PDP - Assignment 1 - Hadoop / HDFS

Student: Caitlyn Smith
Number: 644844
Date: 25-06-2022
Version 1
Github url:
https://github.com/C-Smith-InHolland/HadoopAssignmentsYear3/tree/main/assignment1

The Code:
Import the libraries

```
from mrjob.job import MRJob
from mrjob.step import MRStep
```

Defining the steps. The first step consists of a mapper, a combiner and a reducer and the second step contains a reducer. The purpose of the first step is to get all the ratings and count per movie how many ratings it got. The second step is to sort the data.

```
def steps(self):
    return [
        MRStep(mapper=self.mapper_get_ratings,
               combiner=self.combine_ratings_count,
               reducer=self.reducer_count_ratings
        ),
        MRStep(
              reducer=self.reducer_sort_counts
       )
    ]
```

Read the data. The data consists of a column with userID, movieID, rating and the timestamp. We only need the movieID for now. The yield returns the movieID along with a 1 indicating it is 1 rating.

```
def mapper_get_ratings(self, _, line):
    (userID, movieID, rating, timestamp) = line.split('\t')
    yield movieID, 1
```

The first function returns the key (or the movieID) and the sum of all the occurrences of that movie ID. The second function works similar to the first but returns it in a different format so it can be sorted from high to low in the last step.

```
  def combine_ratings_count (self, key, values):
      yield key, sum(values)

  def reducer_count_ratings(self, key, values):
      yield None, (sum(values), key)
```

The last and second reducer is used to sort the movies on how many ratings they have from high to low. For this it goes through the entire list, sorts it and outputs the movieID and the amount of ratings it received

```
def reducer_sort_counts(self, _, ratings_counts):
    for count, key in sorted(ratings_counts, reverse=True):
        yield key, count
```

To run it:
ssh into hadoop virtual machine

download u.data file and the sort_ratings.py file

Run command:
$ python sort_ratings.py u.data

The result:
In the results we first have the movieID column and then the amount of ratings for that movie. The data is sorted from high to low based on the number of ratings.

The top results

```
No configs found; falling back on auto-configuration
Creating temp directory /tmp/HD_rating.maria_dev.20220625.125923.389163
Running step 1 of 2...
Running step 2 of 2...
Streaming final output from /tmp/HD_rating.maria_dev.20220625.125923.389163/output...
"50"    583
"258"   509
"100"   508
"181"   507
"294"   485
"286"   481
"288"   478
"1"     452
"300"   431
"121"   429
"174"   420
"127"   413
"56"    394
"7"     392
"98"    390
"237"   384
"117"   378
"172"   367
"222"   365
"313"   350
"204"   350
"405"   344
"79"    336
"210"   331
"151"   326
"173"   324
"69"    321
"748"   316
"168"   316
"269"   315
"257"   303
"195"   301
"423"   300
"9"     299
"318"   298
"276"   298
"302"   297
"22"    297
"96"    295
"328"   295
"25"    293
"15"    293
"118"   293
"183"   291
"216"   290
"176"   284
"64"    283
"234"   280
"202"   280
"28"    276
"191"   276
"89"    275
"111"   272
"275"   268
"742"   267
"12"    267
"357"   264
```

The bottom results

```
"1593"  1
"1587"  1
"1586"  1
"1584"  1
"1583"  1
"1582"  1
"1581"  1
"1580"  1
"1579"  1
"1577"  1
"1576"  1
"1575"  1
"1574"  1
"1572"  1
"1571"  1
"1570"  1
"1569"  1
"1568"  1
"1567"  1
"1566"  1
"1565"  1
"1564"  1
"1563"  1
"1562"  1
"1561"  1
"1559"  1
"1557"  1
"1548"  1
"1546"  1
"1543"  1
"1536"  1
"1533"  1
"1526"  1
"1525"  1
"1520"  1
"1515"  1
"1510"  1
"1507"  1
"1505"  1
"1498"  1
"1494"  1
"1493"  1
"1492"  1
"1486"  1
"1482"  1
"1476"  1
"1461"  1
"1460"  1
"1458"  1
"1457"  1
"1453"  1
"1452"  1
"1447"  1
"1414"  1
"1373"  1
"1366"  1
"1364"  1
"1363"  1
"1352"  1
"1349"  1
"1348"  1
"1343"  1
"1341"  1
"1340"  1
"1339"  1
"1329"  1
"1325"  1
"1320"  1
"1310"  1
"1309"  1
"1236"  1
"1235"  1
"1201"  1
"1156"  1
"1130"  1
"1122"  1
Removing temp directory /tmp/HD_rating.maria_dev.20220625.125923.389163...
```