
                {
                        NullWritable val = values.next();
                        output.collect(t_key,NullWritable.get());
                }
        }
}
```

> mapreduce中的reduce阶段会对key自动排序，但是当数组中有多个相同的数时，他们会共享同一个key。需要在reduce阶段对这个key重复输出。输出次数等于value表的长度。

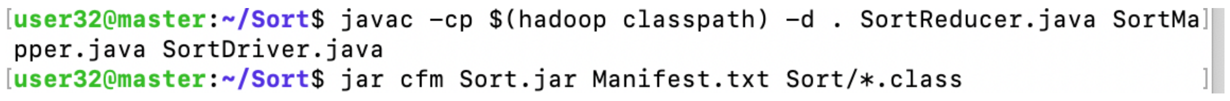
SortDriver.java

```
package Sort;

                import org.apache.hadoop.fs.Path;
                import org.apache.hadoop.io.*;
                import org.apache.hadoop.mapred.*;

public class SortDriver {
        public static void main(String[] args) {
                JobClient my_client = new JobClient();
                // Create a configuration object for the job
                JobConf job_conf = new JobConf(SortDriver.class);

                // Set a name of the Job
                job_conf.setJobName("XySortDemo");

                // Specify data type of output key and value
                job_conf.setOutputKeyClass(IntWritable.class);
                job_conf.setOutputValueClass(NullWritable.class);

                // Specify names of Mapper and Reducer Class
                job_conf.setMapperClass(Sort.SortMapper.class);
                job_conf.setReducerClass(Sort.SortReducer.class);
               // Specify formats of the data type of Input and output
                job_conf.setInputFormat(TextInputFormat.class);
                job_conf.setOutputFormat(TextOutputFormat.class);
                FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
                FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

                my_client.setConf(job_conf);
                try {
                        // Run the job
                        JobClient.runJob(job_conf);
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
        }
}
```

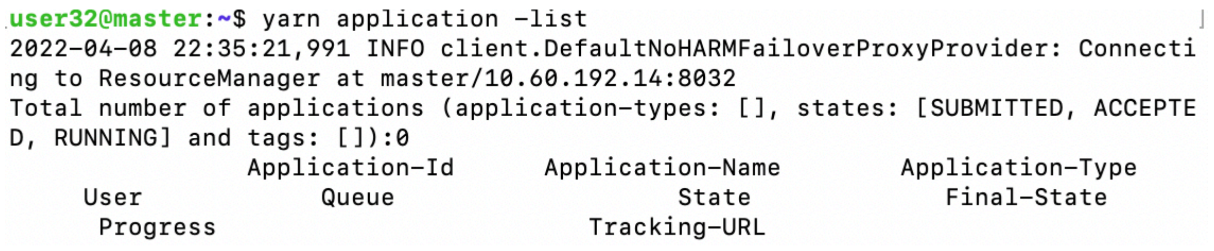
Manifest.txt



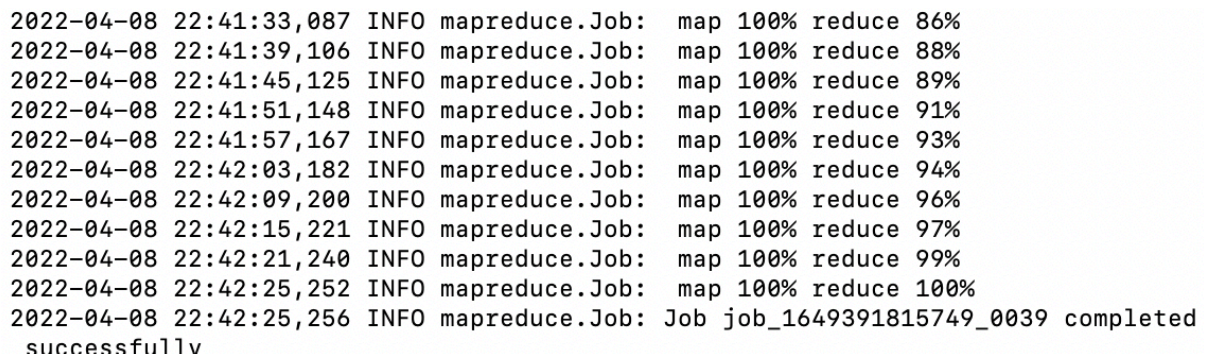
```
Main-Class: Sort.SortDriver

