Report of MapReduce Facility
Project 3 of 15-640
Kailiang Chen(kailianc) & Yang Pan(yangpan)

Development environment

This project is developed with Eclipse IDE for Java Developers, Luna Release (4.4.0), JDK 8u20. If you want to write a new example, implementing interface MRBase, you should work in the same environment. Details will be elaborated in the following tutorial.

Understand our examples

Example 1: WordCounter

This is a classic example in Map-Reduce, WordCounter. The map function breaks up the input value (the content of a file) into single words. And then for every word, construct a pair whose key is the word and value is "1". The reduce function sums up the occurrence of every word and output a pair whose key is the word and the value is the occurrence of that word.

You can see the source code in "Examples/WordCounter/WordCounter.java".

Example 2: Twitter

This example is a new Twitter information extraction. During the map phase, the userId and photoNum posted by this userId are extracted from the Twitter record line by line. During the reduce phase, the photoNum of the same userId is accumulated. The final result is two columns – left is userId, right is photoNum.

You can see the source code in "Examples/Twitter/Twitter.java".
Note: this example needs the support of "org.json" package because the input data is presented in json style.

Run our examples

The follow instructions will use “Twitter” as an example.

1. Run our system as the instructions in System Administrator Manual. Let the master and slaves be ready for work.
2. Create a directory in the root directory of map-reduce master named “Twitter”, and a directory “UserFiles” (or the name you specified in “UserDirName” option in config.txt) in “Twitter”.
3. Copy the “Examples/Twitter/Twitter.jar” and “Examples/Twitter/Twitter.txt” into “UserFiles”.
4. In the terminal of master, type “new Twitter”. Wait for a while (it needs some time because the input file is large) and the master would tell you where to find the result files.

Let our system work for you

Write your mapreduce class

Writing your own mapreduce class is easier than ever before. We provide a bash script to generate a template for you. To take this courtesy, follow the instructions.

1. Change directory to “ProgrammerTools/” in your terminal.
2. Type “./template.sh [TaskName]” and hit the enter.
3. Change directory to “ProgrammerTools/JavaFiles” and you will find “[TaskName].java”.
4. Open that file with any editor you like and implement the map and reduce function!

You will notice that we provide the prototype of map and reduce functions like:

```
void map(String key, String value, PairContainer output)
void reduce(String key,Iterator<String> values, PairContainer output)
```

We design the prototype according to the paper “MapReduce: Simplified Data Processing on Large Clusters”. For the map function, the key is the file name, and the value is the file’s content. The output is a PairContainer class, which is responsible for

collecting the result pairs of map function. You can put your result pair into it by invoking "`output.emit("key", "value");`". Or you can create an instance of Pair by "`Pair outPair = new Pair("key", "value");`" and then put it into the container by "`output.emit(outPair);`". For more information, please refer to the chapter of "I/O Library" of Report of MapReduce Facility.

For the reduce function, the values is an iterator of String. In our system, we merge the values with the same key into an array. So you only need to iterator through the values to access all the values with the same key, just like:

```
while (values.hasNext()) {
      String val = values.next();
}
```

Get your work run

After you finish the class, compile it so that our system can run your code!

1. Keep the java file (say, "Grep.java") under "ProgrammerTools/JavaFiles" directory.
2. Change directory to "ProgrammerTools/" in your terminal.
3. Type "./jargen.sh" and hit enter.

That's it. If no error occurs, you can find the "Grep.jar" file in "ProgrammerTools/JarFiles". That's what we need to get your work run in our system.

Follow the instructions in "Run our examples" to put the jar file and the input data file into the right place and run it in our system. Have fun!