Upload java file to clusters in Local terminal.

scp WordCountImproved.java [wsa50@cluster.cs.sfu.ca](mailto:wsa50@cluster.cs.sfu.ca):

Then in clusters, create .class and .jar files.

javac -classpath `hadoop classpath` WordCount.java

jar cf wordcount.jar WordCount\*.class

(when including other files like in combiner&reducer part, we use modified-version commands here) mention the file path!

then, run on cluster:

yarn jar wordcount.jar WordCount wordcount-1 output-1

check output:

hdfs dfs -ls output-1

hdfs dfs -cat output-1/part-r-00000 | less

also, can remove (and then rerun):

hdfs dfs -rm -r output-1

output-4, output-5 for WordCountImproved

output-6 for refined words

output-7 for RedditAverage w/o combiner

output-8 for RedditAverage w/ combiner

Q for exit output page in Terminal.

* Way to copy/paste/upload file into the terminal:

vi test\_file.txt

then, i for insert.

To finish insert and save, write :wq

Or just exit, wirte :q

Then, in the terminal, check the file:

cat test\_file.txt

ls

When refer to “running time”:

time yarn jar wordcount.jar WordCount wordcount-1 output-1

or:

time yarn jar a1.jar RedditAverage -libjars json-20180813.jar /courses/732/reddit-1 output-7

or:

check localhost hadoop

**Writable**

https://www.linkedin.com/pulse/writable-writablecomparable-hadoop-prateek-kumar

w/o combiner:

w/ combiner:

real 0m25.698s (look at this time!)

user 0m6.203s

sys 0m0.493s

Q1:

Clear about WordCount now. Just need more time to play with Java datatypes.

Q2:

When reducer = 1, the process results in all 319553 data output from just one reducer. All outputs are grouped by same keys with counts summed up and are listed in alphabetic order. But when we submit with -D mapreduce.job.reduces=3, the number of reducer increases to 3. Now the large output is separated into three nearly equal sizes and processed by three reducers respectively, with these three parts of word results grouped by same keys in alphabetic order and their counts summed up in each reducer.

With multiple reducers, it’s safer to process with large output sets. Because other reducers would keep processing even if one reducer failed. Besides, multiple reducers offer faster processing time than only one reducer does.

Q3:

When reducer = 0, the outputs are returned in three mappers. The words not shuffled and “reduced”. They just come directly from mapping process. The output is just randomly split words that are just scattered with 1 count each and not grouped and not summed up in alphabetic order.

Q4:

With a combiner optimization before reducer on large datasets, the running time is faster. (25.698s vs. 24.886s) -> saves shuffle time.