# 91.673 Advanced Database Systems Project

Chang Liu
chang_liu@student.uml.edu
01516178

November 25, 2015

## 1 Overview

For this project, I have configured the Hadoop environment as well as the Eclipse with Hadoop-eclipse-plugin, then setting up the namenode and other server on local machines, start the eclipse and create a Map-Reduce project in Eclipse with separate packages named 'ex', 'ex2', 'ex3' and 'ex4'.

For each package, I used different schemes to handle the source data file like 'city.txt', 'country.txt' and 'countrylanguage.txt'. These different tasks require some handling of mapping and reducing procedure in the Hadoop. By using different Map() and Reduce() function, I can parse the input text file and accumulate the lists that meet the specific requirement. I will give detailed description below about how it works.

I think I have finished all the projects that required by the description, but it seems that there is some mess unicode display problem, I think it's related with local machine's character set, and all of my handling of the string used the unicode version and should be no problem.

## 2 General Hadoop Project

For a general Hadoop project, There is actually four components we should do:

1) Map class that implement a map function. In this map function, the program receive each item from the raw data file like 'city.txt', and then we need to parse the input, add judgement or string handling, emit the necessary key-pair data as the source of reduce function.

2) Reduce class that implement a reduce function. Similarly as above, the reduce accepts the items that shared the same key, and then we add some manipulating of data like counting the number, concatenating the value string or other operations to generate the output.

3) A run function that configures the job, like the input or output of the Mapper/Reducer class, source data path.

4) A main function that starts the program and execute the run() function.

After these four components are done, I can start the running of the source program, with the hadoop server running in the backend. Then it will generate the results and save to the corresponding file we set in the source code.

## 3 Selection

For this first task, it's quite simple, I just need to parse the 'city.txt' file, get the each item in the lists, this is done by the java function 'String.split'. After we parse the data and get city name as well as the population, add a judgement for comparing the number with 300,000, only emits the items that are larger than the value, so the map function has finished.

In the reduce function, the program should accept all the item and just save the key value, since the key is the city name, and that's the only items that needs to be saved or output.

In the run function, set the configurations of input for Map and Reduce, here we both used the type 'Text' since it's the string.

## 4 Projection

For the second task, similar to the previous, the only different is that we emit the city name and the district as the output of map function, in the reduce function, we should connect all the string belonging to the same key, since we should output the city and its corresponding district.

## 5 Natural Joint

For natural joint, here there are two tables we need to parse, the 'country.txt' that stores the country code and its name and the 'countrylanguage.txt' that stores the country with its language.

First we judge the type of data, and know which table it is from, then emit only the English country with key-pair of the countrycode-NULL from language table. For the country table, emit all the countrycode-name pair for fetching the name when we know some English country.

In the reduce process, we have two types of data as the above key-pair shows sorted by the key, we only count the key that has more than 1 lists, because it means the it contains the country-NULL item and it's the English country, then by the value of the other list we know the name of the country, save it to the output. For only 1 list, it is from country table, and no English country is there, we just omit these records.

## 6 Aggregation

Similar to previous experiment, only use the 'city.txt' file and emit the district-city pair, then in the reduce function, by the key of district, count the number of the cities, output the district and its count, that forms the outputs of the total results.

