CS553 Cloud Computing

Programming Assignment #2

Source Code

Pranitha Nagavelli(A20345406)

Shared Memory Sort Java Code:

sort_thread

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.LineNumberReader;
import java.io.RandomAccessFile;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;
public class Sort_Thread extends Thread {
	int filecount=0;
	int lineno=0;
	String tmpath="";
	int threads=0;
	long part_size=150000;
	int begin=1;
	int end=0;
	long ptr;
	String input;
	String output;
	public Sort_Thread(int th, String input,long ptr,String output) {
	threads=th;
	this.input=input;
	this.output=output;
	this.ptr=ptr;
	  public static int BUFFERSIZE = 2048;
	    public BufferedReader br;
	    public File originalfile;
	    private String cache;
	    private boolean empty;
	    br = new BufferedReader(new FileReader(f), BUFFERSIZE);
	}

	@Override
     public synchronized void run() {

		try {
		      String line="";
		     RandomAccessFile rf   =new RandomAccessFile( input,"rw");
		     rf.seek(ptr);
		     Map<String, String> list=new HashMap<String, String>();
		     long length=rf.length();
		     long blocksize=length/threads;
		    long noofparts=length/part_size;
```

```
                long extra=length%part_size;
                while(rf.readLine()!=null)
            {
               line=rf.readLine();
               list.put(line.substring(0,10),line.substring(10));
               lineno++;
                     if(list.size()>part_size)
                  {
                            filecount++;

                    BufferedWriter split = new BufferedWriter(new
FileWriter(output+"split"+filecount+".txt"));
                  try
                  {
                    for(int i=0;i<lineno;i++){
                                split.write(list.get(i)); }
                    list.clear();
                    lineno=0;

                    }
                        catch(Exception e)
                        {
                              throw new RuntimeException(e);
                        }
                        finally
                        {
                        split.close();

                        }
                 }
                     //if listsize is lessthan or equal to partSize
                  BufferedWriter split = new BufferedWriter(new
FileWriter(output+"split"+filecount+".txt"));
             try
             {
               for(int i=0;i<lineno;i++){
                           split.write(list.get(i)); }
               list.clear();
               lineno=0;

               }
                   catch(Exception e)
                   {
                         throw new RuntimeException(e);
                   }
                   finally
                   {
                   split.close();

                   }



             }

             } catch (IOException e) {
                   // TODO Auto-generated catch block
```

```
                    e.printStackTrace();

                }
                mergesort(filecount,output);
        }

        public static void mergesort(int filecount,String output)
        {
                Map<String, String> hmap = new HashMap<String, String>();
                int cn=filecount;
                String out=output;
                for(int i=0;i<cn;i++){
                 BufferedReader in;
                    try {
                            in = new BufferedReader(new
FileReader(out+"split"+i+".txt"));
                    } catch (FileNotFoundException e) {
                            // TODO Auto-generated catch block
                            e.printStackTrace();
                    }
                 String line;
                try {
                    while ((line = in.readLine()) != null)
                        {
                    String key = line.substring(0, Math.min(line.length(), 9));
                    String value = line.substring(9, Math.min(line.length(),
99));
                    hmap.put(key,value);
                        }
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                       Set set = hmap.entrySet();
                       Iterator iterator = set.iterator();
                       while(iterator.hasNext()) {
                             Map.Entry me = (Map.Entry)iterator.next();
                       }
                       Map<String, String> map = new TreeMap<String,
String>(hmap);
                       Set set2 = map.entrySet();
                       Iterator iterator2 = set2.iterator();
                       while(iterator2.hasNext()) {
                            Map.Entry me2 = (Map.Entry)iterator2.next();

                }

                   FileWriter fstream;
                try {
                    fstream = new FileWriter(out+"sort"+i+".txt");
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                    Iterator itera = (Iterator) map.keySet().iterator();
                    Iterator itera1 = (Iterator) map.values().iterator();
                               while( itera.hasNext())
```

```
                                {
                                  String itera2 = itera.next().toString();
                                   String itera3 = itera1.next().toString();
                                   try {
                                                fstream.write(itera2);
                                           } catch (IOException e) {
                                                // TODO Auto-generated catch
block
                                                e.printStackTrace();
                                           }
                                   try {
                                                fstream.write(itera3);
                                           } catch (IOException e) {
                                                // TODO Auto-generated catch
block
                                                e.printStackTrace();
                                           }
                                    try {

      fstream.write(System.getProperty( "line.separator" ));
                                           } catch (IOException e) {
                                                // TODO Auto-generated catch
                                                e.printStackTrace();
                                           }
                                  }
                             fstream.close();
                in.close();
              }
       }

        public String peek() {
              return cache.toString();
          }
}
```

Main Function:

