Program 3 Report

Yangxiao Wang

Contents

1 Documentation

In this project, we are exploring the parallel technique - MapReduce. We used Inversed indexing as an example. My approach is straightforward, just map all the files with the same key (parameter) to the same reducer. And in the reducer, count the total number of the occurrence of the key. This implementation does not require combiner. When using combiner the output will have a extra 1 behind every file count (value). For example:

```
    HDLC rfc2865.txt 22 1 rfc1122.txt 23 1
```

My opinion is that the reducer will run twice when combiner is set.

In addition, my program also sorts the documents result by its count. For every keyword, the result will be sorted by the count of the keyword in the document.

2 Source code

2.1 InvertedIndexing.java

```java
import org.apache.hadoop.conf.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;

import java.io.IOException;
import java.util.*;


public class InvertedIndexing
{

    public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, ↩
        Text, Text>
    {

        JobConf conf;

        public void configure(JobConf job)
        {
            this.conf = job;
        }

        public void map(LongWritable key, Text value, OutputCollector<Text, Text> output, ↩
        Reporter reporter) throws IOException
        {

            // retrieve # keywords from JobConf
            int argc = Integer.parseInt(conf.get("argc"));
            // put args into a String array
            Set<String> args = new HashSet();
            // retrieve keywords
            for (int i = 0; i < argc; i++)
            {
                args.add(conf.get("keyword" + i));
            }
            // get the current file name
            FileSplit fileSplit = (FileSplit) reporter.getInputSplit();
            String filename = "" + fileSplit.getPath().getName();
            String lines = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(lines);
            //collect if next token match one of the args
            while (tokenizer.hasMoreTokens())
            {
                String x = tokenizer.nextToken();
                if (args.contains(x))
                {
                    output.collect(new Text(x), new Text(filename));
                }
            }
        }
    }

    public static class Reduce extends MapReduceBase implements Reducer<Text, Text, Text, ↩
        Text>
    {

        public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text> ↩
        output, Reporter reporter) throws IOException
        {
            HashMap<String, Integer> hm = new HashMap<String, Integer>();
            //Count the occurrence number of key in each file
            while (values.hasNext())
            {
                String name = values.next().toString();
                if (hm.containsKey(name))
                {
                    hm.put(name, hm.get(name) + 1);
                }
                else
                {
                    hm.put(name, 1);
                }
            }
            //create Comparator to sort the result by count number
            Comparator<java.util.Map.Entry<String, Integer>> valueComparator =
            new Comparator<java.util.Map.Entry<String, Integer>>()
            {
                @Override
                public int compare(java.util.Map.Entry<String, Integer> e1, java.util.Map.↩
        Entry<String, Integer> e2)
                {
                    return e1.getValue() - e2.getValue();
                }
            };

            //
            sort the result
            List<java.util.Map.Entry<String, Integer>> listDoc =
            new ArrayList<java.util.Map.Entry<String, Integer>>(hm.entrySet());
            Collections.sort(listDoc, valueComparator);

            //create output string
            StringBuilder sb = new StringBuilder();
            for (java.util.Map.Entry<String, Integer> e : listDoc)
            {
                sb.append(e.getKey());
                sb.append(" ");
                sb.append(e.getValue());
                sb.append(" ");
            }

            //output
            Text docListC = new Text(sb.toString());
            output.collect(key, docListC);
        }
    }

    public static void main(String[] args) throws Exception
    {
        long time = System.currentTimeMillis();
        JobConf conf = new JobConf(InvertedIndexing.class);
        conf.setJobName("invertInd");

        conf.setOutputKeyClass(Text.class);
        conf.setOutputValueClass(Text.class);

        conf.setMapperClass(Map.class);
        //no need to combine because reducer already taken care of it
        //conf.setCombinerClass(Reduce.class);
        conf.setReducerClass(Reduce.class);

        conf.setInputFormat(TextInputFormat.class);
        conf.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(conf, new Path(args[1]));

        // argc maintains #keywords
        conf.set("argc", String.valueOf(args.length - 2));
        for (int i = 0; i < args.length - 2; i++)
        {
            conf.set("keyword" + i, args[i + 2]);
        }

        JobClient.runJob(conf);
        System.out.println("Elapsed time = " + (System.currentTimeMillis() - time) + " ms")↩
        ;
    }
}
```

3 Execution output

3.1 Output analysis

- The performance of sequential version: 253.143 s
- The performance of parallel version: 70.517 s
- Improvement: 253.143 / 70.517 = 3.5898 times
- Cannot use diff to compare those two output because the order of the files with the same number of count is basically random. However, by comparing the result of the first two parameter HDLC and LAN and some large item with large count number, it is safe to say the outputs are the same.

