CS 553 Cloud Computing

Programming Assignment 1

SOURCE CODE

Dinesh Chowday Jasti (A20335710)
Jayanth Vangari (A20337867)
Sai Sravan Rachiraju (A20332891)

NormalJavaWCount.java:

```
import java.io.*;
import java.util.*;
import java.util.concurrent.Semaphore;


public class NormalJavaWCount {
        private static int ThreadCount = 1;
        private static int FileCount = 16;
        private static List<String> fileList = new ArrayList<String>();
        private static HashMap<String, Integer> hmap = new HashMap<String, Integer>();
        private static String filename = null;

        @SuppressWarnings({ "unchecked" })
        public static void main(String[] args) throws InterruptedException,
                        IOException {
                BufferedReader input = new BufferedReader(new InputStreamReader(
                                System.in));
                System.out.println("Enter the File Name: ");
                filename = input.readLine().trim();

                //filesplitter(filename);
                fileListLoader();

                System.out.println("Enter number of Threads: ");
                try {
                        ThreadCount = Integer.parseInt(input.readLine().trim());
```

```
} catch (Exception e) {
}
Thread[] threadnumber = new Thread[ThreadCount];
final Semaphore semFiles = new Semaphore(ThreadCount, false);
long start = System.currentTimeMillis();
for (int i = 0; i < ThreadCount; i++) {
Thread.sleep(i * 10);
        threadnumber[i] = new Thread(new Runnable() {
                @Override
                public void run() {
                        try {
                                semFiles.acquire();
                                while (!fileList.isEmpty()) {
                                        String filename = fileList.remove(0);
                                        Wordcount(filename);
                                }
                                semFiles.release();
                        } catch (Exception ex) {
                        }
                }
        });
        threadnumber[i].start();
}
for (int i = 0; i < ThreadCount; i++) {
        threadnumber[i].join();
}
```

```
            long end = System.currentTimeMillis();
            float diff = end - start;
            System.out.println("\nTotal time taken: " + diff + " milliseconds");
            System.out.println("Execution Time(s): " + diff/1000);
System.out.println("Execution Time(min): " + diff / (1000 *60 ));
System.out.println("Execution Time(hr): " + diff / (1000 * 60 * 60) + "\n");

            FileWriter fwstream = new FileWriter("wordcount-java.txt");
            System.out.println("The total number of unique words are: " + hmap.size());
            BufferedWriter out = new BufferedWriter(fwstream);

            for(Map.Entry<String,Integer> entry : hmap.entrySet()) {
                        String key = entry.getKey();
                        Integer value = entry.getValue();
                        out.write("%20s".format(key) + ": ");
                        out.write(" %10d".format(Integer.toString(value)));
                        out.newLine();
            }

            out.close();
    }

    public static void fileListLoader() {
            String name = "file";
            for (int i = 1; i <= FileCount; i++) {
                    if (i >= 10)
```

```
                name = "file";
            fileList.add(name + i + ".txt");
        }
    }

    public static void createFilesChunk(int start, long end, String fname) {
        File file = new File(filename);
        try {
            FileReader reader = new FileReader(file);
            LineNumberReader lreader = new LineNumberReader(reader);
            String lines = "";
            FileWriter fWriter = new FileWriter(fname);
            BufferedWriter bWriter = new BufferedWriter(fWriter);
            while (lreader.getLineNumber() != start) {
                lines = lreader.readLine();
            }
            lreader.setLineNumber(start);
            while (lreader.getLineNumber() != end) {
                lines = lreader.readLine();
                bWriter.write(lines);
                bWriter.newLine();
            }
            bWriter.close();
            lreader.close();
            System.out.println(fname + ": Starting Line: " + start
                    + "          Ending Line: " + end);
```

```
        } catch (Exception ex) {
        }
}

private static Semaphore sema = new Semaphore(ThreadCount, false);

public static void Wordcount(String filename) {
        String str;
        try {
                System.out.println(Thread.currentThread().getName()
                        + "  ---> assigned to --->   " + filename);
                BufferedReader br = new BufferedReader(new FileReader(filename));
                while ((str = br.readLine()) != null) {
                        str = str.replaceAll("[^a-zA-Z0-9\\s]", "");
                        StringTokenizer stringTok = new StringTokenizer(str);
                        int count = stringTok.countTokens();
                        for (int l = 0; l < count; l++) {
                                String word = stringTok.nextToken();
                                if(word.equals(""))
                                {
                                continue;
                                }
                                else
                                {
                                sema.acquire();
                                hmap.put(word, hmap.get(word) == null ? 1
```

```
                                        : ((Integer) hmap.get(word) + 1));
                                sema.release();
                        }
                        }
                }
        } catch (Exception ex) {
        ex.printStackTrace();
        }
}

private static LineNumberReader lreader;

public static int TotalLines(String fileName) {
        int total_lines = 0;
        try {
                File f = new File(fileName);
                FileReader reader = new FileReader(f);
                lreader = new LineNumberReader(reader);
                while ((lreader.readLine()) != null) {
                }
                total_lines = lreader.getLineNumber();
        } catch (Exception ex) {
        }
        return total_lines;
}
```

```
        public static void filesplitter(String fileName) {
                int start = 0;
                int total_lines = 123015884;
                int diff = total_lines / FileCount;
                int end = diff;
                String a;
                for (int i = 1; i <= FileCount; i++) {
                        a = (i < 10 ? "file" : "file") + i + ".txt";
                        createFilesChunk(start, end, a);
                        start = end;
                        end = end + diff;
                        if ((i == (FileCount - 1)) && (end != total_lines))
                                end = total_lines;
                }
        }
}
```