```
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.EOFException;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.Comparator;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.PriorityQueue;
import java.util.Scanner;

public class Sort {
```

```
public static long partSize=150000;
public static int th=0;
public static int count=0;
public static long ptr=0;
public static long filecount=0;
      public static void main(String[] args) {
            // TODO Auto-generated method stub
            String input=args[0];
            String output=args[1];
            RandomAccessFile inputfile;
            try {
                  inputfile = new RandomAccessFile("input", "r");
            } catch (FileNotFoundException e2) {
                  // TODO Auto-generated catch block
                  e2.printStackTrace();
            }
            long fileSize;
            try {
                  fileSize = inputfile.length();
            } catch (IOException e1) {
                  // TODO Auto-generated catch block
                  e1.printStackTrace();
            }
            String line="";
            System.out.println("Enter num of theards");
            Scanner in=new Scanner(System.in);
            th=in.nextInt();
            int j;
              filecount=fileSize/partSize;
              if(fileSize%partSize!=0)
              {
                    filecount++;
              }
              long start = System.currentTimeMillis();
              for(int i=1;i<=th;i++){
                        count++;
                        Sort_Thread thread=new Sort_Thread(th,input,ptr,output);
                   thread.run();
                  ptr=ptr+partSize*count;
                    }
              try {
                  inputfile.close();
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
for(int i=1;i<=filecount;i++){
                  File fp=new File(output+"split"+i+".txt");;
                  fp.delete();

            }

       mergeSortedFiles(output,filecount);
            long stop = System.currentTimeMillis();
            long time = stop-start;
            System.out.println("Time taken for execution is: "+ time);
            }
```

```
       public static int mergeSortedFiles(String output,long filecount)
throws IOException {

               PriorityQueue<thread> pq = new PriorityQueue<thread>(new
Comparator<BinaryFileBuffer>() {
                        public int compare(thread i, thread j) {
                          return cmp.compare(i.peek(), j.peek());
                        }
                    }
                );
                for (File f : files) {
                    Sort_thread  t = new sort_Thread thread(t);
                    pq.add(t);
                }
                BufferedWriter w = new BufferedWriter(new
FileWriter(output));
                try {
                    while(pq.size()>0) {
                        BinaryFileBuffer fb = pq.poll();
                        fb.write(r);
                        w.newLine();
                        if(fb.empty()) {
                            fb.fbr.close();
                            fb.originalfile.delete();// we don't need you
anymore
                        } else {
                            pq.add(fb); // add it back
                        }
                    }
                } finally {
                    w.close();

                }

          }
}
```

Configuration Files:

Core-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<!-- Put site-specific property overrides in this file. -->
<configuration>
<property>
<name>fs.default.name</name>
<value>hdfs://ec2-52-201-238-108.compute-1.amazonaws.com:8020</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<value>/cache</value>
</property>
</configuration>
```

Hdfs-site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
```

<property>

<name>dfs.name.dir</name>

<value>/cache/name</value>

</property>

<property>

<name>dfs.data.dir</name>

<value>/cache/data</value>

</property>

</configuration>

Mapred-site.xml

<?xml version="1.0"?>

<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>

<name>mapreduce.jobtracker.address</name>

<value>hdfs://ec2-54-201-168-9.us-west-2.compute.amazonaws.com:8021</value>

<description>The host and port that the MapReduce job tracker runs at. if "local" then jobs are run as single map and reduce task </description>

</property>

<property>

<name>mapreduce.framework.name</name>

<value>yarn</value>

</property>

</configuration>

Yarn-site.xml

<?xml version="1.0"?>

<!--

Licensed under the Apache License, Version 2.0 (the "License");

you may not use this file except in compliance with the License.

You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software

distributed under the License is distributed on an "AS IS" BASIS,

WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

See the License for the specific language governing permissions and

limitations under the License. See accompanying LICENSE file.

-->

<configuration>

<!-- Site specific YARN configuration properties -->

<property>

<name>yarn.nodemanager.aux-services</name>

<value>mapreduce_shuffle</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>ec2-52-201-238-108.compute-1.amazonaws.com:8030</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>ec2-52-201-238-108.compute-1.amazonaws.com:8032</value>

</property>

<property>

<name>yarn.resourcemanager.webapp.address</name>

<value>ec2-52-201-238-108.compute-1.amazonaws.com:8088</value>

</property>

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>ec2-52-201-238-108.compute-1.amazonaws.com:8031</value>

</property>

<property>

<name>yarn.resourcemanager.admin.address</name>

<value>ec2-52-201-238-108.compute-1.amazonaws.com:8033</value>

</property>

</configuration>