3.2 Output of sequential version

```
HDLC     rfc2865.txt 1 rfc1122.txt 1 rfc891.txt 2 rfc907.txt 2 rfc2863.txt 3 rfc1662.↩
    txt 4
LAN      rfc2613.txt 1 rfc1044.txt 1 rfc4862.txt 1 rfc1123.txt 1 rfc2348.txt 1 rfc3461.↩
    txt 1 rfc1661.txt 1 rfc1155.txt 2 rfc5321.txt 2 rfc2115.txt 2 rfc1629.txt 3 rfc1559↩
    .txt 3 rfc1724.txt 3 rfc2895.txt 4 rfc1660.txt 5 rfc1213.txt 5 rfc1659.txt 5 ↩
    rfc1658.txt 5 rfc1212.txt 5 rfc1748.txt 6 rfc1694.txt 7 rfc1122.txt 10 rfc2427.txt ↩
    11 rfc950.txt 12 rfc2067.txt 17
PPP      rfc5531.txt 1 rfc2115.txt 1 rfc4861.txt 1 rfc1981.txt 1 rfc2427.txt 1 rfc4862.↩
    txt 1 rfc1659.txt 1 rfc4941.txt 2 rfc5036.txt 2 rfc2460.txt 3 rfc2863.txt 12 ↩
    rfc2865.txt 16 rfc1762.txt 19 rfc1994.txt 21 rfc1662.txt 22 rfc1989.txt 26 rfc1661.↩
    txt 40 rfc5072.txt 61 rfc1990.txt 72
TCP      rfc6152.txt 1 rfc907.txt 1 rfc919.txt 1 rfc5322.txt 1 rfc2289.txt 1 rfc4456.↩
    txt 1 rfc2067.txt 1 rfc922.txt 1 rfc868.txt 1 rfc5730.txt 1 rfc1155.txt 1 rfc1658.↩
    txt 1 rfc4941.txt 1 rfc1870.txt 1 rfc3550.txt 1 rfc2355.txt 2 rfc1044.txt 2 rfc1188↩
    .txt 2 rfc1132.txt 2 rfc1201.txt 2 rfc5065.txt 2 rfc1288.txt 2 rfc3986.txt 2 ↩
    rfc1390.txt 2 rfc894.txt 2 rfc895.txt 2 rfc1184.txt 2 rfc862.txt 3 rfc5531.txt 3 ↩
    rfc863.txt 3 rfc792.txt 3 rfc3912.txt 3 rfc3801.txt 3 rfc2895.txt 3 rfc867.txt 3 ↩
    rfc1042.txt 3 rfc866.txt 3 rfc1055.txt 3 rfc865.txt 3 rfc1356.txt 3 rfc1034.txt 5 ↩
    rfc1772.txt 5 rfc864.txt 5 rfc959.txt 5 rfc3551.txt 6 rfc4862.txt 7 rfc1939.txt 8 ↩
    rfc2741.txt 8 rfc2920.txt 9 rfc4861.txt 9 rfc854.txt 10 rfc2865.txt 10 rfc5321.txt ↩
    10 rfc2132.txt 11 rfc791.txt 12 rfc2460.txt 12 rfc1035.txt 12 rfc1981.txt 18 ↩
    rfc1006.txt 24 rfc1191.txt 25 rfc1213.txt 33 rfc1002.txt 38 rfc1123.txt 41 rfc5734.↩
    txt 42 rfc5036.txt 58 rfc5681.txt 83 rfc1001.txt 123 rfc4271.txt 126 rfc1122.txt ↩
    221 rfc793.txt 278
UDP      rfc868.txt 1 rfc1629.txt 1 rfc2348.txt 1 rfc2132.txt 1 rfc1055.txt 1 rfc950.↩
    txt 1 rfc5531.txt 2 rfc791.txt 2 rfc4862.txt 2 rfc1034.txt 2 rfc3411.txt 2 rfc2453.↩
    txt 2 rfc867.txt 3 rfc862.txt 3 rfc1981.txt 3 rfc1350.txt 3 rfc863.txt 3 rfc792.txt↩
     3 rfc1191.txt 3 rfc866.txt 3 rfc865.txt 3 rfc2895.txt 3 rfc864.txt 4 rfc3551.txt 4↩
     rfc4502.txt 5 rfc2131.txt 5 rfc768.txt 6 rfc2460.txt 8 rfc5036.txt 10 rfc951.txt ↩
    11 rfc3417.txt 12 rfc1035.txt 13 rfc3550.txt 15 rfc1213.txt 19 rfc1542.txt 21 ↩
    rfc2865.txt 24 rfc1123.txt 25 rfc1001.txt 33 rfc1002.txt 50 rfc1122.txt 65
```