SortedJavaWCount.java

```
import java.io.*;
import java.util.*;
import java.util.concurrent.Semaphore;


public class SortedJavaWCount {
        private static int ThreadCount = 1;
        private static int FileCount = 16;
        private static List<String> fileList = new ArrayList<String>();
        private static TreeMap<String, Integer> hmap = new TreeMap<String, Integer>();
        private static String filename = null;

        @SuppressWarnings({ "unchecked" })
        public static void main(String[] args) throws InterruptedException,
                        IOException {
                BufferedReader input = new BufferedReader(new InputStreamReader(
                                System.in));
                System.out.println("Enter the File Name: ");
                filename = input.readLine().trim();


                //filesplitter(filename);
                fileListLoader();


                System.out.println("Enter number of Threads: ");
```

```
try {
        ThreadCount = Integer.parseInt(input.readLine().trim());
} catch (Exception e) {
}
Thread[] threadnumber = new Thread[ThreadCount];
final Semaphore semFiles = new Semaphore(ThreadCount, false);
long start = System.currentTimeMillis();
for (int i = 0; i < ThreadCount; i++) {
Thread.sleep(i * 10);
        threadnumber[i] = new Thread(new Runnable() {
                @Override
                public void run() {
                        try {
                                semFiles.acquire();
                                while (!fileList.isEmpty()) {
                                        String filename = fileList.remove(0);
                                        Wordcount(filename);
                                }
                                semFiles.release();
                        } catch (Exception ex) {
                        }
                }
        });
        threadnumber[i].start();
}
for (int i = 0; i < ThreadCount; i++) {
```

```
                threadnumber[i].join();
            }
            FileWriter fwstream = new FileWriter("sort1mb-java.txt");
            System.out.println("The total number of unique words are: " + hmap.size());
            BufferedWriter out = new BufferedWriter(fwstream);
                        out.write("Sorted word count without frequency based on the ASCII");
            for(Map.Entry<String,Integer> entry : hmap.entrySet()) {
                        String key = entry.getKey();
                        Integer value = entry.getValue();
                        out.write("%20s".format(key));
                        out.newLine();
            }

            out.close();

            long end = System.currentTimeMillis();
            float diff = end - start;
            float splitdiff = 1348032;
            float total_time=splitdiff+diff;
            System.out.println("\nTotal time taken: " + total_time + " milliseconds");
            System.out.println("Execution Time(s): " + total_time/1000);
System.out.println("Execution Time(min): " + total_time / (1000 *60 ));
System.out.println("Execution Time(hr): " + total_time / (1000 * 60 * 60) + "\n");
}
```

```
public static void fileListLoader() {
        String name = "file";
        for (int i = 1; i <= FileCount; i++) {
                if (i >= 10)
                        name = "file";
                fileList.add(name + i + ".txt");
        }
}

public static void createFilesChunk(int start, long end, String fname) {
        File file = new File(filename);
        try {
                FileReader reader = new FileReader(file);
                LineNumberReader lreader = new LineNumberReader(reader);
                String lines = "";
                FileWriter fWriter = new FileWriter(fname);
                BufferedWriter bWriter = new BufferedWriter(fWriter);
                while (lreader.getLineNumber() != start) {
                        lines = lreader.readLine();
                }
                lreader.setLineNumber(start);
                while (lreader.getLineNumber() != end) {
                        lines = lreader.readLine();
                        bWriter.write(lines);
                        bWriter.newLine();
                }
```

```
                bWriter.close();
                lreader.close();
                System.out.println(fname + ": Starting Line: " + start
                        + "          Ending Line: " + end);
        } catch (Exception ex) {
        }
}

private static Semaphore sema = new Semaphore(ThreadCount, false);

public static void Wordcount(String filename) {
        String str;
        try {
                System.out.println(Thread.currentThread().getName()
                        + " ---> assigned to ---> " + filename);
                BufferedReader br = new BufferedReader(new FileReader(filename));
                while ((str = br.readLine()) != null) {
                        str = str.replaceAll("[^a-zA-Z0-9\\s]", "");
                        StringTokenizer stringTok = new StringTokenizer(str);
                        int count = stringTok.countTokens();
                        for (int l = 0; l < count; l++) {
                                String word = stringTok.nextToken();
                                if(word.equals(""))
                                {
                                continue;
                                }
```

```
                                else
                                {
                                sema.acquire();
                                hmap.put(word, hmap.get(word) == null ? 1
                                        : ((Integer) hmap.get(word) + 1));
                                sema.release();
                        }
                        }
                }
        } catch (Exception ex) {
        ex.printStackTrace();
        }
}

private static LineNumberReader lreader;

public static int TotalLines(String fileName) {
        int total_lines = 0;
        try {
                File f = new File(fileName);
                FileReader reader = new FileReader(f);
                lreader = new LineNumberReader(reader);
                while ((lreader.readLine()) != null) {
                }
                total_lines = lreader.getLineNumber();
        } catch (Exception ex) {
```

```
            }
            return total_lines;
        }

        public static void filesplitter(String fileName) {
            int start = 0;
            int total_lines = TotalLines(fileName);
            int diff = total_lines / FileCount;
            int end = diff;
            String a;
            for (int i = 1; i <= FileCount; i++) {
                a = (i < 10 ? "file" : "file") + i + ".txt";
                createFilesChunk(start, end, a);
                start = end;
                end = end + diff;
                if ((i == (FileCount - 1)) && (end != total_lines))
                    end = total_lines;
            }
        }
    }
```

