Assignment Overview & Mapreduce Programming

CSCI4180

Qin Chuan

Mapreduce Program

```
public class MapRedProg {
// Define Mapper Class
        public static class MyMap extends Mapper<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {
                ......
        }
// Define Reducer Class
        public static class MyReduce extends Reducer<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {
                .......
        }
// Main Function, Job Configuration and Starting Point
        public static void main(String [] args) {
                ......
        }
}
```

Mapper Class

```
public static class MyMap extends Mapper<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {
//Define class fields if you like
        ......
        protected void setup(Context context) {
                //Execute ONE TIME at the beginning of the map task
        }
        protected void cleanup(Context context) {
                //Execute ONE TIME at the end of the map task
        }
        protected void map(KEY_IN key, VAL_IN val, Context context) {
                //Take input (key, val) pair to do map job
                //Execute MANY TIMES depend on the input
        }
}
```

Reducer Class

```
public static class MyReduce extends Reducer<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {
//Define class fields if you like
        ......
        protected void setup(Context context) {
                //Execute ONE TIME at the beginning of the reduce task
        }
        protected void cleanup(Context context) {
                //Execute ONE TIME at the end of the reduce task
        }
        protected void reduce(KEY_IN key, Iterable<VAL_IN> vals, Context context) {
                //Execute MANY TIMES depend on the number of keys
        }
}
```

Reducer Class

```
public static class MyReduce extends Reducer<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {

//public static class MyMap extends Mapper<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {
        //Define class fields if you like
        ......
        protected void setup(Context context) {
                //Execute ONE TIME at the beginning of the reduce task
        }
        protected void cleanup(Context context) {
                //Execute ONE TIME at the end of the reduce task
        }
        protected void reduce(KEY_IN key, Iterable<VAL_IN> vals, Context context) {
                //Execute MANY TIMES depend on the number of keys
        }
}
```

Job Configuration

```
public static void main(String [] args) throws Exception {
        Configuration conf = new Configuration();
        Job job = new Job(conf, "wordcount");
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);
        job.setJarByClass(WordCount.class);
        job.setMapperClass(Map.class);
        job.setCombinerClass(Reduce.class); // optional
        job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));
        job.waitForCompletion(true);
}
```

```
public static class MyReduce extends
Reducer<KEY_IN, VAL_IN,
KEY_OUT, VAL_OUT>
```

Type Matching

- KEY_IN, VAL_IN, KEY_OUT, VAL_OUT should all implement the Writable Interface.
  - e.g. **Text, FloatWritable, IntWritable** or implement the Writable Interface for customized type

- The output type of Mapper Class should be consistent with the input type of Reducer Class

- By default, the KEY_IN/VAL_IN of the mapper class is **LongWritable/Text** as each input pair represents a line number with the text in that line

Assignment 1

- Due on **Oct. 24**
- Configure VMs & Azure platform
- Write Java program
  - Word length count
  - N-gram count
  - N-gram relative frequency
- Test on the KJV & shakespeare data
- Do some optimizations

How to Compile?

- To compile a map-reduce program, the library required is the hadoop-core-*.jar
- On a machine without hadoop setting, get the hadoop-core-*.jar on one of your VM (The file should be under path ~/hadoop/)
- Compile as told in lec3.pdf

```
$ mkdir wordcount
$ javac -classpath /usr/local/hadoop/hadoop-core-*.jar WordCount.java -d wordcount
$ jar -cvf wordcount.jar -C wordcount/ .
```

Part 1

- Configure the hadoop on Openstack
- Start the hadoop service
- Compile the sample wordcount.java
- Run wordcount on the given data sets

Part 2

- Word Length Count
  - Instead of (word, count) pair, we focus on (length, count) pair. Eg, (3 5) means there are 5 words of length 3.
  - Pay attention to the **output type** of Mapper
- Words of Same Length
  - Eg. “who is it” - (3 1)(2 1)(2 1)
  - We might combine (3 1)**(2 1)(2 1)** into (3 1)**(2 2)**
  - Avoid too many emit pairs

Part 3

- N-gram Initial
  - Eg. N = 3, for “who is it” we have (w i i 1)
  - N-gram means N consecutive words
  - Initial means first character of the word
  - **Alphabet** means A-Z and a-z
- N-gram across Rows
  - Eg. “how can I finish this assignment on time without the help of my groupmates?”
  - N = 3, “on time without” should count (o t w 1) and “time without the” should count (t w t 1)

Pass Arguments

```
public class MapRedProg {
// Define Mapper Class
        public static class MyMap extends Mapper<KEY_IN, VAL_IN, KEY_OUT, VAL_OUT> {
                ......
                protected void map(KEY_IN key, VAL_IN val, Context context) {
                        Configuration conf = context.getConfiguration();
                        gram = Integer.parseInt(conf.get("ngram"));
                }
        }
// Main Function, Job Configuration and Starting Point
        public static void main(String [] args) {
                conf.set("ngram",args[2]);
                ......
        }
}
```

Part 4

- N-gram Initial Relative Frequency
  - Eg. N = 3 “who is it? We want to know”
  - How frequent is initial w followed initial i i?
  - (w i i 1)(w w t 1)(w t k 1)
  - RF(w i i) = ⅓ = 0.333
- Only Alphabet counts
  - Eg. (w > i 1)(w “ a 1) won’t count
  - You need to think about data structure to store intermediate data to compute RF

Part 5

- Redeem the Azure Code
- Create 4 VMs
- Install Hadoop and set the cluster
- Configure the hadoop
- Start the hadoop service
- Compile the sample wordcount.java
- Run wordcount on the given data sets

Data Set

- The two data sets on the course page is relatively smaller than the data set during the demo, just for you to test correctness.

- The largest data set less than 2G

- Time limit will be set reasonable, depending on the size of datasets.

Thank you