~
~
```



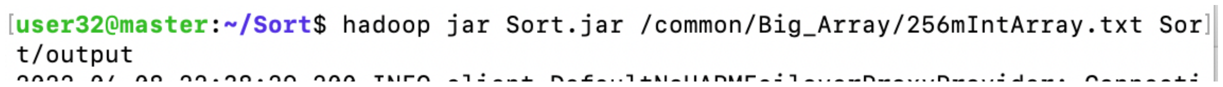
```
user32@master:~/Sort$ javac -cp $(hadoop classpath) -d . SortReducer.java SortMa
pper.java SortDriver.java
user32@master:~/Sort$ jar cfm Sort.jar Manifest.txt Sort/*.class
```



```
user32@master:~$ yarn application -list
2022-04-08 22:35:21,991 INFO client.DefaultNoHARMFailoverProxyProvider: Connecti
ng to ResourceManager at master/10.60.192.14:8032
Total number of applications (application-types: [], states: [SUBMITTED, ACCEPTE
D, RUNNING] and tags: []):0
                Application-Id      Application-Name        Application-Type
     User           Queue                   State             Final-State
      Progress                        Tracking-URL
```



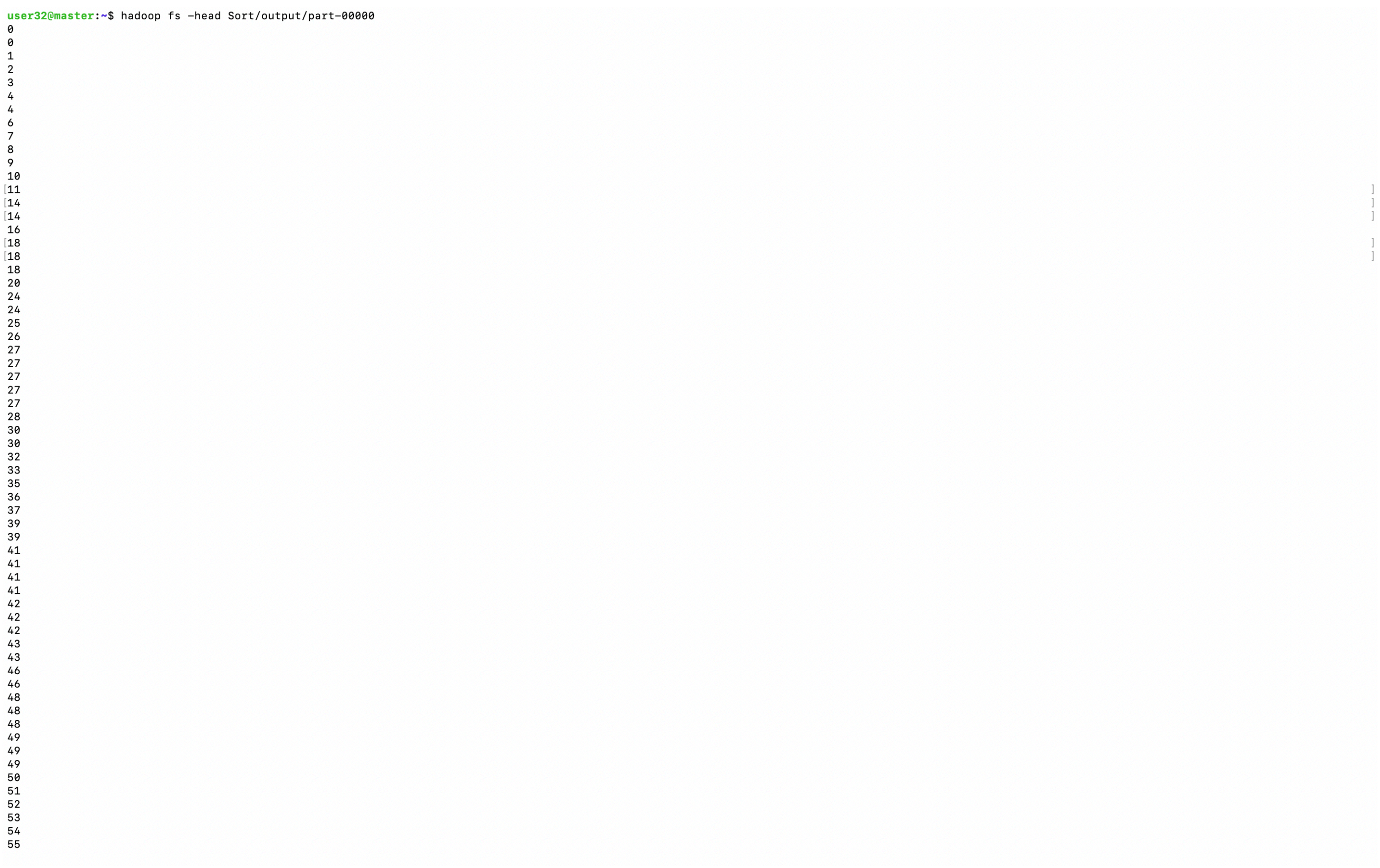
```
2022-04-08 22:41:33,087 INFO mapreduce.Job:  map 100% reduce 86%
2022-04-08 22:41:39,106 INFO mapreduce.Job:  map 100% reduce 88%
2022-04-08 22:41:45,125 INFO mapreduce.Job:  map 100% reduce 89%
2022-04-08 22:41:51,148 INFO mapreduce.Job:  map 100% reduce 91%
2022-04-08 22:41:57,167 INFO mapreduce.Job:  map 100% reduce 93%
2022-04-08 22:42:03,182 INFO mapreduce.Job:  map 100% reduce 94%
2022-04-08 22:42:09,200 INFO mapreduce.Job:  map 100% reduce 96%
2022-04-08 22:42:15,221 INFO mapreduce.Job:  map 100% reduce 97%
2022-04-08 22:42:21,240 INFO mapreduce.Job:  map 100% reduce 99%
2022-04-08 22:42:25,252 INFO mapreduce.Job:  map 100% reduce 100%
2022-04-08 22:42:25,256 INFO mapreduce.Job: Job job_1649391815749_0039 completed
 successfully
```