Hadoop Word Count:

Wc.jar

```
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.mapreduce.Job;

public class WordCount {

  public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable>
  {
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException
    {
      String line = value.toString();
        String temp;
      StringTokenizer tokenizer = new StringTokenizer(line);
      while (tokenizer.hasMoreTokens())
      {
```

```
                temp = tokenizer.nextToken();
                temp = temp.replaceAll("[^a-zA-Z0-9\\s]","");
                word.set(temp);
            output.collect(word, one);
        }
      }
    }


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


 public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable>
output, Reporter reporter) throws IOException
        {


        int sum = 0;
        while (values.hasNext())
           {
           sum += values.next().get();
        }
        output.collect(key, new IntWritable(sum));
      }
  }
```

```
  public static class Map_sort extends MapReduceBase implements Mapper<Object, Text,
IntWritable, Text>
  {


      public void map(Object key, Text value, OutputCollector<IntWritable, Text> collector,
Reporter arg3) throws IOException
          {
          String line = value.toString();
          StringTokenizer stringTokenizer = new StringTokenizer(line);
          {
             int number=0;
             String word=null;


             if (stringTokenizer.hasMoreTokens())
                  {
                String str0 = stringTokenizer.nextToken();
                word = str0.trim();
             }


             if (stringTokenizer.hasMoreElements())
                  {
                String str1 = stringTokenizer.nextToken();
                number = Integer.parseInt(str1.trim());
             }
             collector.collect(new IntWritable(number), new Text(word));
          }
```

```
    }


  }


  public static class Reduce_sort extends MapReduceBase implements Reducer<IntWritable, Text, IntWritable, Text>

  {


    public void reduce(IntWritable key, Iterator<Text> values, OutputCollector<IntWritable, Text> arg2, Reporter arg3) throws IOException

      {

      while ((values.hasNext()))

        {

        arg2.collect(key, values.next());

      }

    }


  }


  public static void main(String[] args) throws Exception {

    JobConf conf = new JobConf(WordCount.class);

    conf.setJobName("wordCount");


    conf.setOutputKeyClass(Text.class);

    conf.setOutputValueClass(IntWritable.class);
```

```
    conf.setMapperClass(Map.class);
    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("/tmp/temp"));

       Job job1 = new Job(conf);
       job1.submit();
//JobClient.runJob(conf);
    //----------------------------------------------------------------
    JobConf conf2  = new JobConf(WordCount.class);
    conf2.setJobName("WordCount1");

    conf2.setOutputKeyClass(IntWritable.class);
    conf2.setOutputValueClass(Text.class);

    conf2.setMapperClass(Map_sort.class);
    conf2.setCombinerClass(Reduce_sort.class);
    conf2.setReducerClass(Reduce_sort.class);
```

```
        conf2.setInputFormat(TextInputFormat.class);
        conf2.setOutputFormat(TextOutputFormat.class);

        FileInputFormat.setInputPaths(conf2, new Path("/tmp/temp/part-00000"));
        FileOutputFormat.setOutputPath(conf2, new Path(args[1]));


        Job job2 = new Job(conf2);
        if (job1.waitForCompletion(true))
           {
           job2.submit();
              job2.waitForCompletion(true);
        }

    }
}
```

