API Guide

Note: This document contains a tutorial for programmers to write an application. For implementation details, please see the **report.pdf**.

In order to make it convenient for the programmers to write the application for our framework, we mimic Hadoop's API. As you will see, the program looks very similar to an actual Hadoop application. Below is a tutorial for writing a simple wordcount application:

1.Mapper:

```
public class WordCountMapper extends Mapper {

        @Override
        public void map(String key, String value, Context context) {
                String[] toks = value.split(" ");
                for(String s : toks) {
                        context.write(s, "1");
                }
        }
}
```

The mapper is almost identical to the wordCount tutorial you can find anywhere, except that all the keys and values are required to be String. This limitation is explained in the "What we can improve" section in the report.

2.Reducer

```
public class WordCountReducer extends Reducer {

   @Override
   public void reduce(String key, Iterable<String> values, Context context) {
      int num = 0;
      for(String value:values){
         num += Integer.parseInt(value);
      }
      context.write(key,Integer.toString(num));
   }
}
```

The reducer is also fairly simple and looks no different to any other WordCount implementation. The only difference is that you need to convert integers into string explicitly yourself.

Another example is the InDegreeCount example, which is illustrated below:

Mapper:

```
public class InDegreeCountMapper extends Mapper {
   @Override
   public void map(String key, String value, Context context) {
      if(value != null && value.length() > 0 && value.charAt(0) != '#'){
         String[] fields = value.split("\t");
         if(fields.length == 2){
            String inNode = fields[1];
            context.write(inNode,"1");
         }
      }
   }
}
```

Reducer:

```
public class InDegreeCountReducer extends Reducer {
   @Override
   public void reduce(String key, Iterable<String> values, Context context) {
      int inDegree = 0;
      for(String value:values){
         inDegree += Integer.parseInt(value);
      }
      context.write(key,Integer.toString(inDegree));
   }
}
```

This example program takes <Scr,Dest> Node pairs as input, and count how many times the a Node appears as a Dest, that is the node's in degree.

In the mapper, we add a line of code to avoid potential errors. This is because the input file is found on the internet(http://snap.stanford.edu/data/wiki-Vote.html), and contains more information more than node pair records, so extra checking is necessary for us to filter out those redundant lines.

As you write the mapper and reducer, you should also write method to submit the job, here is how this is done:

```
public class WordCountMain {
   public static void main(String[] args){
      Job job = new Job();

      job.setMapClass("cmu.cs.distsystems.hw3.examples.WordCountMapper");
      job.setReduceClass("cmu.cs.distsystems.hw3.examples.WordCountReducer");

      job.setConfigFile(args[0]);

      job.setJar(args[1]);
      job.setInputDir(args[2]);
      job.setOutputDir(args[3]);

      job.setNumReducers(2);

      JobClient.submitJobAndWaitForCompletion(job);
   }
}
```

Here we first create a Job instance, and specify the Mapper and Reducer name. Then we set the config file path through job.setConfigFile method, because the JobClient will use this file to find the jobtracker and submit the job. Programmer should also specifies where the Jar files are in order for the task workers to find the entrance to the map and reduce function. The arguments afterwards are pretty straightforward.

To sum up, to write an application, the programmer should implement three components:
1.Mapper class (WordCountMapper.java)
2.Reducer class (WordCountReducer.java)
3.Program to submit the job.(WordCountMain.java)

Compile those files into a single Jar file, and execute the main function.
Note unlike Hadoop, the three programs should be put into separate java files.