```
user32@master:~/Sort$ hadoop jar Sort.jar /common/Big_Array/256mIntArray.txt Sor
t/output
2022-04-08 22:38:29,200 INFO client.DefaultNoHARMFailoverProxyProvider: Connecti
```

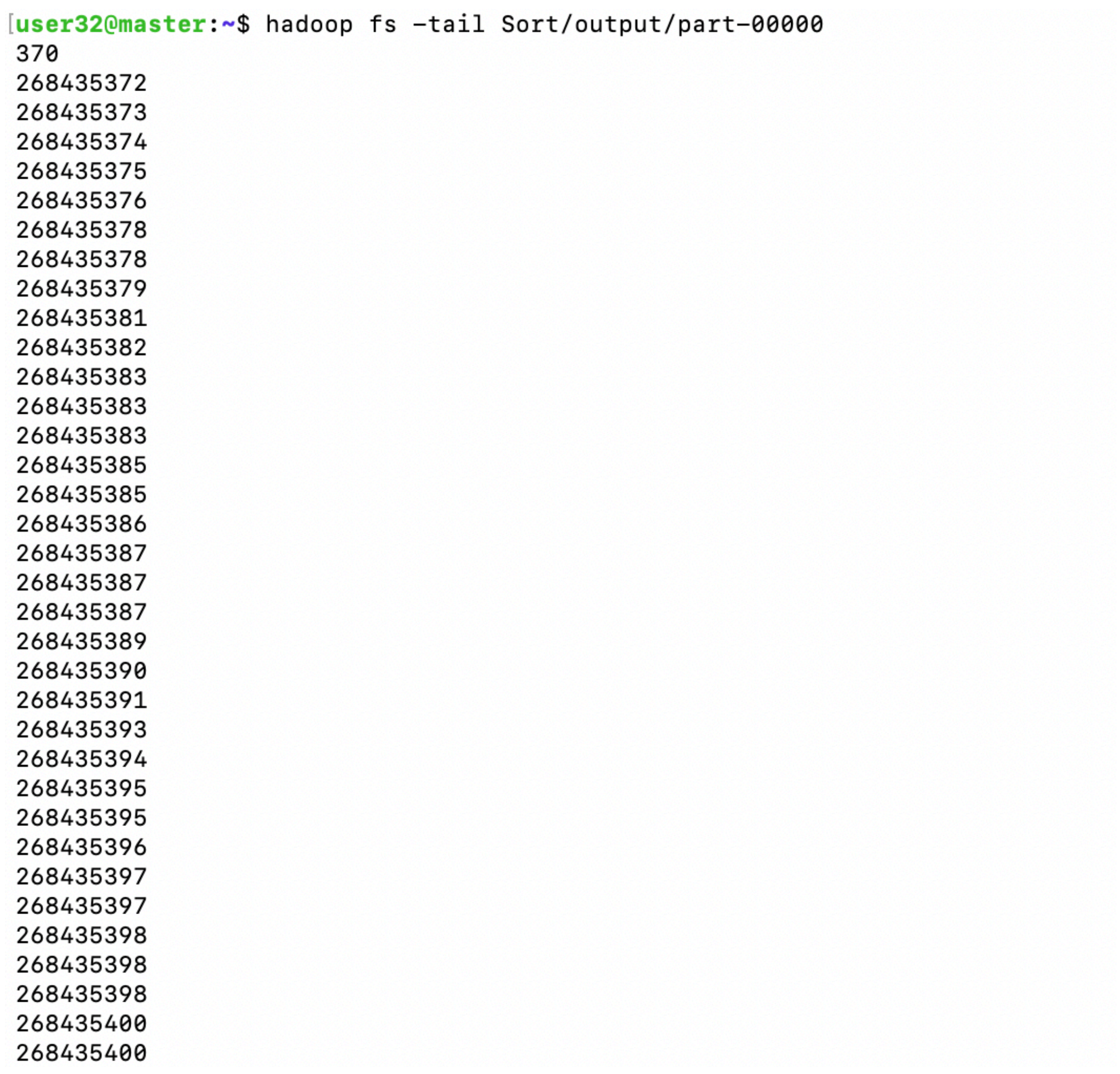
查看开头

> $hadoop fs -head path



```
user32@master:~$ hadoop fs -head Sort/output/part-00000
0
0
1
2
3
4
4
6
7
8
9
10
11
14
14
16
18
18
18
20
24
24
25
26
27
27
27
27
27
28
30
30
32
33
35
36
37
39
39
41
41
41
41
42
42
42
43
43
46
46
48
48
48
49
49
49
50
51
52
53
54
55
```

查看结尾



```
[user32@master:~$ hadoop fs -tail Sort/output/part-00000
370
268435372
268435373
268435374
268435375
268435376
268435378
268435378
268435379
268435381
268435382
268435383
268435383
268435383
268435385
268435385
268435386
268435387
268435387
268435387
268435389
268435390
268435391
268435393
268435394
268435395
268435395
268435396
268435397
268435397
268435398
268435398
268435398
268435400
268435400
```

> 出现370可能因为tail是按字节算不是按行算的

由此看出结果有序