Ws.jar: for sorted hadoop:

```
import java.io.IOException;

import java.util.*;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.*;

import org.apache.hadoop.mapred.*;

import org.apache.hadoop.mapreduce.Job;


public class WordSort {


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

   {

      private Text word = new Text();


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

      {

         String line = value.toString();

            String temp;

         StringTokenizer tokenizer = new StringTokenizer(line);

         while (tokenizer.hasMoreTokens())

         {

                  temp = tokenizer.nextToken();

                  temp = temp.replaceAll("[^a-zA-Z0-9\\s]","");

                  word.set(temp);
```

```
            output.collect(word,  new Text(""));
          }
        }
      }

    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
          {
          output.collect(key,  new Text(""));
        }
    }

    public static void main(String[] args) throws Exception {
        JobConf conf = new JobConf(WordSort.class);
        conf.setJobName("WordSort");

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

        conf.setMapperClass(Map.class);
        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]));

           Job job1 = new Job(conf);
           job1.submit();
        job1.waitForCompletion(true);
    }
}
```

Configuration files hadoop:

The following files will have to be modified to complete the Hadoop setup:

- ~/.bashrc
- / conf /hadoop-env.sh
- / conf /core-site.xml
- / conf /yarn-site.xml
- / conf /hdfs-site.xml
- / conf /mapred-site.xml.template

1. ~/.bashrc:

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
export HADOOP_INSTALL=/usr/ubuntu/installs/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
eval ($ssh-agent);
ssh-add/home/ubuntu/dj_key.pem
#HADOOP VARIABLES END
```

   Reboot the connection now for the changes to be reflected.

2. / conf /hadoop-env.sh:

   This is to set the JAVA Path by modifying **hadoop-env.sh** file

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-amd64
```

3. / conf /core-site.xml:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up. This file can be used to override the default settings that Hadoop starts with.

```
<configuration>
<property>
<name>fs.default.name</name>
 <value>hdfs://<172.31.22.90>:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/home/ubuntu/installs/hadoop/hadoop_tmp/tmp</value>
</property>
</configuration>
```

4. / conf /yarn-site.xml:

```
<configuration>
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value><172.31.22.90>:8025</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value><172.31.22.90>:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value><172.31.22.90>:8040</value>
</property>
</configuration>
```

5. / conf /hdfs-site.xml:

```
<configuration>
<property>
<name>dfs.replication</name>
 <value>16</value>
</property>
<property>
 <name>dfs.permissions</name>
 <value>false</value>
 </property>
</configuration>
```

6. /conf /mapred-site.xml:

By default, the /usr/local/hadoop/etc/hadoop/ contains the /usr/local/hadoop/etc/hadoop/mapred-site.xml.template file which has to be renamed/copied with the name mapred-site.xml:

"cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template/usr/local/hadoop/etc/hadoop/mapred-site.xml"

The **mapred-site.xml** file is used to specify which framework is being used for MapReduce. We need to enter the following content in between the <configuration></configuration> tag:

```
<configuration>
<property>
 <name>mapreduce.framework.name</name>
<value>yarn</value>
 </property>
</configuration>
```