3.3 Output of parallel version

```
HDLC     rfc2865.txt 1 rfc1122.txt 1 rfc891.txt 2 rfc907.txt 2 rfc2863.txt 3 rfc1662.↩
    txt 4
LAN rfc2613.txt 1 rfc1044.txt 1 rfc4862.txt 1 rfc1123.txt 1 rfc2348.txt 1 rfc3461.txt ↩
    1 rfc1661.txt 1 rfc1155.txt 2 rfc5321.txt 2 rfc2115.txt 2 rfc1629.txt 3 rfc1559.txt↩
     3 rfc1724.txt 3 rfc2895.txt 4 rfc1660.txt 5 rfc1213.txt 5 rfc1659.txt 5 rfc1658.↩
    txt 5 rfc1212.txt 5 rfc1748.txt 6 rfc1694.txt 7 rfc1122.txt 10 rfc2427.txt 11 ↩
    rfc950.txt 12 rfc2067.txt 17
PPP rfc5531.txt 1 rfc2115.txt 1 rfc4861.txt 1 rfc1981.txt 1 rfc2427.txt 1 rfc4862.txt ↩
    1 rfc1659.txt 1 rfc4941.txt 2 rfc5036.txt 2 rfc2460.txt 3 rfc2863.txt 12 rfc2865.↩
    txt 16 rfc1762.txt 19 rfc1994.txt 21 rfc1662.txt 22 rfc1989.txt 26 rfc1661.txt 40 ↩
    rfc5072.txt 61 rfc1990.txt 72
TCP rfc6152.txt 1 rfc907.txt 1 rfc919.txt 1 rfc5322.txt 1 rfc2289.txt 1 rfc4456.txt 1 ↩
    rfc2067.txt 1 rfc922.txt 1 rfc868.txt 1 rfc5730.txt 1 rfc1155.txt 1 rfc1658.txt 1 ↩
    rfc4941.txt 1 rfc1870.txt 1 rfc3550.txt 1 rfc2355.txt 2 rfc1044.txt 2 rfc1188.txt 2↩
```

```
         rfc1132.txt 2 rfc1201.txt 2 rfc5065.txt 2 rfc1288.txt 2 rfc3986.txt 2 rfc1390.txt ↩
        2 rfc894.txt 2 rfc895.txt 2 rfc1184.txt 2 rfc862.txt 3 rfc5531.txt 3 rfc792.txt 3 ↩
        rfc863.txt 3 rfc3912.txt 3 rfc3801.txt 3 rfc2895.txt 3 rfc867.txt 3 rfc1042.txt 3 ↩
        rfc866.txt 3 rfc1055.txt 3 rfc865.txt 3 rfc1356.txt 3 rfc1034.txt 5 rfc1772.txt 5 ↩
        rfc959.txt 5 rfc864.txt 5 rfc3551.txt 6 rfc4862.txt 7 rfc1939.txt 8 rfc2741.txt 8 ↩
        rfc2920.txt 9 rfc4861.txt 9 rfc854.txt 10 rfc2865.txt 10 rfc5321.txt 10 rfc2132.txt↩
         11 rfc791.txt 12 rfc2460.txt 12 rfc1035.txt 12 rfc1981.txt 18 rfc1006.txt 24 ↩
        rfc1191.txt 25 rfc1213.txt 33 rfc1002.txt 38 rfc1123.txt 41 rfc5734.txt 42 rfc5036.↩
        txt 58 rfc5681.txt 83 rfc1001.txt 123 rfc4271.txt 126 rfc1122.txt 221 rfc793.txt ↩
        278
    UDP rfc868.txt 1 rfc1629.txt 1 rfc2348.txt 1 rfc2132.txt 1 rfc1055.txt 1 rfc950.txt 1 ↩
        rfc5531.txt 2 rfc791.txt 2 rfc4862.txt 2 rfc1034.txt 2 rfc3411.txt 2 rfc2453.txt 2 ↩
        rfc862.txt 3 rfc867.txt 3 rfc1981.txt 3 rfc1350.txt 3 rfc863.txt 3 rfc792.txt 3 ↩
        rfc1191.txt 3 rfc866.txt 3 rfc865.txt 3 rfc2895.txt 3 rfc864.txt 4 rfc3551.txt 4 ↩
        rfc4502.txt 5 rfc2131.txt 5 rfc768.txt 6 rfc2460.txt 8 rfc5036.txt 10 rfc951.txt 11↩
         rfc3417.txt 12 rfc1035.txt 13 rfc3550.txt 15 rfc1213.txt 19 rfc1542.txt 21 rfc2865↩
        .txt 24 rfc1123.txt 25 rfc1001.txt 33 rfc1002.txt 50 rfc1122.txt 65
```

