(Hao Zhang, WustlKey: h.zhang633, ID: 452003), (Hanming Li, WustlKey: lihanming, ID: 451802)

Homework 4

Problem 1

(a) there are not any differences in the job execution.

(b) there are not any differences in the job execution because I set caseSensitive true by default. So the result is the same to the previous one.

(c) Hadoop jar wc.jar solution.AvgWordLength -D caseSensitive=false shakespeare avgResult

```
[training@localhost src]$ hadoop jar wc.jar solution.AvgWordLength -D caseSensit
ive=false shakespeare avgResult
```

The order of command line inputs matters. The implementation cannot be run and a useful error message will be printed:

```
[training@localhost src]$ hadoop jar wc.jar solution.AvgWordLength shakespeare -
D caseSensitive=false avgResult
Usage: AvgWordLength [-D caseSensitive=true|false] <input dir> <output dir>
```

a 3.275899648342265

w 4.373096283946263

z 5.053333333333334

(Hao Zhang, WustlKey: h.zhang633, ID: 452003), (Hanming Li, WustlKey: lihanming, ID: 451802)

Problem2:

(a) Data format: IP address, operation time, operation type, required file, server's response code and length of file.

Data records: each hit made from unique IP address.
Each record contains one piece of detailed information about each hit like address and the time.
The fraction is 5000/4477843=0.12%

(b) map input:

10.223.157.186 - - [15/Jul/2009:21:24:17 -0700] "GET /assets/img/media.jpg HTTP/1.1" 200 110997

10.223.157.186 - - [15/Jul/2009:21:24:18 -0700] "GET /assets/img/pdf-icon.gif HTTP/1.1" 200 228

10.216.113.172 - - [16/Jul/2009:02:51:28 -0700] "GET / HTTP/1.1" 200 7616

10.216.113.172 - - [16/Jul/2009:02:51:29 -0700] "GET /assets/js/lowpro.js HTTP/1 .1" 200 10469

10.216.113.172 - - [16/Jul/2009:02:51:29 -0700] "GET /assets/css/reset.css HTTP/1.1" 200 1014

map output:

(10.223.157.186, 1)

(10.223.157.186, 1)

(10.216.113.172, 1)

(10.216.113.172, 1)

(10.216.113.172, 1)

reduce input：

(10.223.157.186, [1, 1])

(10.216.113.172, [1, 1, 1])

reduce output：

(10.223.157.186, 2)

(10.216.113.172, 3)

reducer sums up the total access time.

Seen this in word count.

(Hao Zhang, WustlKey: h.zhang633, ID: 452003), (Hanming Li, WustlKey: lihanming, ID: 451802)

(e) (i) when testing the program locally, the input and output of both mapper and reducer are in the local paths, but for running it on the cluster, the input of mapper are from the HDFS data nodes, and the output of mapper are in local and transfer them to reducer as the input locally, the output of reducer are transferred to the HDFS service.

The print statement of testing locally is to stderr, because it prints the output directly, but running on the cluster is to stdout.

Testing locally does not require any Hadoop daemons be running. Running on cluster uses YARN, which has 4 types of daemons, which are ResourceManager, ApplicationMaster, NodeManager and JobHistory.

(ii) The commend line is:

$ Hadoop jar ProcessLogs.jar stubs.ProcessLogs -fs=file:/// -jt=local /home/training/Downloads/test_access_log /home/training/workspace/log_file_analysis/commendout

(iii) I prefer running the program using the commend line, because when I apply the toolrunner in the code, it only need a one line commend to run the job, but to run it with eclipse, I have to add the external libraries and set the run configuration every time.

(f) As the output of the testlog shows, there are 10 different IP address in the testlog data. There are 5000 lines in the testlog file, the total count of the output is 1+21+12+547+18+21+2860+1196+115+209= 5000, so we can say that every line in testlog contribute to a count.

(g) The test on local has some limitations: Distributed Cache does not work, the job can only specify a single reducer and some 'beginner' mistakes may not be caught, all this limitation may cause running error or other problems when running on cluster.

There are 333923 different IP address in the weblog data.

The numbers of hits of the following IP addresses are:

10.1.100.199: 35hits;

10.1.100.5: 1hit;

10.99.99.58: 21hits.

The IP addresses are sorted because when they are handled by the mapper, the output key-value pairs are sorted and grouped by the master controller.