Swift:

regularexpressionmapper.swift:

```
type filee;

type count;

app (count t) countwords (filee f) {

wc "-w" @filename(f) stdout=@filename(t);}

messagefile inputfile <"regularexpressionmapper.words.txt">;

countfile c <regularexpressionmapper;
source=@inputfile,match="(.*)txt",transform="\\1count">;

c =countwords(inputfile);
```

mapper.swift:

```
type filee;

type count;

app (count t) countwords(filee f) {

wc "-w" @filename(f) stdout=@filename(t);

}

string inputNames = "mapper1.txt";

string outputNames = "mapper1.count ";

filee inputfiles[] <mapper;files=inputNames>;

count outputfiles[] <mapper;files=outputNames>;

outputfiles[0] = countwords(inputfiles[0]);
```

wordcount.swift:

```
type filee ;
type count;
app (count t) countwords (filee f) {
wc "-w" @filename(f) stdout=@filename(t);
}
string inputName = "wiki10gb.txt";
filee inputs[] <mapper;files=inputName>;
foreach f in inputs {
count c<regularexpressionmapper;source=@f, match="(.*)txt",transform="\\1count">;
c = countwords(f);
}
```

MPI Sorting:

```
#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
#include <boost/mpi.hpp>
#include <boost/algorithm/string.hpp>
#include <boost/serialization/map.hpp>

using namespace std;
namespace mpi=::boost::mpi;

void print_help()
{
   cout << "Usage: mpiexec word-count <path to file>"<< endl;
}

char separator[] = " ,.\n-+;:!?()\t[]{}<>'`\"";

bool isseparator (char c) {
   char* e = separator + sizeof(separator) / sizeof(separator[0]);
   char* pos = std::find(separator, e, c);
   return (pos != e);
}
```

```
struct sort_reverse {
    bool operator()(const std::pair<string,int> &left, const std::pair<string,int> &right) {
        return left.second > right.second;
    }
};

int main(int argc, char *argv[])
{
    mpi::environment env(argc, argv);
    mpi::communicator world;

    if (argc != 2)
    {
        if (world.rank() == 0)
        {
            print_help();
        }
        return 1;
    }

    int chunksize;
    int mapnodescount = world.size() - 1;
    int masterrank = 0;
    int chunk_sizemax;
    unsigned int maxoutputlines = 25;
```

```
//masterrank vars
char* buf;
int buf_len;
vector<map<string, unsigned long long int>> stats;
map<string, unsigned long long int> result;

//slaves vars
map<string, unsigned long long int> stat;
// vector<string, unsigned long long int> stat_plain;
char* input = NULL;

if (world.rank() == masterrank)
{
   // read input file
   string word;
   ifstream infile(argv[1]);

   infile.seekg(0, ios::end);
   buf_len = infile.tellg();
   infile.seekg(0, ios::beg);
   buf = new char[buf_len];
   infile.read(buf, buf_len);
   infile.close();
```

```
chunksize  = buf_len  / mapnodescount;
chunk_sizemax  = chunksize  * 2;

mpi::broadcast(world,  chunk_sizemax,  masterrank);

// split in chunks and send
int i, start_index  = 0;
for (i = 1; i < mapnodescount;  i ++){
   if (start_index  >= buf_len)
   {
      world.send(i,  0, buf + buf_len,  1);
   }
   else
   {
      int size  = chunksize;
      while  (size < chunk_sizemax  && start_index  + size <= buf_len  &&
            !isseparator(buf[start_index  + size  - 1]))
      {
         size++;
      }
      buf[start_index  + size  - 1] = 0;
      world.send(i,  0, buf + start_index,  max(size,  1));
      start_index  += size;
   }
}
```

```
    world.send(i, 0, buf + min(start_index, buf_len), max(buf_len - start_index + 1, 1));
}
else {
    // get inputs
    mpi::broadcast(world, chunk_sizemax, masterrank);
    input = new char[chunk_sizemax];
    world.recv(masterrank, 0, input, chunk_sizemax);
    //cout << "worker #" << world.rank() << ": '" << input << "'" << endl; //DEBUG
}
//barrier and timing
world.barrier();
mpi::timer timer;

// do word count
if(world.rank() != masterrank) {
    char * word;
    word = strtok(input,separator);
    while (word != NULL)
    {
        string s_word(word);
        boost::algorithm::to_lower(s_word);
        stat[s_word] += 1;
        word = strtok (NULL, separator);
    }
    vector<string> smth;
```

```
        world.send(masterrank, 1, smth);
    }

    // collect results
    mpi::gather(world, stat, stats, masterrank);

    if(world.rank() == masterrank)
    {
        for(vector<map<string, unsigned long long int>>::iterator it = stats.begin(); it != stats.end();
++it) {
            for (map<string, unsigned long long int>::iterator vit = it->begin(); vit != it->end();
++vit) {
                result[vit->first] += vit->second;
            }
        }
    }

    // output sorted results
    if (world.rank() == masterrank)
    {
        cout << "Time: " << timer.elapsed() << "s" << endl;
        cout << "Total words: " << result.size() << endl;
        vector<pair<string, unsigned long long int>> output;
        for (map<string, unsigned long long int>::iterator it = result.begin(); it != result.end(); ++it)
{
            output.push_back(make_pair(it->first, it->second));
```

```
        }
        sort(output.begin(), output.end(), sort_reverse());
        for (unsigned int i = 0; i < output.size() && i < maxoutputlines; i++) {
            cout << output[i].first << " => " << output[i].second << endl;
        }
    }
    return 0;
}
```