4 Discussions

1. File distribution over a cluster system

   MPI is Message Passing Interface, it does not need a file system to store its data. And the data is sent to another node to be computed.

   On the other hand, MapReduce is usually used with Hadoop Distributed File System. And the data is stored in local storage on each data node.

2. Collective/Reductive operation to create inverted indexing

   The implementation with MPI will be:

   - read all data, distribute the data to each node.
   - each node will compute and find the number of keywords' occurrence in their portion of data.
   - share the whole results and do the reduction.

   The biggest problem with MPI when doing inverted indexing is that it uses network to transfer data. When the size of files is large, the MPI's performance will not be ideal.

3. Amount of boilerplate code

   Comparing to MapReduce, MPI would have more boilerplate code like send, receive, and parse data. And MapReduce only need implement the Map class and Reducer class.

4. Anticipated execution performance

   Again, the limitation of MPI is the network, when the size of data/file is large and the performance could be slower than MapReduce. However, if the data is not too large, using MPI could be more efficient because the Hadoop's fault tolerance system can slow down the process.

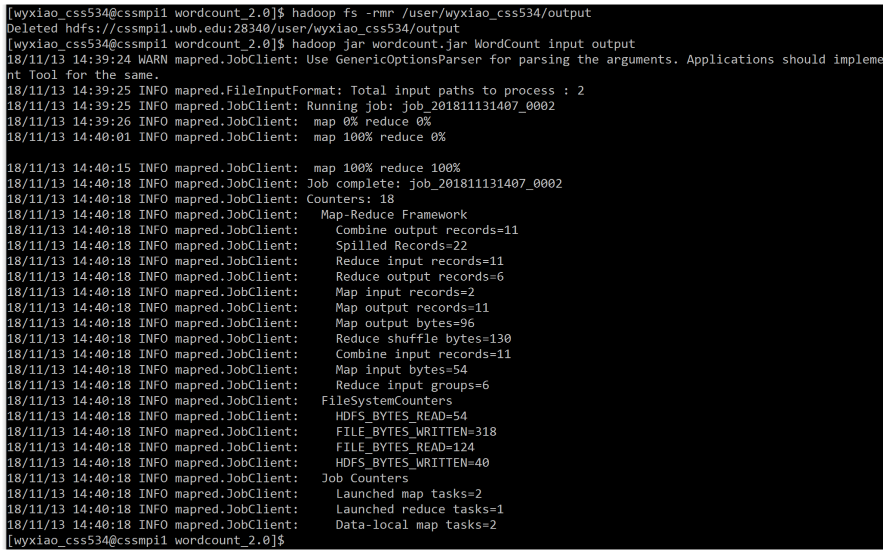
5. Fault tolerance; recovery from a crash

   MPI support checkpoint to restart from the checkpoint if anything goes wrong. It does not have Message logging techniques, data Reliability and network fault tolerance, User directed and communicator driven fault tolerance. Basically, developer can set multiple checkpoints before passing data or during iteration. However, the data could be still lost if the network is not stable.

Hadoop has its own built-in fault tolerance and fault compensation capabilities. Every data block has a copy that is stored on other servers. And it also generate logs during the execution process.