## 7 Summary

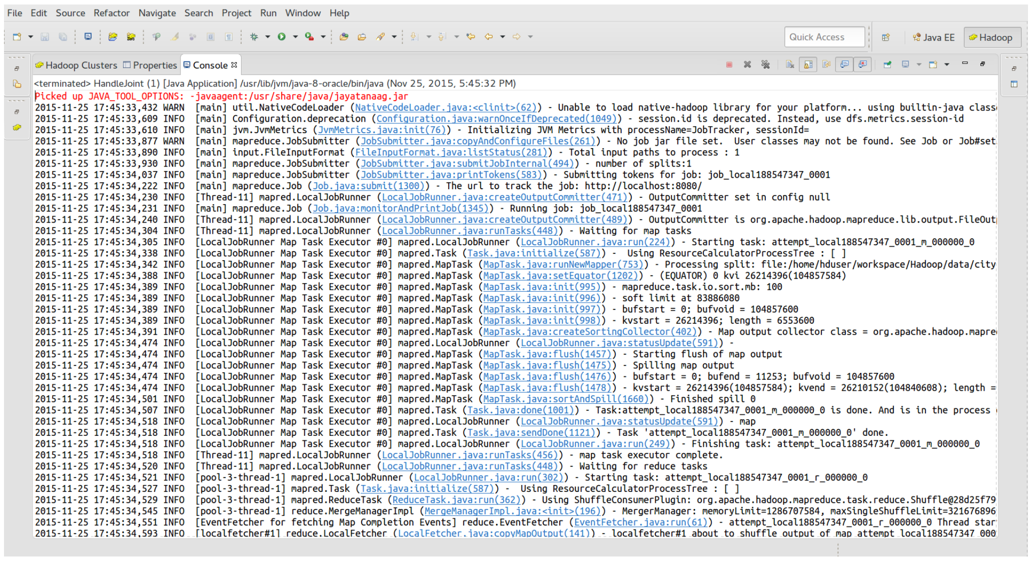
I think these project are quite straight-forward. The main work lays on the setting up of the environment and getting a rough idea of how MapReduce program works, then by the knowledge from class, devise the correct map and reduce function to deal with data, we can get the correct outputs. I learned a lot from the books and it helps me a lot when starting the project and get to know how it works and how the program framework should be.

## 8 Reference

Install Guide: `https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/SingleCluster.html`
Books: <Hadoop in Action>

## 9 Screenshot

All the running of program is similar so I just select the first output in eclipse, the final result is not output here, they're store under 'ex*/part-r-00000'. For the final result, please check the output folder in my submission, they have stored all the necessary items.