Config:

```
#####################################
## StarCluster Configuration File ##
#####################################
[global]
DEFAULT_TEMPLATE=akshara

#############################################
## AWS Credentials and Connection Settings ##
#############################################
[aws info]
AWS_ACCESS_KEY_ID =AKIAI2QU45XPVS6SEAQQ
AWS_SECRET_ACCESS_KEY =GjbFZIf4xks9oQeWzFf1lkb3g74jxtz2414K6vfj
#your_secret_access_key
AWS_USER_ID= akshara #your userid
```

```
AWS_REGION_NAME = eu-west-2a
AWS_REGION_HOST=eu-west-2a.amazonaws.com

## Defining EC2 Keypairs ##
[key akshu.rsa]
KEY_LOCATION=/home/ubuntu/akshu.rsa

## Defining Cluster Templates ##
    $ starcluster start akshara

[cluster akshara]
KEYNAME = akshu.rsa
CLUSTER_SIZE = 2
CLUSTER_USER = sgeadmin
CLUSTER_SHELL = bash




# Uncomment to prepent the cluster tag to the dns name of all nodes created
# using this cluster config. ie: mycluster-master and mycluster-node001
# If you choose to enable this option, it's recommended that you enable it in
# the DEFAULT_TEMPLATE so all nodes will automatically have the prefix
# DNS_PREFIX = True
```

#AMI to use for cluster nodes. These AMIs are for the us-east-1 region.

Use the 'listpublic' command to list StarCluster AMIs in other regions

The base i386 StarCluster AMI is ami-9bf9c9f2

The base x86_64 StarCluster AMI is ami-3393a45a

The base HVM StarCluster AMI is ami-6b211202

NODE_IMAGE_ID = ami-6b211202

instance type for all cluster nodes

(options: m3.large, i2.8xlarge, c3.2xlarge, hs1.8xlarge, c1.xlarge, r3.4xlarge, g2.2xlarge, m1.small, c1.medium, m3.2xlarge, c3.8xlarge, m2.xlarge, r3.2xlarge, t1.micro, cr1.8xlarge, r3.8xlarge, cc1.4xlarge, m1.medium, r3.large, c3.xlarge, i2.xlarge, m3.medium, cc2.8xlarge, m1.large, cg1.4xlarge, i2.2xlarge, c3.large, i2.4xlarge, c3.4xlarge, r3.xlarge, m1.xlarge, hi1.4xlarge, m2.4xlarge, m2.2xlarge, m3.xlarge)

NODE_INSTANCE_TYPE = c3.large

Launch cluster in a VPC subnet (OPTIONAL)

#SUBNET_ID=subnet-99999999

Uncomment to assign public IPs to cluster nodes (VPC-ONLY) (OPTIONAL)

WARNING: Using public IPs with a VPC requires:

1. An internet gateway attached to the VPC

2. A route table entry linked to the VPC's internet gateway and associated

with the VPC subnet with a destination CIDR block of 0.0.0.0/0

WARNING: Public IPs allow direct access to your VPC nodes from the internet

#PUBLIC_IPS=True

Uncomment to disable installing/configuring a queueing system on the

cluster (SGE)

#DISABLE_QUEUE=True

Uncomment to specify a different instance type for the master node (OPTIONAL)