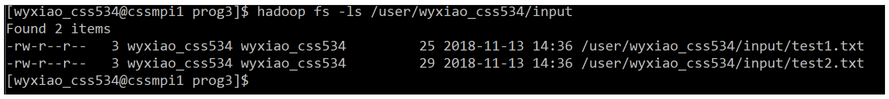
5 Lab Sessions 3

5.1 Execution output



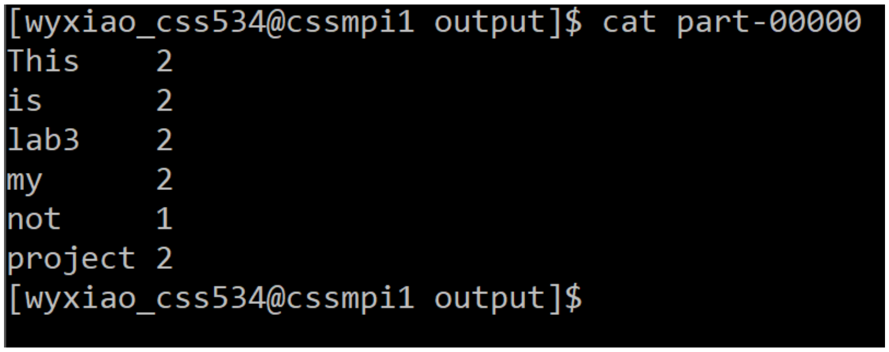
```
[wyxiao_css534@cssmpi1 wordcount_2.0]$ hadoop fs -rmr /user/wyxiao_css534/output
Deleted hdfs://cssmpi1.uwb.edu:28340/user/wyxiao_css534/output
[wyxiao_css534@cssmpi1 wordcount_2.0]$ hadoop jar wordcount.jar WordCount input output
18/11/13 14:39:24 WARN mapred.JobClient: Use GenericOptionsParser for parsing the arguments. Applications should impleme
nt Tool for the same.
18/11/13 14:39:25 INFO mapred.FileInputFormat: Total input paths to process : 2
18/11/13 14:39:25 INFO mapred.JobClient: Running job: job_201811131407_0002
18/11/13 14:39:26 INFO mapred.JobClient:  map 0% reduce 0%
18/11/13 14:40:01 INFO mapred.JobClient:  map 100% reduce 0%

18/11/13 14:40:15 INFO mapred.JobClient:  map 100% reduce 100%
18/11/13 14:40:18 INFO mapred.JobClient: Job complete: job_201811131407_0002
18/11/13 14:40:18 INFO mapred.JobClient: Counters: 18
18/11/13 14:40:18 INFO mapred.JobClient:   Map-Reduce Framework
18/11/13 14:40:18 INFO mapred.JobClient:     Combine output records=11
18/11/13 14:40:18 INFO mapred.JobClient:     Spilled Records=22
18/11/13 14:40:18 INFO mapred.JobClient:     Reduce input records=11
18/11/13 14:40:18 INFO mapred.JobClient:     Reduce output records=6
18/11/13 14:40:18 INFO mapred.JobClient:     Map input records=2
18/11/13 14:40:18 INFO mapred.JobClient:     Map output records=11
18/11/13 14:40:18 INFO mapred.JobClient:     Map output bytes=96
18/11/13 14:40:18 INFO mapred.JobClient:     Reduce shuffle bytes=130
18/11/13 14:40:18 INFO mapred.JobClient:     Combine input records=11
18/11/13 14:40:18 INFO mapred.JobClient:     Map input bytes=54
18/11/13 14:40:18 INFO mapred.JobClient:     Reduce input groups=6
18/11/13 14:40:18 INFO mapred.JobClient:   FileSystemCounters
18/11/13 14:40:18 INFO mapred.JobClient:     HDFS_BYTES_READ=54
18/11/13 14:40:18 INFO mapred.JobClient:     FILE_BYTES_WRITTEN=318
18/11/13 14:40:18 INFO mapred.JobClient:     FILE_BYTES_READ=124
18/11/13 14:40:18 INFO mapred.JobClient:     HDFS_BYTES_WRITTEN=40
18/11/13 14:40:18 INFO mapred.JobClient:   Job Counters
18/11/13 14:40:18 INFO mapred.JobClient:     Launched map tasks=2
18/11/13 14:40:18 INFO mapred.JobClient:     Launched reduce tasks=1
18/11/13 14:40:18 INFO mapred.JobClient:     Data-local map tasks=2
[wyxiao_css534@cssmpi1 wordcount_2.0]$
```

Figure 1: MapReduce execution



```
[wyxiao_css534@cssmpi1 prog3]$ hadoop fs -ls /user/wyxiao_css534/input
Found 2 items
-rw-r--r--   3 wyxiao_css534 wyxiao_css534         25 2018-11-13 14:36 /user/wyxiao_css534/input/test1.txt
-rw-r--r--   3 wyxiao_css534 wyxiao_css534         29 2018-11-13 14:36 /user/wyxiao_css534/input/test2.txt
[wyxiao_css534@cssmpi1 prog3]$
```

Figure 2: /user/yourAccount/output



```
[wyxiao_css534@cssmpi1 output]$ cat part-00000
This    2
is      2
lab3    2
my      2
not     1
project 2
[wyxiao_css534@cssmpi1 output]$
```

Figure 3: part-00000