File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java EE
Hadoop
Hadoop Clusters
Properties
Console

```
<terminated> HandleJoint (1) [Java Application] /usr/lib/jvm/java-8-oracle/bin/java (Nov 25, 2015, 5:45:32 PM)
Picked up JAVA_TOOL_OPTIONS: -javaagent:/usr/share/java/jayatanaag.jar
2015-11-25 17:45:33,432 WARN  [main] util.NativeCodeLoader (NativeCodeLoader.java:<clinit>(62)) - Unable to load native-hadoop library for your platform... using builtin-java class
2015-11-25 17:45:33,609 INFO  [main] Configuration.deprecation (Configuration.java:warnOnceIfDeprecated(1049)) - session.id is deprecated. Instead, use dfs.metrics.session-id
2015-11-25 17:45:33,610 INFO  [main] jvm.JvmMetrics (JvmMetrics.java:init(76)) - Initializing JVM Metrics with processName=JobTracker, sessionId=
2015-11-25 17:45:33,877 WARN  [main] mapreduce.JobSubmitter (JobSubmitter.java:copyAndConfigureFiles(261)) - No job jar file set.  User classes may not be found. See Job or Job#set
2015-11-25 17:45:33,890 INFO  [main] input.FileInputFormat (FileInputFormat.java:listStatus(281)) - Total input paths to process : 1
2015-11-25 17:45:33,930 INFO  [main] mapreduce.JobSubmitter (JobSubmitter.java:submitJobInternal(494)) - number of splits:1
2015-11-25 17:45:34,037 INFO  [main] mapreduce.JobSubmitter (JobSubmitter.java:printTokens(583)) - Submitting tokens for job: job_local188547347_0001
2015-11-25 17:45:34,222 INFO  [main] mapreduce.Job (Job.java:submit(1300)) - The url to track the job: http://localhost:8080/
2015-11-25 17:45:34,230 INFO  [Thread-11] mapred.LocalJobRunner (LocalJobRunner.java:createOutputCommitter(471)) - OutputCommitter set in config null
2015-11-25 17:45:34,231 INFO  [main] mapreduce.Job (Job.java:monitorAndPrintJob(1345)) - Running job: job_local188547347_0001
2015-11-25 17:45:34,240 INFO  [Thread-11] mapred.LocalJobRunner (LocalJobRunner.java:createOutputCommitter(489)) - OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOut
2015-11-25 17:45:34,304 INFO  [Thread-11] mapred.LocalJobRunner (LocalJobRunner.java:runTasks(448)) - Waiting for map tasks
2015-11-25 17:45:34,305 INFO  [LocalJobRunner Map Task Executor #0] mapred.LocalJobRunner (LocalJobRunner.java:run(224)) - Starting task: attempt_local188547347_0001_m_000000_0
2015-11-25 17:45:34,338 INFO  [LocalJobRunner Map Task Executor #0] mapred.Task (Task.java:initialize(587)) -  Using ResourceCalculatorProcessTree : [ ]
2015-11-25 17:45:34,342 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:runNewMapper(753)) - Processing split: file:/home/hduser/workspace/Hadoop/data/city
2015-11-25 17:45:34,388 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:setEquator(1202)) - (EQUATOR) 0 kvi 26214396(104857584)
2015-11-25 17:45:34,389 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:init(995)) - mapreduce.task.io.sort.mb: 100
2015-11-25 17:45:34,389 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:init(996)) - soft limit at 83886080
2015-11-25 17:45:34,389 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:init(997)) - bufstart = 0; bufvoid = 104857600
2015-11-25 17:45:34,389 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:init(998)) - kvstart = 26214396; length = 6553600
2015-11-25 17:45:34,391 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:createSortingCollector(402)) - Map output collector class = org.apache.hadoop.mapre
2015-11-25 17:45:34,474 INFO  [LocalJobRunner Map Task Executor #0] mapred.LocalJobRunner (LocalJobRunner.java:statusUpdate(591)) -
2015-11-25 17:45:34,474 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:flush(1457)) - Starting flush of map output
2015-11-25 17:45:34,474 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:flush(1475)) - Spilling map output
2015-11-25 17:45:34,474 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:flush(1476)) - bufstart = 0; bufend = 11253; bufvoid = 104857600
2015-11-25 17:45:34,474 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:flush(1478)) - kvstart = 26214396(104857584); kvend = 26210152(104840608); length =
2015-11-25 17:45:34,501 INFO  [LocalJobRunner Map Task Executor #0] mapred.MapTask (MapTask.java:sortAndSpill(1660)) - Finished spill 0
2015-11-25 17:45:34,507 INFO  [LocalJobRunner Map Task Executor #0] mapred.Task (Task.java:done(1001)) - Task:attempt_local188547347_0001_m_000000_0 is done. And is in the process
2015-11-25 17:45:34,518 INFO  [LocalJobRunner Map Task Executor #0] mapred.LocalJobRunner (LocalJobRunner.java:statusUpdate(591)) - map
2015-11-25 17:45:34,518 INFO  [LocalJobRunner Map Task Executor #0] mapred.Task (Task.java:sendDone(1121)) - Task 'attempt_local188547347_0001_m_000000_0' done.
2015-11-25 17:45:34,518 INFO  [LocalJobRunner Map Task Executor #0] mapred.LocalJobRunner (LocalJobRunner.java:run(249)) - Finishing task: attempt_local188547347_0001_m_000000_0
2015-11-25 17:45:34,518 INFO  [Thread-11] mapred.LocalJobRunner (LocalJobRunner.java:runTasks(456)) - map task executor complete.
2015-11-25 17:45:34,520 INFO  [Thread-11] mapred.LocalJobRunner (LocalJobRunner.java:runTasks(448)) - Waiting for reduce tasks
2015-11-25 17:45:34,521 INFO  [pool-3-thread-1] mapred.LocalJobRunner (LocalJobRunner.java:run(302)) - Starting task: attempt_local188547347_0001_r_000000_0
2015-11-25 17:45:34,527 INFO  [pool-3-thread-1] mapred.Task (Task.java:initialize(587)) -  Using ResourceCalculatorProcessTree : [ ]
2015-11-25 17:45:34,529 INFO  [pool-3-thread-1] mapred.ReduceTask (ReduceTask.java:run(362)) - Using ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@28d25f79
2015-11-25 17:45:34,545 INFO  [pool-3-thread-1] reduce.MergeManagerImpl (MergeManagerImpl.java:<init>(196)) - MergerManager: memoryLimit=1286707584, maxSingleShuffleLimit=321676896
2015-11-25 17:45:34,551 INFO  [EventFetcher for fetching Map Completion Events] reduce.EventFetcher (EventFetcher.java:run(61)) - attempt_local188547347_0001_r_000000_0 Thread star
2015-11-25 17:45:34,593 INFO  [localfetcher#1] reduce.LocalFetcher (LocalFetcher.java:copyMapOutput(141)) - localfetcher#1 about to shuffle output of map attempt_local188547347_000
```