(defaults to NODE_INSTANCE_TYPE if not specified)

```
MASTER_INSTANCE_TYPE = c3.large
# Uncomment to specify a separate AMI to use for the master node. (OPTIONAL)
# (defaults to NODE_IMAGE_ID if not specified)
MASTER_IMAGE_ID = ami-6b211202
# availability zone to launch the cluster in (OPTIONAL)
# (automatically determined based on volumes (if any) or
# selected by Amazon if not specified)
#AVAILABILITY_ZONE = us-east-1c
# list of volumes to attach to the master node (OPTIONAL)
# these volumes, if any, will be NFS shared to the worker nodes
# see "Configuring EBS Volumes" below on how to define volume sections
#VOLUMES = oceandata, biodata
# list of plugins to load after StarCluster's default setup routines (OPTIONAL)
# see "Configuring StarCluster Plugins" below on how to define plugin sections
#PLUGINS = myplugin, myplugin2
# list of permissions (or firewall rules) to apply to the cluster's security
# group (OPTIONAL).
#PERMISSIONS = ssh, http
# Uncomment to always create a spot cluster when creating a new cluster from
# this template. The following example will place a $0.50 bid for each spot
# request.
#SPOT_BID = 0.50
# Uncomment to specify one or more userdata scripts to use when launching
# cluster instances. Supports cloudinit. All scripts combined must be less than
# 16KB
```

```
#USERDATA_SCRIPTS = /path/to/script1, /path/to/script2


##########################################
## Defining Additional Cluster Templates ##
##########################################
# You can also define multiple cluster templates. You can either supply all
# configuration options as with smallcluster above, or create an
# EXTENDS=<cluster_name> variable in the new cluster section to use all
# settings from <cluster_name> as defaults. Below are example templates that
# use the EXTENDS feature:

# [cluster mediumcluster]
# Declares that this cluster uses smallcluster as defaults
# EXTENDS=smallcluster
# This section is the same as smallcluster except for the following settings:
# KEYNAME=myotherkey
# NODE_INSTANCE_TYPE = c1.xlarge
# CLUSTER_SIZE=8
# VOLUMES = biodata2

# [cluster largecluster]
# Declares that this cluster uses mediumcluster as defaults
# EXTENDS=mediumcluster
# This section is the same as mediumcluster except for the following variables:
# CLUSTER_SIZE=16
```

```
#############################
## Configuring EBS Volumes ##
#############################
# StarCluster can attach one or more EBS volumes to the master and then
# NFS_share these volumes to all of the worker nodes. A new [volume] section
# must be created for each EBS volume you wish to use with StarCluser. The
# section name is a tag for your volume. This tag is used in the VOLUMES
# setting of a cluster template to declare that an EBS volume is to be mounted
# and nfs shared on the cluster. (see the commented VOLUMES setting in the
# example 'smallcluster' template above) Below are some examples of defining
# and configuring EBS volumes to be used with StarCluster:

# Sections starting with "volume" define your EBS volumes
# [volume biodata]
# attach vol-c9999999 to /home on master node and NFS-shre to worker nodes
# VOLUME_ID = vol-c999999
# MOUNT_PATH = /home

# Same volume as above, but mounts to different location
# [volume biodata2]
# VOLUME_ID = vol-c999999
# MOUNT_PATH = /opt/

# Another volume example
```

```
# [volume oceandata]
# VOLUME_ID = vol-d7777777
# MOUNT_PATH = /mydata

# By default StarCluster will attempt first to mount the entire volume device,
# failing that it will try the first partition. If you have more than one
# partition you will need to set the PARTITION number, e.g.:
# [volume oceandata]
# VOLUME_ID = vol-d7777777
# MOUNT_PATH = /mydata
# PARTITION = 2

############################################
## Configuring Security Group Permissions ##
############################################
# Sections starting with "permission" define security group rules to
# automatically apply to newly created clusters. IP_PROTOCOL in the following
# examples can be can be: tcp, udp, or icmp. CIDR_IP defaults to 0.0.0.0/0 or
# "open to the # world"

# open port 80 on the cluster to the world
# [permission http]
# IP_PROTOCOL = tcp
# FROM_PORT = 80
# TO_PORT = 80
```

```
# open https on the cluster to the world
# [permission https]
# IP_PROTOCOL = tcp
# FROM_PORT = 443
# TO_PORT = 443

# open port 80 on the cluster to an ip range using CIDR_IP
# [permission http]
# IP_PROTOCOL = tcp
# FROM_PORT = 80
# TO_PORT = 80
# CIDR_IP = 18.0.0.0/8

# restrict ssh access to a single ip address (<your_ip>)
# [permission ssh]
# IP_PROTOCOL = tcp
# FROM_PORT = 22
# TO_PORT = 22
# CIDR_IP = <your_ip>/32


########################################
## Configuring StarCluster Plugins ##
########################################
```

```
# Sections starting with "plugin" define a custom python class which perform
# additional configurations to StarCluster's default routines. These plugins
# can be assigned to a cluster template to customize the setup procedure when
# starting a cluster from this template (see the commented PLUGINS setting in
# the 'smallcluster' template above). Below is an example of defining a user
# plugin called 'myplugin':

# [plugin myplugin]
# NOTE: myplugin module must either live in ~/.starcluster/plugins or be
# on your PYTHONPATH
# SETUP_CLASS = myplugin.SetupClass
# extra settings are passed as __init__ arguments to your plugin:
# SOME_PARAM_FOR_MY_PLUGIN = 1
# SOME_OTHER_PARAM = 2

######################
## Built-in Plugins ##
######################
# The following plugins ship with StarCluster and should work out-of-the-box.
# Uncomment as needed. Don't forget to update your PLUGINS list!
# See http://star.mit.edu/cluster/docs/latest/plugins for plugin details.
#
# Use this plugin to install one or more packages on all nodes
# [plugin pkginstaller]
# SETUP_CLASS = starcluster.plugins.pkginstaller.PackageInstaller
```

```
# # list of apt-get installable packages
# PACKAGES = mongodb, python-pymongo
#
# Use this plugin to create one or more cluster users and download all user ssh
# keys to $HOME/.starcluster/user_keys/<cluster>-<region>.tar.gz
# [plugin createusers]
# SETUP_CLASS = starcluster.plugins.users.CreateUsers
# NUM_USERS = 30
# # you can also comment out NUM_USERS and specify exact usernames, e.g.
# # usernames = linus, tux, larry
# DOWNLOAD_KEYS = True
#
# Use this plugin to configure the Condor queueing system
# [plugin condor]
# SETUP_CLASS = starcluster.plugins.condor.CondorPlugin
#
# The SGE plugin is enabled by default and not strictly required. Only use this
# if you want to tweak advanced settings in which case you should also set
# DISABLE_QUEUE=TRUE in your cluster template. See the plugin doc for more
# details.
# [plugin sge]
# SETUP_CLASS = starcluster.plugins.sge.SGEPlugin
# MASTER_IS_EXEC_HOST = False
#
# The IPCluster plugin configures a parallel IPython cluster with optional
```

```
# web notebook support. This allows you to run Python code in parallel with low
# latency message passing via ZeroMQ.
# [plugin ipcluster]
# SETUP_CLASS = starcluster.plugins.ipcluster.IPCluster
# # Enable the IPython notebook server (optional)
# ENABLE_NOTEBOOK = True
# # Set a password for the notebook for increased security
# # This is optional but *highly* recommended
# NOTEBOOK_PASSWD = a-secret-password
# # Set a custom directory for storing/loading notebooks (optional)
# NOTEBOOK_DIRECTORY = /path/to/notebook/dir
# # Set a custom packer. Must be one of 'json', 'pickle', or 'msgpack'
# # This is optional.
# PACKER = pickle
#
# Use this plugin to create a cluster SSH "dashboard" using tmux. The plugin
# creates a tmux session on the master node that automatically connects to all
# the worker nodes over SSH. Attaching to the session shows a separate window
# for each node and each window is logged into the node via SSH.
# [plugin tmux]
# SETUP_CLASS = starcluster.plugins.tmux.TmuxControlCenter
#
# Use this plugin to change the default MPI implementation on the
# cluster from OpenMPI to MPICH2.
# [plugin mpich2]
```

```
# SETUP_CLASS = starcluster.plugins.mpich2.MPICH2Setup
#
# Configure a hadoop cluster. (includes dumbo setup)
# [plugin hadoop]
# SETUP_CLASS = starcluster.plugins.hadoop.Hadoop
#
# Configure a distributed MySQL Cluster
# [plugin mysqlcluster]
# SETUP_CLASS = starcluster.plugins.mysql.MysqlCluster
# NUM_REPLICAS = 2
# DATA_MEMORY = 80M
# INDEX_MEMORY = 18M
# DUMP_FILE = test.sql
# DUMP_INTERVAL = 60
# DEDICATED_QUERY = True
# NUM_DATA_NODES = 2
#
# Install and setup an Xvfb server on each cluster node
# [plugin xvfb]
# SETUP_CLASS = starcluster.plugins.xvfb.XvfbSetup
```